Interactive Knowledge Discovery over Web of Data

Mehwish Alam

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Découverte interactive de connaissances dans le web des données.

Interactive Knowledge Discovery over Web of Data.

THÈSE

présentée et soutenue publiquement le 1^{er} Décembre 2015

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Mehwish Alam

Composition du jury

Président :	Claire Gardent	Directrice de recherche CNRS, LORIA
Rapporteurs :	Sébastien Ferré	Maître de conferences (HDR), Université de Rennes 1
	Karell Bertet	Maître de conferences (HDR), Université de La Rochelle
Examinateurs :	Sebastian Rudolph	Professor, University of Dresden, Germany
	Nathalie Aussenac-Gilles	Directrice de recherche CNRS, IRIT, Toulouse
Directeurs :	Amedeo NAPOLI	Directeur de recherche CNRS, LORIA
Co-Directeurs :	Malika Smail-Tabbone	Maître de conferences (HDR), Université de Lorraine, LORIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

First of all, I would like to express my gratitude to my advisor and mentor Amedeo NAPOLI for the continuous support during my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in research and writing of this thesis. I could not have imagined having a better mentor for my Ph.D study. Along with Amedeo NAPOLI, I would like to thank my co-supervisor Malika SMAIL-TABBONE and Adrien COULET for their support during my thesis.

Besides my advisor, I would like to thank my thesis committee: Karell BERTET, Sébastien FERRE, Sebastian RUDOLPH, Nathalie AUSSENAC-GILLES and Claire GARDENT for their detailed report, encouragement, insightful comments, questions and valuable suggestions.

A very special thanks to my co-authors, Aleksey BUZMAKOV, Victor CODOCEDO and Matthieu OSMUK for working with me and exchanging different ideas for making my work being applicable to broader field of research and providing a different perspective to my work. They were knowledgeable, fascinating, engaged, and friendly— in short, they were tremendous colleagues in every sense.

I want to thank all the past and present members of the Orpailleur team for their friendly, positive, and helpful attitude. I am in particular grateful to Elias EGHO and Sébastien DA-SILVA, whose suggestions and support has been extremely important during my research activities. Thanks to my colleagues and friends Laura INFANTE-BLANCO and Shashi NARAYAN who have been with me for better and worse parts of my life. I am grateful to all other members of the team for being such good colleagues and friends. I would like to thank my family, my parents for supporting me throughout my life and my siblings who have been guiding me through my whole life.

This work would not have been possible without the financial and logistic support of Inria and University of Lorraine. A special thanks for the continuous support and precious help provided from the very first day of my experience in the team.

To my family,

Contents

Appendix B
Enriching Transcriptomic Data with Linked Open Data

Introduction

Currently, the WWW contains huge amount of web documents in the form of "human processable" HTML or text documents. In the last years, due to the efforts of Semantic Web community the "Web of Documents" is turning into the "Web of Data" i.e., the documents are annotated in the form of entity and relationship making the human processable data directly processable by machines. Over the past few years, there has been a huge explosion in the on-line publication of data in the form of entities and relationships linked together, giving rise to a huge cloud containing billions of RDF triples called Linked Data Bizer et al. (2009a). There is a desperate need to use and mold existing methodologies to target the challenges faced by this representation. This thesis contributes to using knowledge discovery process for solving problems encountered by machine processable Linked Data. Here, we introduce the notion of Knowledge Discovery in Databases (KDD), then we give a brief introduction to exploratory data mining. Finally, we discuss how exploratory data mining can be applied to Web of Data. During this thesis, we discuss overall architecture of web clustering engines and briefly discuss the studies which use formal concept analysis as the clustering algorithm for clustering web search results. We divide these studies into two parts, the first part discusses the systems allowing basic user navigation over the clusters and the second part discusses the studies using interactive exploration over the clusters. Afterwards, the details of existing systems following the architecture of web clustering engines for clustering SPARQL query answers. We study three research directions, i.e., 1) Creating views over RDF graphs by clustering SPARQL query answers and allowing user interaction over clustered answers through RV-Xplorer, 2) assessing the quality and correcting RDF triples created by automated conversion of Wikipedia and finally, 3) allowing simultaneous navigation/exploration over distributed resources over Linked Data by directly clustering RDF triples instead of clustering only SPARQL query answers. In this thesis we discuss the solution to each of the problems described before.

1 Knowledge Discovery in Databases

A large amount of data has produced belonging to all the fields such as biomedical data, customer data, financial data etc. It is very important to create new algorithms and processes to effectively extract knowledge from the existing data or using the existing one. Knowledge Discovery in Databases (KDD) is mainly concerned with providing a number of techniques for making sense of these data. The main goal of the KDD process is to map low-level data which is usually large in volume into compact representation which are easily interpretable such as patterns. Multiple algorithms for discovering unknown hidden patterns are used for traversing from data to knowledge which may be used for understanding or prediction. The phrase KDD was first introduced in Fayyad et al. (1996a,b) to emphasize on the fact that knowledge is the end product of the data driven discovery. Figure 1 depicts the KDD process step by step. Following is the

description of each phase of the KDD process:

Selection. The selection process develops a clear understanding of the application domain and the relevant prior knowledge. Identification of the goal of the KDD process from the user point-of-view is also an important task. Based on this understanding and overall plan the dataset or subset of the data set is selected on which a variety of KDD tasks are performed.

Preprocessing. During the preprocessing step the data is cleaned and processed. It includes noise reduction if needed and collecting necessary information through which noise can be taken into account and modeled. It also includes designing a way to deal with missing data fields.

Transformation. includes data reduction and projection which focuses on finding useful features representing the data depending on the goal. Dimensionality reduction or transformation methods are used to reduce the effective number of variables under consideration or to find invariant representations for the data. After the transformation of data, the goal of the KDD process should be identified i.e., whether it is summarization, classification, clustering or prediction. This process includes selecting methods to be used for searching for patterns in the data and deciding which models and parameters may be appropriate. Finally, matching a particular data mining method with the overall criteria of the KDD process.

Data Mining. allows searching the patterns of interest in a particular representational form (or a set of such representations) such as classification rules or trees, regression functions, clusters etc.

Interpretation and Evaluation. The final step involves the interpretation of the mined patterns. This step can also involve visualization techniques aiding the target user in interpretation. This knowledge can then be further fed to another system or it can simply be documented to the interested parties.

2 Exploratory Data Mining

Exploratory Data Mining van Leeuwen (2014) aims at summarizing the main characteristics of the data for analysis purposes. Usually, visualization methods are used for performing such kind of analysis. It allows the user to explore the data for formulating hypothesis by giving an insight into the data. One of the major challenges is to find a way to only process user/task specific information by directly involving the user in the KDD process. This is only possible by combining data mining algorithms with visualization and human-computer-interaction. After the patterns are discovered using data mining algorithms, the paradigm of interactive data exploration enables the user to provide feedback to the system. The goal is to make pattern mining practically more useful, by enabling the user to interactively explore the data and identify interesting structure. This way, the resulting patterns will be more relevant and interesting to the user. Another challenge is to generate algorithms which perform well under restricted resources because the existing technique are usually computationally intensive. Finally, these methods are only applied to smaller datasets.

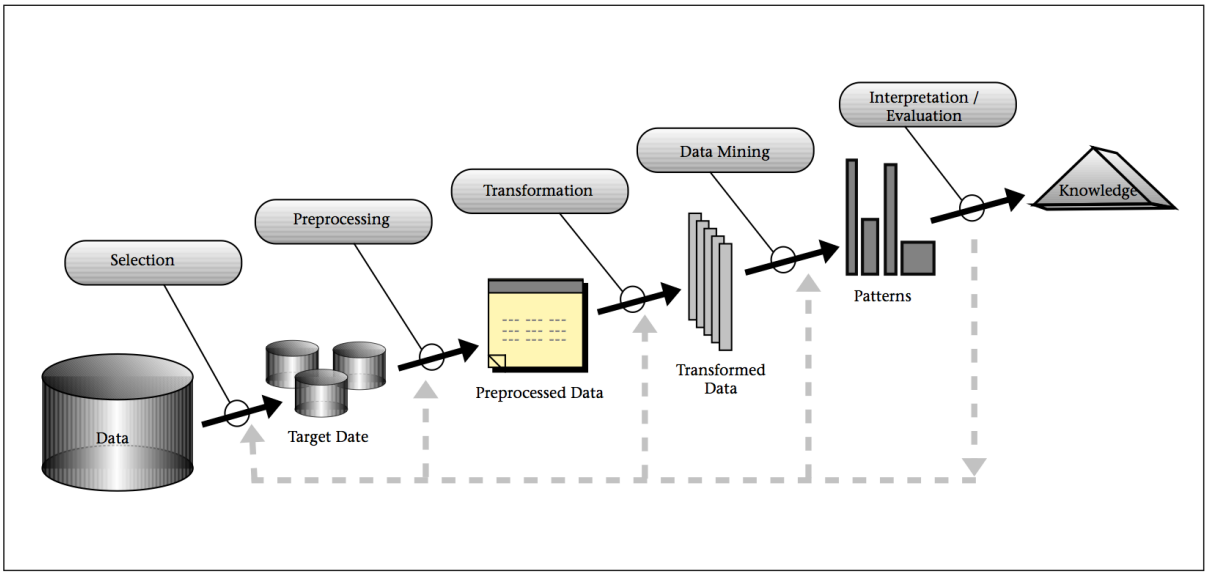


Figure 1: Knowledge Discovery in Databases Fayyad et al. (1996a,b)

3 Exploratory Data Mining and Web of Data

Recently, there has been a substantial growth in the interest of publishing data on-line in the form of entity-relationship i.e., Resource Description Framework (RDF)[1] and in the form of ontologies. There has been an explosion of data sources published in RDF and then linked to other data sources called Linked Data (LD) Bizer et al. (2009a). It provides a set of principles to be followed for publishing data in this format. These data are then accessible through many ways i.e., in the form of RDF dumps, crawling from one entity to another by following edges and through querying using standard query language SPARQL. This increased interest in the publication of data in RDF format gives rise to many interesting challenges.

One of the challenges faced by web search engines is that they retrieve a long list of answers. The user has to sift through all the answers to retrieve the relevant ones. To solve this problem Web Clustering Engines (WCE) Carpineto et al. (2009) were introduced. WCE basically allow the user to send query to search engine and cluster the ranked results (documents) for displaying it to the user. Such a clustering helps the user in selecting only those web pages which are more related to her needs. Previously, we mentioned that *"Web of Documents"* has turned into a *"Web of Data"* which is accessible through SPARQL queries. The answers generated by SPARQL query are in the form of a list. Even if the user gets hundreds of answers she is unable to learn the structure of these answers or find the hidden patterns. She faces a similar problem of sifting through thousands of irrelevant answers to get the relevant ones. Moreover, the navigation through this sea of answers creates problems of interpretation. In this thesis we target this problem and provide user with navigational capabilities over these answers by following the KDD process using FCA as a clustering algorithm. We also discuss implementations of a visualization tool which allows guided navigation and user interaction.

One of the major issues with the crowd-sourced resources on Linked Data such as DBpedia is that they contain some missing information. DBpedia is the RDF version of Wikipedia info boxes. It also uses Wiki-categories i.e., the hierarchy of categories introduced by Wikipedia for

[1] `http://www.w3.org/RDF/`

organizing Wikipedia articles. In the current thesis we use association rule mining to complete RDF-based descriptions of DBpedia using Wiki-categories.

In the past there have been several studies which apply FCA to RDF graphs such as Kirchberg et al. (2012) and d'Aquin and Motta (2011). Both the studies directly apply standard FCA to deal with such a structure and do not directly target the heterogeneity present in such a data due to tree structure present in RDF Schema or ontology (we call it background knowledge). Following this paradigm, RDF triples can be clustered w.r.t. this background knowledge. There are several approaches which provide document clustering with the help of Formal Concept Analysis using background knowledge. We use these methodologies for providing navigation and exploration over the part of RDF graph which are relevant to the user. This approach constitutes a way of clustering SPARQL query answers based on background knowledge, where the background knowledge is contained in the Knowledge Base over which the SPARQL query is posed.

Here it can be remarked that each approach described above follows exactly the process of KDD i.e., *selection* where we allow the user to select the dataset to be queried, resources defined by domain expert which are necessary for domain specific applications, data sets containing incompletion, raw data sets to be enriched by background knowledge etc. These data sets are then *preprocessed* and *transformed* to be used by *Formal Concept Analysis*. This generates patterns which are then *visualized* for *interpretation* purposes. The visualization tool proposed in this thesis allows user interaction and feedback for providing better understanding of that data, answers to the user questions and refining the obtained results to only relevant ones.

4 Contributions

This section generally discusses each of the contributions detailed in the rest of the thesis one by one. These contributions provide the solution to each of the challenges described in the previous section. This thesis targets three research directions. First part uses Formal Concept Analysis for defining views over RDF graphs with the help SPARQL queries. It also gives detailed description of the tool developed for navigating these views for interpretation purposes along with the comparison to existing approaches. The second part of the thesis deals with mining Description Logic (DL) concept definitions from DBpedia using association rule mining. The third part allows interactive exploration over several RDF resources from one platform. The RDF triples contained in these resources are clustered using background knowledge. Finally, these clusters can be navigated for analysis, interpretation and information retrieval purposes Baeza-Yates and Ribeiro-Neto (1999).

Creating Views over SPARQL Query: As described above one of the ways of accessing Linked Data is through SPARQL queries. These SPARQL queries over semantic web data usually produce list of tuples as answers that may be hard to understand and interpret. For solving this problem, an FCA based framework is proposed namely Lattice-Based View Access (LBVA) Alam and Napoli (2014a). This framework provides a classification of the answers of SPARQL queries based on a concept lattice, that can be navigated for retrieving or mining specific patterns in query results. This concept lattice can be seen as a *materialized view* over the underlying knowledge base which is accessed with the help of SPARQL. Finally, this view keeping the classification of the answers is visualized to allow user interaction and help the user in interpreting the obtained classes. Moreover, implications are extracted from this view which are termed as the knowledge units.

With the help of `View By` clause a concept lattice is created as an answer to the SPARQL query which can then be visualized and navigated using RV-Xplorer (Rdf View eXplorer).

Accordingly, this paper discusses the support provided to the expert for answering certain questions through the navigation strategies provided by RV-Xplorer. Moreover, the paper also provides a comparison existing state of the art approaches.

Completing RDF Data: The popularization and quick growth of Linked Open Data (LOD) has led to challenging aspects regarding quality assessment and data exploration of the RDF triples that shape the LOD cloud. Particularly, we are interested in the completeness of the data and their potential to provide concept definitions in terms of necessary and sufficient conditions. Hence, we propose a novel technique based on Formal Concept Analysis which organizes RDF data into a concept lattice. This allows data exploration as well as the discovery of implication rules which are used to automatically detect missing information and then to complete RDF data. Moreover, this is a way of reconciling syntax and semantics in the LOD cloud. Experiments on the DBpedia knowledge base show that the approach is well-founded and effective.

Revisiting Pattern Structures for Structured Attribute Sets: During this work, we revisit an original proposition on pattern structures for structured sets of attributes. There are several reasons for carrying out this kind of research work. The original proposition does not give many details on the whole framework, and especially on the possible ways of implementing the similarity operation. There exists an alternative definition without any reference to pattern structures. we would like to make a parallel between two original propositions i.e., Carpineto and Romano (1996b, 2004a) and Ganter and Kuznetsov (2001a). Moreover we discuss an efficient implementation of the intersection operation in the corresponding pattern structure. Finally, we discovered that pattern structures for structured attribute sets are very well adapted to the classification and the analysis of RDF data. Experiments show that the provided implementation of pattern structures for structured attribute sets is quite efficient.

After defining the algorithm we discuss its application to RDF graphs. These RDF data are distributed over independent resources which need to be centralized and explored for domain specific applications. We propose a new approach based on interactive data exploration paradigm using Pattern Structures to provide exploration and navigation over Linked Data through concept lattices. It takes RDF triples and RDF Schema based on user requirements and provides one navigation space resulting from several RDF resources. This navigation space allows user to navigate and search only the part of data that is interesting for her.

5 Road-map of the Thesis

This thesis is structured as follows:

Chapter 1 introduces the fundamentals of Formal Concept Analysis and its variants i.e., Pattern Structures and Relational Concept Analysis. It discusses basic concepts behind web clustering engines and then FCA-based web clustering engines are detailed. All the studies are divided into two parts i.e., the WCEs with and without user interaction. It also discusses the work that has already been done for adding background knowledge to concept lattices. Finally, some details on the visualization techniques developed for concept lattices are discussed.

Chapter 2 discusses the fundamentals of semantic web and the techniques applied to "Web of Documents" to turn it into "Web of Data". It also details the studies following the overall

architecture of web clustering engines for clustering the answers obtained by querying semantic web data. Finally, it gives a comparison between the functionalities of the state-of-the-art tools discussed in Chapter 1 and 2.

Chapter 3 focuses on how the overall architecture of web clustering engines can be adapted to Semantic Web data for defining views over RDF graphs with the help of Formal Concept Analysis. The proposed approach is called *Lattice-Based View Access*. After this view is obtained by applying proposed transformations the concept lattice can be navigated and accessed to find answers to user questions. This approach can be applied to any domain and can be applied to large data sets. However, the maximum limit defined by several SPARQL endpoints is 100,000 answers, so the maximum number of triples on which our approach is tested is this maximum limit. Moreover, we also developed a tool called RV-Xplorer (Rdf View eXplorer) which allows the user to perform several navigational operations. During this exploration several incompletions were revealed which are targeted in the next chapter with the help of association rule mining.

Chapter 4 discusses the notion of data quality assessment. It discusses the problem of incompletion found in the RDF datasets while exploring the concept lattice obtained through LBVA. Currently we compute implications from the formal context and compute their confidence in the opposite direction. We use this measure to rank the implications for evaluation purposes. Now, the question arises that given a background knowledge, how can one classify these RDF triples obtained by SPARQL query. We discuss these issues in the next chapter.

Chapter 5 revisits the pattern structures for structured attribute sets. It focuses on the already existing works Carpineto and Romano (1996b, 2004a); Ganter and Kuznetsov (2001a) which allows to embed the background knowledge to data while building a concept lattice using Least Common Ancestor. The original proposition does not give many details on the whole framework, and especially on the possible ways of implementing the similarity operation. In this chapter we detail the algorithm for computing Least Common Ancestor and provide several experimentations.

Chapter 6 applies the pattern structures for structured attribute sets to RDF data. It discusses how RDF triples can be clustered using background knowledge by using the similarity measure defined in the previous chapter. It allows the user to be directly involved during this process, where user can define her prior beliefs for selection of task specific data sets. Afterwards, the pattern structures is applied to obtain a *navigation space* which allows simultaneous navigation over RDF triples and RDF Schema and interactive exploration.

This document is concluded with a summary of contributions and some perspectives of our work. After concluding the thesis, we present some of the studies in the appendix of the thesis.

Appendix A. In this appendix we study the correspondence between FCA and $\mathcal{ELI}$ Ontologies. This work was done during the Post-Doc of Melisachew Wudage Chekol in LORIA. This is important to mention because it is not very straight forward to directly connect FCA with semantic web. We present this work in Appendix because this work is still on it's course of completion.

Appendix B. During this study we show how FCA can be applied to complex transcriptomic data. It then ranks the concepts with respect to their stability measure. Finally, these concepts are analyzed by the experts to obtain useful knowledge.

Chapter 1

Formal Concept Analysis and Web Clustering Engines

Contents

This chapter discusses the basic notions used in the rest of this thesis and the state-of-the-art. It first discusses the fundamentals of Formal Concept Analysis and its variants such as Pattern Structures and Relational Concept Analysis in section 1.1, 1.2, 1.3. Afterwards, it discusses the basic architecture of Web Clustering Engines and gives a brief overview of the overall architecture of web clustering engines. It also discusses the modification of the overall architecture of the web clustering engines to provide user feedback (see section 1.4). Then, the state-of-the art systems which use Formal Concept Analysis as the web clustering engine are briefly introduced

in section 1.5. One of the basic component of web clustering engines is visualization. Section 1.6 details some of the existing tools which allow the user to visually navigate the concept lattice.

1.1 Formal Concept Analysis

Formal Concept Analysis (FCA) Ganter and Wille (1999) is a mathematical framework which represents objects and their attributes in the form of formal contexts, represented as binary tables. From this formal context, a concept lattice is built. To date, FCA has been applied to various fields of data mining more precisely classification and data analysis, information retrieval and knowledge discovery Carpineto and Romano (2004a). This section gives a detailed introduction to FCA and is based on the definitions from Ganter and Wille (1999).

Definition 1 (Formal Context)**.** *A formal context $\mathcal{K} := (G, M, I)$, consists of two sets, a set of* **objects** *G, a set of* **attributes** *M and a* **binary relation** *I between G and M. The binary relation $(g, m) \in I$ or gIm is interpreted as "object g is in a relation I with an attribute m".*

Example 1. *Table 1.1 presents a formal context related to museums which display the work of some artists. A formal context is represented as a binary table. The museums are the set of objects while the artists are the set of attributes in the context. The fact that a museum displays the work of an artist defines a relationship I and is represented as a cross in the binary context. According to the first row, "$museum_1$ displays the paintings of Raphael, Da Vinci and Caravaggio".*

From the binary context formal concepts are obtained keeping the classes of objects sharing some attributes. These concepts are computed by applying derivation operators. Given $A \subseteq G$ and $B \subseteq M$, two derivation operators, both denoted by $'$, formalize the sharing of attributes for objects. Dually, the sharing of objects for attributes:

$$A' = \{m \in M \mid gIm \ for\ all\ g \in A\} \tag{1.1}$$

$$B' = \{g \in G \mid gIm \ for\ all\ m \in B\} \tag{1.2}$$

The two derivation operators $'$ form a *Galois connection* between the powersets $\wp(G)$ and $\wp(M)$. The composition of these two operators is a closure operator. Maximal sets of objects related to maximal set of attributes correspond to closed sets of the composition of both operators $'$ (denoted by $''$).

Definition 2 (Formal Concept)**.** *A formal concept of the context $\mathcal{K} := (G, M, I)$ is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is the extent and B is the intent of the concept (A, B). $\mathcal{B}(G, M, I)$ denotes the set of all concepts of the context (G, M, I).*

Example 2. *Consider the binary context in Table 1.1, the pair* ($\{museum_3, museum_4\}$, {Goya, Caravaggio}) *is a formal concept because* $\{museum_3, museum_4\}' =$ {Goya, Caravaggio} *and* {Goya, Caravaggio}$' = \{museum_3, museum_4\}$*, which means that the set of artists whose work is displayed in both the museums $museum_3$ and $museum_4$ are* {Goya, Caravaggio}*. It is represented as a maximal rectangle shown in gray background in Table 1.1.*

Definition 3. *Let $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$ be two concepts, then C_1 is a* **subconcept** *of C_2 and C_2 is a* **superconcept** *of C_1, denoted by $C_1 \leqslant C_2$, iff $A_1 \subseteq A_2$ and $B_2 \subseteq B_1$. This relation is represented as $C_1 \leqslant C_2$ where $\leqslant$ is called the partial order of the concept. The set of all concepts $\mathcal{B}(G, M, I)$ ordered in this way is called a* **concept lattice**.

	Artist				
Museum	Raphael	Da Vinci	Picasso	Caravaggio	Goya
$museum_1$	×	×		×	
$museum_2$			×		
$museum_3$	×			×	×
$museum_4$		×		×	×

Table 1.1: Formal Context $\mathcal{K}$.

Example 3. *An example of $\leqslant$-relation between two concepts with respect to Table 1.1 will be: ($\{museum_3, museum_4\}$,{Goya, Caravaggio}) $\leqslant$ ($\{museum_1, museum_3, museum_4\}$, {Caravaggio}). Here, $\{museum_3, museum_4\} \subseteq \{museum_1, museum_3, museum_4\}$ and {Caravaggio} $\subseteq$ {Goya, Caravaggio}. Figure 1.1 shows a complete lattice for Table 1.1 with reduced labeling. Reduced labeling takes into account the inheritance relation between the intents of super and sub concept i.e., if the super concept contains an attribute in the intent of the concept then all its sub-concepts will contain this attribute.*

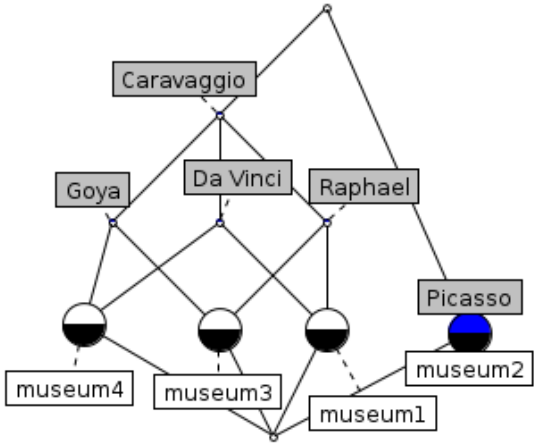


Figure 1.1: Concept Lattice for Museums for the context Table 1.1.

Definition 4. *For an object $g \in G$ we write g' instead of $\{g\}'$ for the* **object intent** *$\{m \in M|gIm\}$ of the object g. Correspondingly, $m' := \{g \in G|gIm\}$ is called the attribute extent of the attribute m. The object concept can then be written as (g'', g') and the attribute concept (m', m'').*

Example 4. *According to the running example in Table 1.1, the object concept for $museum_1$ is given as ($\{museum_1\}$,{Raphael, Da Vinci, Caravaggio}) because $\{museum_1\}'' = museum_1$ and $\{museum_1\}'$ = {Raphael, Da Vinci, Caravaggio}. Similarly, the attribute concept for* Raphael *is given as* {Raphael, Caravaggio},$\{museum_1, museum_3\}$ *because {Raphael}'' = {Raphael, Caravaggio} and {Raphael}'= $\{museum_1, museum_3\}$.*

Several algorithms have been proposed such as CloseByOne Krajca et al. (2010), Next Closure Borchmann (2012) and AddIntent van der Merwe et al. (2004) to build a concept lattice

Museum	*Building*	*Paintings*
$museum_1$	good	excellent
$museum_2$	poor	satisfactory
$museum_3$	excellent	excellent
$museum_4$	excellent	good

Table 1.2: Many-Valued Context (Museum Ratings).

which also focus on efficiency of building the lattices for large number of objects. In van der Merwe et al. (2004) the authors define a fast algorithm *AddIntent* which is an incremental algorithm for building concept lattices. This algorithm takes as an input the lattice produced by the first i objects the contexts and inserts the next object g to generate a new lattice. The construction of the lattice consists of four kinds concepts, modified concepts, generator concepts, new concepts and old concepts. A concept is a new concept if it's intent does not already exist in the lattice. A concept is modified if a concept already exists having attributes of an object g and g has to be added to its extent. For an existing concept (A, B) and $B \cap g' = D \neq B$ for some concept (C, D), if (C, D) does not already exist in the lattice then (A, B) is called the generator of the (C, D) otherwise (A,B) is an old concept.

1.1.1 Many-Valued Context

In some cases, a many-valued context is more directly available or can be deduced instead of a binary context. A many-valued context is defined as follows:

Definition 5. *A many-valued context is denoted by (G, M, W, I) and consists of G the set of objects, M the set of (many-valued) attributes, W the set of attribute values and a ternary relation I between G, M and W i.e., $I \subseteq G \times M \times W$ for which it holds that $(g, m, w) \in I$ meaning that* "the attribute m has the value w for the object g".

If W has n values then G, M, W, I is called the n-valued context. Like **one-valued** contexts, many-valued context can be represented by tables, the rows of which are labeled by the objects and the columns labeled by the attributes. The entry in row g and column m represents the attribute value $m(g)$. If the attribute m does not have a value for the object g, there will be no entry.

Example 5. *Let us consider the ratings of each of the museums for the state/architecture of the building and the quality of the collection of paintings they display. The values that each of the attribute can take is poor, satisfactory, good and excellent. Then the many-valued context for the museum ratings are shown in Table 1.2. The first row depicts that the rating of the $museum_1$ for the paintings it displays is excellent.*

Conceptual Scaling. Now the question arises: how can we built a concept lattice from a many-valued context? For doing so, a many-valued context is transformed into one-valued context. The concepts obtained from one-valued context are then interpreted as many-valued context. This interpretation process is called as **conceptual scaling**.

Definition 6 (Scale)**.** *A scale S_m of an attribute m of a many-valued context is a one-valued context (G_m, M_m, I_m) with $m(G) = S_m$ and $m(G) \subseteq G_m$ for $m \in M$ and then the new set of attributes is $M_s = \bigcup_{m \in M} S_m$. The objects of a scale are called* **scale values**, *the attributes are called* **scale attributes**.

S_B	poor	good	excellent
poor	×		
good		×	
excellent		×	×

Table 1.3: S_B.

S_P	satisfactory	good	excellent
satisfactory	×		
good	×	×	
excellent	×	×	×

Table 1.4: S_P.

	?keywords				?paintings		
Museum	poor	satisfactory	good	excellent	satisfactory	good	excellent
$museum_1$		×	×		×	×	×
$museum_2$	×				×		
$museum_3$		×	×	×	×	×	×
$museum_4$		×	×	×	×	×	

Table 1.5: Scaled Context (Museum Ratings).

During plain scaling the object set G remains unchanged, every many-valued attribute m is replaced by the scale attributes of scale S_m.

Definition 7. *If (G, M, W, I) is a many-valued context and S_m, $m \in M$ are scale contexts, then the derived context with respect to plain scaling is the context (G, N, J) with*

$$N := \bigcup_{m \in M} M_m$$

and

$$gJ(m, n) \Longleftrightarrow m(g) = w \text{ and } wI_m n$$

Example 6. *Scales S_B and S_P for the attributes Building and Paintings respectively in Table 1.2 are given in Table 1.3 and Table 1.4 respectively. In this example, if the rating of the museum is excellent then it is also good. The final scaled context is shown in Table 1.5.*

1.1.2 Iceberg Concept Lattice

Concept lattices sometimes contain a huge number of concepts. In order to restrict the number of concepts for facilitating interpretation and only obtaining frequent concepts, iceberg concept lattices were introduced in Stumme et al. (2001). Iceberg concept lattices contain only the top most part of the lattice. For a given concept (A,B), the support of B is the cardinality of A denoted by $|A|$. Relative support is given by $|A|/|G|$, and belongs to the interval $[0, 1]$ where $|G|$ is the total number of objects in the context $\mathcal{K}$.

Definition 8. *Let $B \subseteq M$ and let minimum support denoted by* minsupp $\in [0, 1]$. *The support of the attribute set (also called itemset) B in $\mathcal{K}$ is $supp(B) = |B'|/|G|$. An itemset B is said to be frequent if $supp(B) \geqslant minsupp$.*

A concept is called a frequent concept if its intent is frequent. The set of all frequent concepts of $\mathcal{K}$, for a given threshold, is called an iceberg concept lattice of $\mathcal{K}$ Stumme et al. (2001). Because the support function is monotonously decreasing i.e., $B_1 \subseteq B_2 \Rightarrow supp(B_1) \geqslant minsupp$, the iceberg concept lattice is an order filter of the concept lattice and thus a join semi-lattice. Consider a complete concept lattice in Figure 1.1, let us consider that the $minsupp = 2$ for computing the iceberg concept lattice then the concepts having the extent size less than 2 will not be generated. The resulting concept lattice is shown in Figure 1.2.

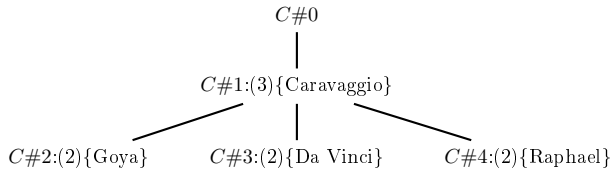


Figure 1.2: Iceberg Concept lattice for Museum for $minsupp \geqslant 2$.

1.1.3 Stability Index

An alternative way of filtering huge concept lattices is to rely on some indices. The stability measure of a formal concept was first introduced in Kuznetsov (2007). It measures how much an intent in the concept depends on the objects contained in the extent of the concept and vice versa. In this section we discuss two kinds of stability: 1) Intensional stability and 2) Extensional stability.

Intensional Stability. The intentional stability of a concept refers to the fact that if a random set of object is removed from the extent of a concept the intent of the concept would change. More formally, the intensional stability of a concept (A, B) is defined as follows:

$$\sigma_i(A, B) = \frac{|\{C \subseteq A | C' = B\}|}{2^{|A|}}$$

The underlying intuition can be explained as follows: $|A|$ represent the set of objects in the extent each concept (A, B) and $2^{|A|}$ represents the total number of subsets of such objects.

Extensional Stability. Dually, the extensional stability measures the change in the extent of a concept if a random set of attributes is removed from the intent of the concept. More formally, the extensional stability indexes for a concept (A, B) are defined as follows:

$$\sigma_e(A, B) = \frac{|\{D \subseteq B | D' = A\}|}{2^{|B|}}$$

The above equation can be interpreted in the same way as described for intentional stability by replacing object with attribute and attribute with object.

Example 7. *Figure 1.3 shows the concept lattice labeled with the corresponding intensional and extensional stability. Therefore, only the concepts with stability index* $\geqslant 0.5$ *can be selected. In Figure 1.3, the red color shows the concepts with stability index* < 0.5*. The lattice obtained after filtering w.r.t. stability does not conserve the monotonicity property.*

1.1.4 Association Rule Mining

FCA also allows knowledge discovery using association rules. Association rule mining is one of the most researched topics in data mining for extracting interesting frequent patterns and association among frequent patterns in the transactional database Agrawal et al. (1993). In order to mine association rules, first frequent itemsets are extracted. The extraction of frequent itemsets consists of extracting from formal binary contexts sets of properties occurring with a support, i.e., the number of individuals sharing the properties, greater than a given threshold. From these frequent itemsets, it is then possible to generate association rules of the form $A \rightarrow B$

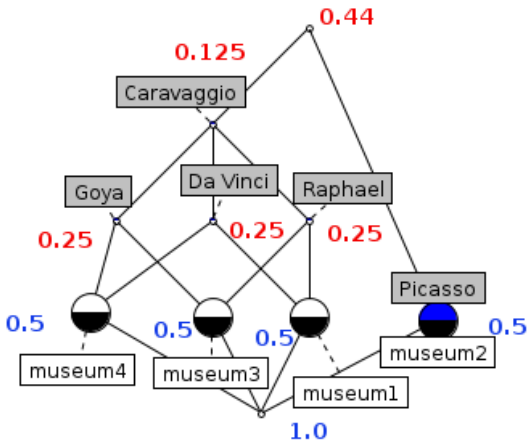


Figure 1.3: Concept Lattice for Museums for the context Table 1.1 where each concept is labeled with intensional stability.

relating A with B, where $A, B \subseteq M$, that can be interpreted as follows: the individuals including A also include B with a certain support and confidence.

Given a set of objects G and a set of properties M, an item corresponds to a property of an object and an itemset or a pattern, to a set of items: an object is said to include an item. The number of items in an itemset determines the length of an itemset. The image of an itemset corresponds to the set of objects including the item.

Definition 9 (Support of Rule). *Let $\mathcal{K} = (G, M, I)$ be a formal context and $A \rightarrow B$ be an association rule, where $A, B \subseteq M$, then the support of this association rule is given as follows:*

$$support(A \rightarrow B) = \frac{supp(A \cup B, \mathcal{K})}{|G|}$$

Definition 10 (Confidence of Rule). *Let $\mathcal{K} = (G, M, I)$ be a formal context and $A \rightarrow B$ be an association rule, where $A, B \subseteq M$, then the confidence is given as follows:*

$$conf(A \rightarrow B) = \frac{supp(A \cup B, \mathcal{K})}{supp(A, \mathcal{K})}$$

A rule is called valid if its confidence is greater than a confidence threshold σ_c and its support is greater than the frequency threshold σ_s. For example, for formal context in Table 1.1 if $\sigma_s = 2/5$ and $\sigma_c = 1/3$ then the rule $Raphael \rightarrow Caravaggio$ is frequent because the $supp(Raphael \rightarrow Caravaggio) = 2/5$ and $conf(Raphael \rightarrow Caravaggio) = 2/2 = 1$.

Implications. An implication over the attribute set M in a formal context is of the form $B_1 \Rightarrow B_2$, where $B_1, B_2 \subseteq M$. The implication holds iff every object in the context with an attribute in B_1 also has all the attributes in B_2. Meaning that an association rule is called an implication iff its confidence is 100%, i.e., $supp(A \cup B) = supp(A)$. Hence, when $(A_1, B_1) \leqslant (A_2, B_2)$ in the lattice, we have that $B_1 \Rightarrow B_2$.

Duquenne-Guigues ($\mathcal{DG}$) basis for implications Guigues and Duquenne (1986), also called as canonical bassis, is the minimal set of implications equivalent to the set of all valid implications for a formal context $\mathcal{K} = (G, M, I)$. Actually, the $\mathcal{DG}$-basis contains all information lying in the concept lattice. For example, the Duquenne Guigues Basis for the formal context in Table 1.1 contains three rules: 1) Raphael $\Rightarrow$ Caravaggio, 2) Da Vinci $\Rightarrow$ Caravaggio and 3) Goya $\Rightarrow$ Caravaggio. These rules can be interpreted as: the museums which display the paintings of Raphael, Leonardo Da Vinci or Goya also display the painting of Caravaggio.

1.2 Pattern Structures

Formal Concept Analysis Ganter and Wille (1999) can process only binary context, more complex data such as graphs can not be directly processed by FCA. Moreover, the concept lattice obtained by a binary context mixes between several types of attributes. Pattern structures Ganter and Kuznetsov (2001a), an extension of FCA, allows direct processing of such kind of context. The pattern structures were introduced in Ganter and Kuznetsov (2001a).

A pattern structure is a triple $(G, (D, \sqcap), \delta)$, where G is the set of objects, $(D, \sqcap)$ is a meet-semilattice of descriptions D and $\delta : G \rightarrow D$ maps an object 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 structures then the derivation operators can be defined as:

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

Now the pattern concept can be defined as follows:

Definition 11 (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.*

Let us consider the context in Table 1.6, such kind of context can not be directly processed by FCA without going through the nominal/inter-ordinal scaling step which converts such a context into a binary context. The first record shows that the object g_1 has the numeric value 5. We will use interval pattern structures introduced to especially deal with numerical data.

Interval Pattern Structures. Let us explain pattern structures with the help of Interval Pattern Structures, which was first introduced in Kaytoue et al. (2011a) for dealing with numerical data instead of binary data. Consider two descriptions $\delta(g_1) = \langle [l_i^1, r_i^1] \rangle$ and $\delta(g_2) = \langle [l_i^2, r_i^2] \rangle$, with $i \in [1..n]$ where n is the number of intervals used for the description of entities. The similarity operation $\sqcap$ and the associated subsumption relation $\sqsubseteq$ between descriptions are defined as the convex hull of two descriptions as follows:

$$\delta(g_1) \sqcap \delta(g_2) = \langle [min(l_i^1, l_i^2), max(r_i^1, r_i^2)] \rangle$$
$$\delta(g_1) \sqsubseteq \delta(g_2) \iff \delta(g_1) \sqcap \delta(g_2) = \delta(g_1)$$
$$\delta(g_2) \sqcap \delta(g_4) = \langle [4, 6], [8, 9], [4, 8] \rangle$$
$$(\delta(g_2) \sqcap \delta(g_4)) \sqsubseteq \delta(g_3)$$

	m_1	m_2	m_3
g_1	5	7	6
g_2	6	8	4
g_3	4	8	5
g_4	4	9	8

Table 1.6: Formal Context with Numerals.

Following the definition of a pattern concept (see Definition 11), $\{g_2, g_3, g_4\}^{\square} = \langle [4, 6], [8, 9], [4, 8] \rangle$ is a pattern concept and is called as *ip-concept* (interval pattern concept). The obtained concept lattice is called as *pattern concept lattice*. A complete pattern concept lattice is shown in Figure 1.4. Pattern structures have also been introduced to deal with graphical data Ganter and Kuznetsov (2001a), sequential data Buzmakov et al. (2013) etc.

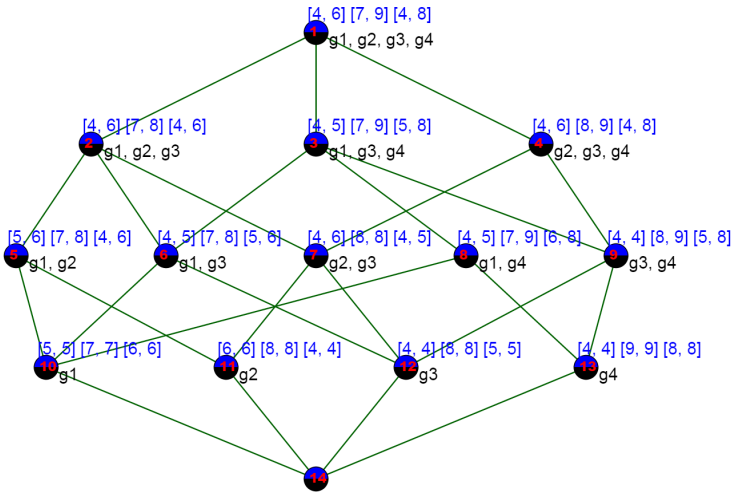


Figure 1.4: Pattern Concept Lattice for Interval Pattern Structures.

1.3 Relational Concept Analysis

Beside class hierarchies provided by concept lattices, an integrated class model must include relations available between classes, and possibly, abstractions of these relations. The abstraction of relations requires an encoding of roles into a formal context together with their attributes. The Relational Concept Analysis framework addresses these concerns, allowing FCA to take effectively and efficiently into account relational data. Relational Concept Analysis (RCA) Hacene et al. (2007), Rouane-Hacene et al. (2013) is an extension of FCA which is close to Entity-Relationship model for relational databases. The datasets provided as an input for RCA are the same as FCA i.e., formal contexts containing object-attribute relations. In addition to formal contexts it contains "relational contexts" consisting of relations between the object sets of two formal contexts. This relational context family is defined as follows:

Definition 12 (Relational Context Family (RCF)). *An RCF is a pair* $(\mathbf{K}, \mathbf{R})$ *where:*

- $\mathbf{K} = \{\mathcal{K}_i\}_{i=1,\ldots,n}$ *is a set of contexts* $\mathcal{K}_i = (G_i, M_i, I_i)$,

Museum/Location	Madrid	Paris	Montpellier	Spain	France
$museum_1$	×			×	
$museum_2$	×			×	
$museum_3$		×			×
$museum_4$			×		×

Table 1.7: Formal Context $\mathcal{K}_{museum}$.

Museum	Raphael	Da Vinci	Picasso	Caravaggio	Goya
$museum_1$	×	×		×	
$museum_2$			×		
$museum_3$	×			×	×
$museum_4$		×		×	×

Table 1.8: Relational Context $r_{displays}$.

Artists/Movements	High Renaissance	Cubism	Baroque	Romanticism
Raphael	×			
Da Vinci	×			
Picasso		×		
Caravaggio			×	
Goya				×

Table 1.9: Formal Context $\mathcal{K}_{artist}$.

- $\mathbf{R} = \{r_k\}_{k=1,...,m}$ *is a set of relations* $r_k \subseteq G_i \times G_j$ *for some* $i, j = 1, ..., n$

Here it can be noticed that all the object sets $G_i (i \in \{1, \ldots, n\})$ are pair-wise disjoint. Moreover, G_i (domain of r_k) and G_j (range of r_k) are the object sets of the context $\mathcal{K}_i$ and $\mathcal{K}_j$ respectively.

Example 8. *Let us consider the example of museum and extend it with relations. Let museums be the set of objects* G_1*, the location of the museum be the set of attributes* M_1 *and located_in be the binary relation* I_1*, then the first context of the relational context family* $\mathcal{K}_1 := (G_1, M_1, I_1)$ *is shown in Table 1.7. Now let the second set of objects be the artists* G_2*, let the attributes* M_1 *of the artists be the movements in which they participated, let participated_in be the binary relation* I_2 *then the context* $\mathcal{K}_2 := (G_2, M_2, I_2)$ *is shown in Table 1.9. Finally, there is a relation between museums and artists i.e., museums display the work of artists. We use this as relation in the current scenario because now artists have their own sets of properties that need to be addressed. The associated relational context is shown in Table 1.8.*

Now, the question is how to incorporate relations when constructing concept descriptions. For doing so, the Description Logic (DL) Baader et al. (2003) formalism is used. DL offers a collection of constructors to express relational information. In description logic, two types of constructs are used at schema level, *concepts* and *roles*. By properly restricting the roles of a given concept, one can easily define a sub-concept. Typical constructs for role restriction include existential quantification ($\exists R.C$), universal quantification ($\forall R.C$), strict universal quantification ($\forall \exists R.C$), cardinality restriction ($\geqslant nR$), qualified cardinality restriction ($\geqslant nR.C$), etc. Here R stands for a role, i.e., the equivalent of a relation from an RCF, whereas C is a conceptual expression that might be either a name or a formula involving logical connectors on sub-formula(s). From a mathematical point of view, the scaling of K_i along $r \in rel(K_i)$ ($rel(K_i)$ is the set of relations where the domain of each relation is the set of objects in K_i) with $ran(r) = G_j$ and with respect to a lattice L_j extends M_i by adding a set of new attributes and completes I_i accordingly.

Definition 13 (Existential Scaling Operator)**.** *Given* $K = (G, M, I)$ *and* $r \in rel(K)$*, let* j *be*

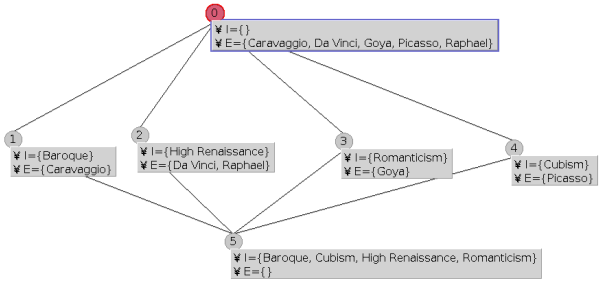


Figure 1.5: Concept Lattice for Artist

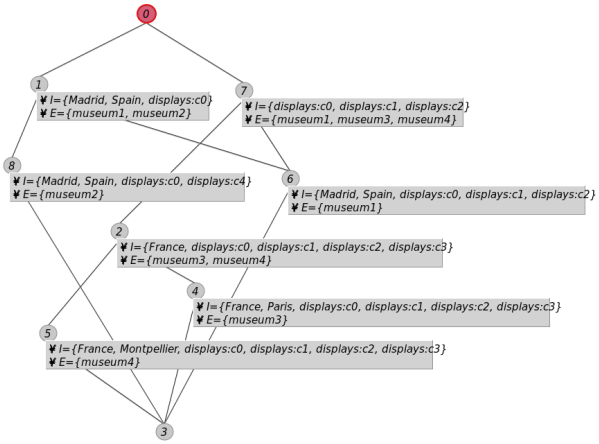


Figure 1.6: Concept Lattice for Museum with relations to Concept Lattice in Figure 1.5

such that $ran(r) = G_j$ *where* $K_j = (G_j, M_j, I_j)$. *Let also* L_j *be the lattice of* K_j . *The existential scaling operator* $S(r, \exists)$, L_j *maps* K *into the derived context* $K_i^+ = (G_i^+, M_i^+, I_i^+)$, *where:*

- $G_i^+ = G_i$
- $M_i^+ = \{\exists r : c | c \in \mathcal{L}_j\}$, *where each* $\exists r : c$ *is a relational attribute*
- $I_i^+ = \{(g, \exists r : c) | g \in G, c \in L_j, r(g) \cap Ext(c) = \varnothing\}$

Similarly, we define the universal scaling operator with respect to r and L_j , denoted by $S(r, \forall)$, L_j . The difference is that instead of non-empty intersection, the object image $r(o)$ must be completely included in the extent of c in order for o to get the relational attribute $\forall r : c$. Finally, $\forall \exists r : c$ corresponds to the DL expression $\forall R.C \sqcap \exists R.s$.

Example 9. *The final concept lattices using the existential quantification are shown in Figure 1.6 and 1.5. For example, take* $C\#6$ *in Figure 1.6, it says that* $museum_2$ *is located in Madrid, Spain and it displays the work of Picasso whose movement was Cubism* $C\#2$ *in the lattice for artists (see Figure 1.5).*

1.4 Interactivity over Web Clustering Engines

1.4.1 Exploratory Data Mining

This section is based on van Leeuwen (2014). One of the fundamental challenges faced nowadays is the explosion of generated data. It is not feasible to manually sift through this large amount

of data mainly because given a particular data there is no way to know what a domain expert is looking for. KDD process focuses on extracting knowledge from data and provide interpretation support for the user. Exploratory data analysis (EDA) was first introduced by John Tukey in 1970's Tukey (1977). The field of exploratory data mining aims at providing an insight into the data to the domain expert. Exploratory data mining aims at explaining the data by finding patterns and models while visual analytics takes advantage of visualization and human-computer interaction to allow the user to understand the data. In order to find patterns in the data, exploratory data mining uses pattern mining. A pattern is defined as a knowledge from a dataset while pattern mining is a method which extracts patterns from a data set. First problem encountered while using pattern mining algorithm is that a huge number of patterns are found which may contain redundancy. The second challenge is that most of the algorithms proposed for pattern mining compute the interestingness of a pattern from data called as *objective interestingness*, hence completely ignoring the background knowledge of the domain expert.

In van Leeuwen (2014), the authors provide an overall view of the pattern mining algorithms which allow subjective interestingness with the help of user interaction instead of only taking into account the object interestingness. Subjective interestingness is learnt from feedback provided by the user for the computed set of patterns. Initially, the system mines patterns and these patterns are then shown to the user, she provides feedback and then the system learns the user's preferences. Based on these newly learnt preferences a new set of patterns is mined. Usually, standard machine learning algorithms are used to learn these preferences. Bie (2011) discusses subjective interestingness in detail. The author proposes a mathematical framework for formalizing subjective interestingness.

1.4.2 Web Clustering Engines

This section discusses the basic paradigm of Web Clustering Engines and then details some of the studies where Formal Concept Analysis is used for clustering web search results. When user submits a query to a search engine, she gets ranked list of documents with snippets (i.e., the partial contents of the web page). This ranked list is still insufficient since the number of answers obtained for a query can be thousands in number. Most of the users only sift through the top results. Moreover, in case of ambiguity the ranked list may not represent the expected answers i.e., user has to traverse through irrelevant links to find the relevant ones. There are several ways to target this problem. One way is to cluster the whole Web, manually or automatically and show the user those categories of results which best match their query. These categories may or may not cover all the pages present on the web. Another way is to group the results retrieved and show the user all the categories of the results. Then, let the user choose the category of documents she wants to further explore. Later is the case of Web clustering engines Carpineto et al. (2009), it automatically group similar documents to facilitate the presentation of these results in more compact form and allow browsing through these results. Major search engines take into account this problem and they interweave the documents relevant to different topics knows as *implicit clustering*. In case of queries like "Jaguar" which are ambiguous, there is no way to know for the search engine if she is looking for the animal Jaguar or the car. However, web clustering engines are based on taking the results of the search engine as an input and execute the clustering algorithm and show the clustered output to the user. One of the successful commercial application which clusters the search results is Vivisimo[2]. It performs standard web search and provides a hierarchical clustering of these search results. These search results display a taxonomy

[2] www.vivisimo.com

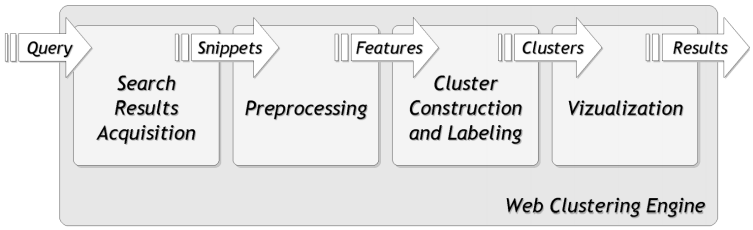


Figure 1.7: Overall Architecture of Web Clustering Engines

of results e.g., if the user query is *Jaguar* then it displays Car (The car Jaguar), the Mac operating system and the animal etc (see Figure 1.8). Such grouping allows fast browsing of the results. An overall architecture of web clustering engines is shown in Figure 1.7. This section discusses the functionality of the web clustering search engines step by step and is based on the survey on web clustering engines Carpineto et al. (2009).

Acquiring Search Results. For web clustering engines the source of acquiring search results are the standard search engines such as Yahoo, Google or Live Search. Based on user query, the search engines should return from 50 - 500 results. These results are acquired from these search engines with the help of application programming interfaces (APIs) provided by these engines and these results should contain the title of the page, a snippet from the page and the URL. For example, in case of Yahoo it can be obtained from `https://developer.yahoo.com/boss/search/`. Alternative way to obtain these results is HTML scrapping using regular expressions or other markup detection to extract the title, snippets and URLs.

Preprocessing of the pages retrieved is performed on the snippet. It applies basic natural language processing tasks such as tokenization and stemming etc. Tokenization splits the text of each search result into a sequence of units called *tokens* which include word, alphabet, symbol etc. Afterwards, Stemming is done which strips the inflectional prefixes and suffixes of the words to a common base called as a *stem*. Finally, it gives a set of documents with their features which is used by the clustering algorithms.

Constructing the clusters and *assigning labels* to these clusters is the next step. The features obtained by the previous step is given as an input to clustering algorithm. There are several types of clustering algorithms that can be used for this task. Carpineto et al. (2009) divides these clustering algorithms into three categories, i.e., data-centric algorithms, description aware algorithms and description-centric algorithms. *Data-centric algorithms* consists of conventional data clustering algorithms such as hierarchical, optimization and spectral algorithms. The problem with such algorithms is creating a comprehensive description from the text that is not prepared for this purpose. *Description-aware* algorithms are aware of the labeling problem and they ensure that the construction of cluster descriptions is feasible and yields human interpretable descriptions. It includes Suffix Tree Clustering (STC) Weiner (1973). *Description-centric algorithms* are designed specifically for clustering search results. If a cluster can not be described than this cluster is presumed to be of no value and is removed from the view entirely.

Visualization of these clustered search results is the last step. There can be several ways to visualize these clusters such as tree-folder as adopted by Vivisimo (Figure 1.8), Carrot (Fig-

ure 1.8), CREDO and SnakeT Ferragina and Gulli (2004a,b). Another way is to adopt nesting and zooming approach, such as by Grokker[3] (Figure 1.9) as circles with smaller circles within it.

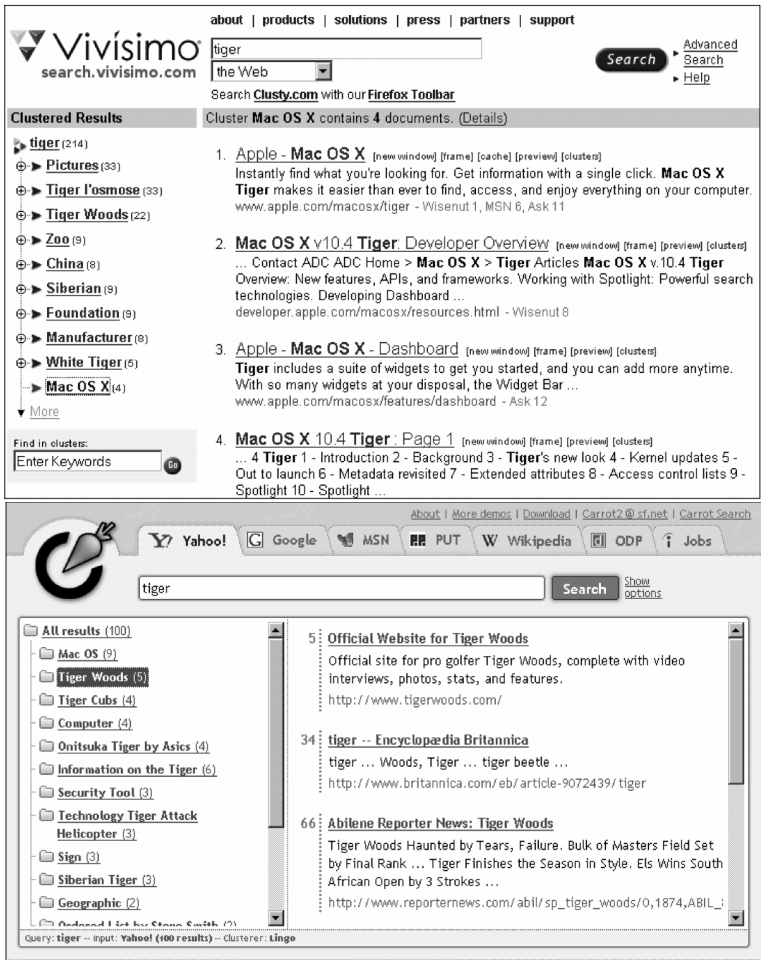


Figure 1.8: Clustered search results for the query "tiger". The image above is taken from Vivisimo and the image below is taken from Carrot2.

The idea of web clustering engine was first introduced in Scatter/Gather system Cutting et al. (1992, 1993) which uses K-means algorithm. Then it was followed by Grouper Zamir and Etzioni (1999) which implemented a phrase analysis algorithm called Suffix Tree Clustering (STC) which groups the snippets sharing some sequence of words. Afterwards, Carrot2 was released (which was actually built during the time of Vivisimo but was released later). Since the time of release of Carrot2 Stefanowski and Weiss (2003); Osinski and Weiss (2005) many improvements have been introduced i.e., several clustering algorithms have been implemented such as agglomerative techniques techniques, K-means, Suffix Tree Clustering etc.

1.4.3 Interactive Exploration over Web Search Results

Here we discuss how the subjective interestingness of the user can be learnt in case of web clustering engines. In Figure 1.10, we generalize the framework of Web Clustering Engines to allow interactive exploration i.e., to allow the user to provide feedback. During this process the

[3] `www.grokker.com`

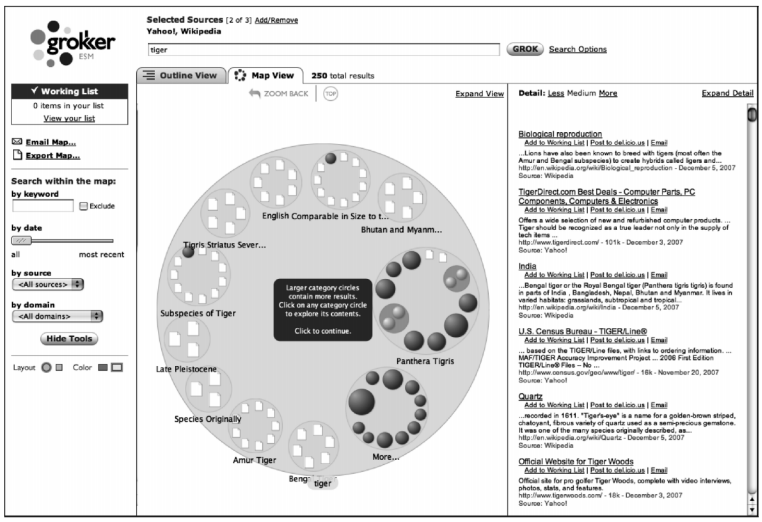


Figure 1.9: Clustered search results for the query, "tiger" for the system Grokker.

user sends a query and the meta-search engine acquires the results. The system extracts the snippets from the returned results. These snippets are then preprocessed in the same way as detailed before to obtain features to be fed to clustering algorithm. This clustering algorithm generates the clusters which are then visualized by the user. This visualization allows the user to interact with the obtained clusters and refine her search. It allows the user to mark the clusters as interesting or uninteresting by interaction. The uninteresting clusters are then either hidden from the user which in turn reduces the interaction space of the user or the clusters are re-computed. Some of the visualizations also allow the user to go back to the original query and refine the query and re-run the query. Interactive exploration is an iterative process which allows the user to refine his results and get only those results which are interesting for her.

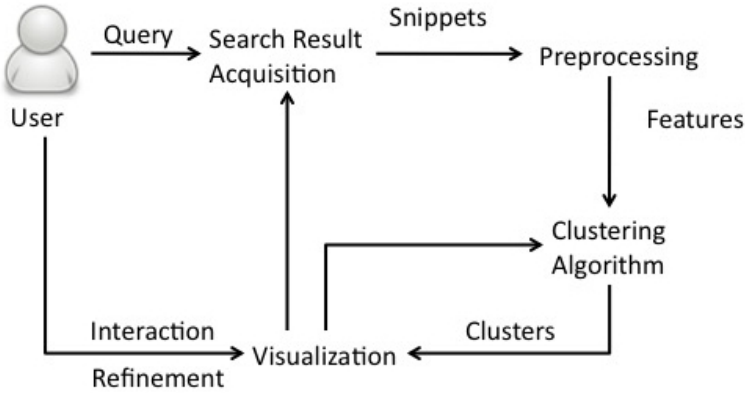


Figure 1.10: A Framework for Interactive Exploration of Web Search Results.

1.5 Formal Concept Analysis for Web Clustering Engines

Formal Concept Analysis is a conceptual clustering technique which simultaneously generates clusters and the cluster labels. Several FCA-based web clustering engines were introduced. These

studies follow the fundamental architecture of Web Clustering Engines as detailed in the previous section. Section 1.5.1 briefly details how FCA is used in the scenario of web clustering engines and user interaction is allowed in some of the FCA-based web clustering engines. Section 1.5.2 discusses the studies which take into account background knowledge in the form of a taxonomy for clustering documents.

1.5.1 Clustering Web Search Results

CREDO was one of the first FCA-based web clustering engines introduced in Carpineto and Romano (2004b). *CREDO*, it stands for Conceptual REorganization of DOcuments. It takes as an input a user query. The query is forwarded to an external Web search engines and only first 100 pages returned by the search engines are used for further processing. CREDO parses only snippet from the retrieved document to extract the set of index terms using standard natural language processing tasks. These tasks involve text segmentation, word stemming, stop wording, word weighing. Afterwards a concept lattice is designed from the document-term relation. The concept lattice of the retrieved documents may contain many irrelevant concepts resulting from spurious combinations of the document terms. CREDO takes a hybrid approach, in which the lower levels of the CREDO hierarchy are built using a larger set of terms than those used to build the top level. CREDO follows a two-step classification procedure, in which the first layer identifies the main topics and the other layers contain the subtopics of each main topic.

Finally the obtained CREDO hierarchy is visualized using conventional tree-folder display, where the top contains all the documents. On click it shows the children of the top which classify the documents based on the terms. The user can click on one concept and see its children, thus narrowing down the scope of the search. This operation can be repeated on the newly-displayed concepts to further narrow the scope of the search, or it can be performed on other top concepts to browse unexplored branches. Figure 1.11 shows a snapshot of CREDO for the query *"Jaguar"*.

CreChainDo. Nauer and Toussaint (2007) improved the system CREDO by allowing the user to provide feedback instead of just navigating the concept lattice. CreChainDo allows the user to mark the concepts which are relevant or irrelevant to the user. If the user marks a concept as irrelevant then all the sub-concepts of this concept are also marked irrelevant. After the user has marked this, a new lattice is computed by removing the irrelevant objects or attributes. This way the iterations continue until a fix-point is achieved where the concept lattice contains only the relevant documents. Figure 1.12 shows the snapshot of the system CreChainDo for the query *"Carpineto Romano"*.

JBraindead. Cigarrán et al. (2004) use free-text search engine with Formal Concept Analysis for organizing query results. When a user queries the search engine it retrieves a ranked list of results. JBraindead takes these results and decides on the set of attributes which best represent the documents. In order to do so the authors first define a *lattice distillation factor* which measures how well the lattice prevents the user from accessing irrelevant information. Secondly, it computes lattice browsing complexity which measures how many concept the user has visited to finally reach the relevant information. This measure is referred to as *(minimal browsing area)*. JBraindead2 Cigarrán et al. (2005) updates the previous version by defining three methodologies for automatically selecting the noun phrases which are used as the attributes of a document, based on which a concept lattice is built. This lattice is then evaluated using *lattice distillation factor* and *minimal browsing area* as described earlier. It also introduces a tree-folder display for its user interface.

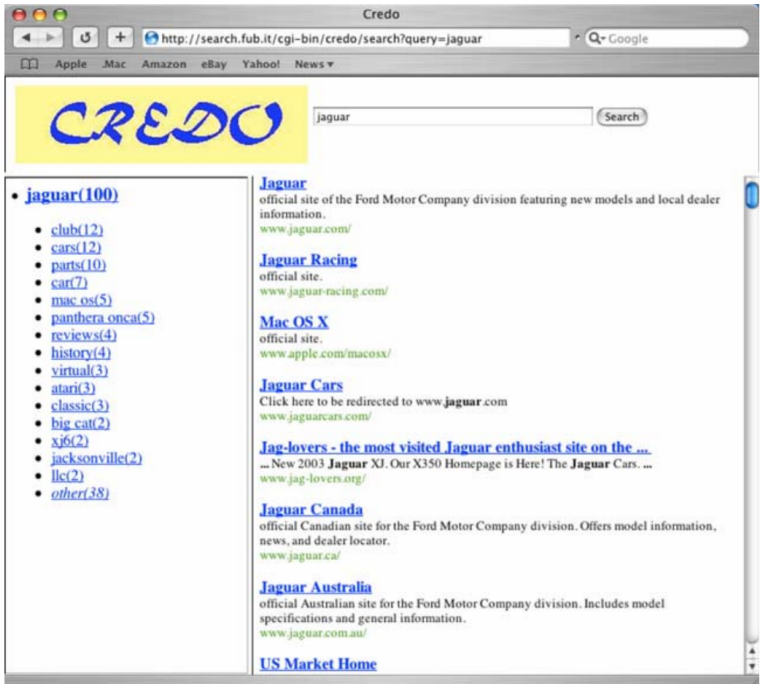


Figure 1.11: Clustered search results for the query "Jaguar". Image from CREDO. The left panel shows the cluster labels and the right panel show the ranked results belonging to the selected clusters.

Figure 1.12: Clustered search results for the query "Carpineto Romano". Image from CreChainDo. The red crosses allow the user to mark the irrelevant concepts and green crosses allow the user to mark relevant concepts.

FooCA. Koester (2005, 2006) uses title, short description and URL of the results for building a context. FooCA allows iterative exploration of the context. The user can click on any attribute to either search for this attribute or launch a new query based on this attribute which qualifies the previous query. The selection/removal of attributes allows the user to refine the search. The user is further allowed to refine the search by limiting the number of objects to be shown. It also preserves the original ranking performed by the search engine by ranking the objects and also ranking the related attributes. Finally, the lattice file can be exported for visualization in any

interface of choice such as ToscanaJ Becker et al. (2002). Figure 1.13 show the clustered search results for the query "ICDM 2006" in FooCA.

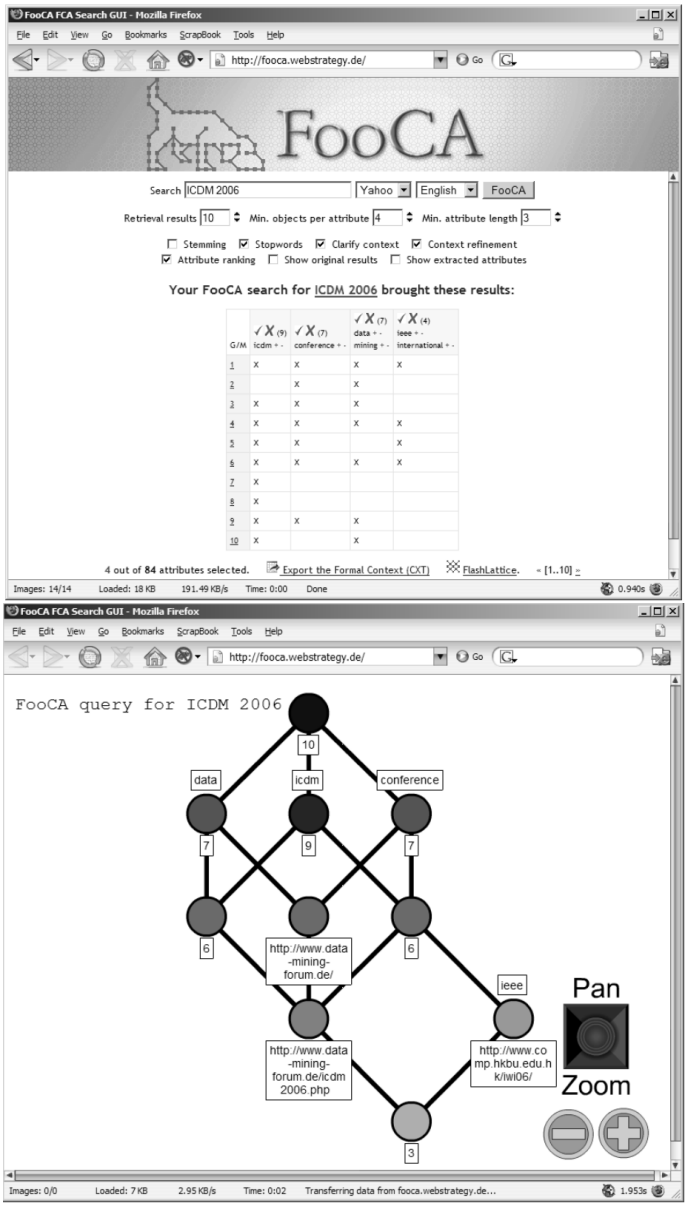


Figure 1.13: Clustered search results for the query "ICDM 2006" in FooCA. The first image allows the user to interact with the formal context. The second image allows the user to navigate through the lattice.

SearchSleuth. Dau et al. (2008) is a web clustering engine based on FCA. It employs conceptual neighborhood paradigm for displaying the clustered web results. Like other web clustering engines SearchSleuth also creates the formal context for each query. After the results are obtained from search engine, it follows the processing steps defined for the web clustering engines. In addition to providing the basic functionalities of web clustering engines, SearchSleuth allows the reduction of attributes with the support less than 5%. After the creation of formal context

a *search concept* is created by taking into account query terms as attributes. As a next step the *upper neighbors* of this formal concept are derived. Then the formal context is expanded by querying the search engine with the attributes of each *upper neighbor* and then a limited number of these results are inserted into the context.

After context expansion, the search concept is computed again as this process can invalidate this process. In the next step, the upper and lower neighbors are computed. Finally, *Sibling concepts* are computed by finding all of the lower neighbors of upper neighbors which are upper neighbors of lower neighbors. Figure 1.14 shows the results of the query *"Formal Concept Analysis"*.

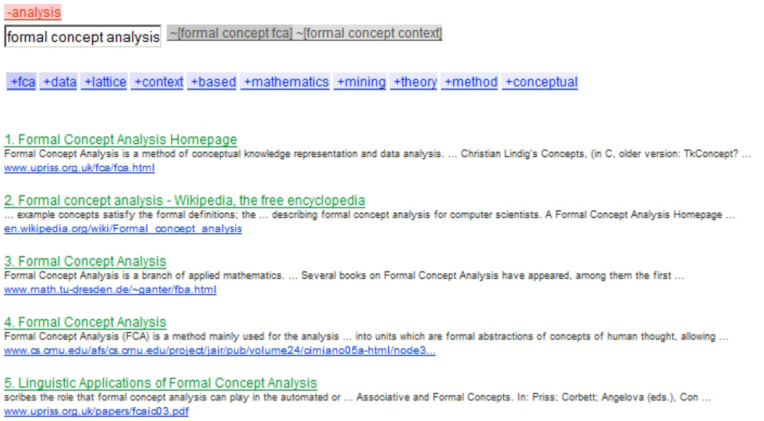


Figure 1.14: Clustered search results for the query "formal concept analysis" over SearchSleuth. It shows more general and more specific formal concepts above and below the the search box. Siblings are shown on the right.

1.5.2 Adding Background Knowledge to Concept Lattices

Several studies have been conducted which focus on how background knowledge can be added to concept lattice. This background knowledge can be present in the form of thesaurus, taxonomy or ontologies. One of the earlier studies which focus on defining this problem are Carpineto and Romano (1996a); **?**. Where the document-term relation is considered as a formal context derived by text processing. One of the simple methods for using structured descriptor would be to expand the formal context that describes the data. Meaning that the description of each document keep all the keywords that are implied by the original keyword Carpineto and Romano (2004a). Another way to add background knowledge is to modify the intersection operation defined w.r.t. to the terms (i.e., attributes in the formal context). If (X_1, Y_1) and (X_2, Y_2) are two concepts then for each pair (d_1, d_2) such that $d_1 \in Y_1$ and $d_2 \in Y_2$, the most specific keywords in the taxonomy which are general than d_1 and d_2 are selected and then only the most specific elements of the set of terms generated are retained.

In Hotho et al. (2003), the authors propose how textual documents can be clustered using background knowledge which can be in the form of ontology. They first process text and enrich their representations by background knowledge present as ontology such as Wordnet. Then, the documents are clustered using a partition algorithm. Second, the clustering partitions the large number of documents to a relatively small number of clusters, which is then analyzed by conceptual clustering. The conceptual clustering method used in this case is Formal Concept Analysis. These approaches use the same kind of document term relationship as used by web

clustering engines. However, in order to use the background knowledge it is important to know the domain of the posed query. There should be ways to search for the background knowledge automatically based on the user query. As this goal is not very feasible, these approaches are applied to pre-defined set of documents only. Further details are discussed in Chapter 5.

1.6 User Interfaces for Navigating Concept Lattices

One of the most important issue is to provide readability to the user for the clusters created by the clustering algorithm in web clustering engines. This task is achieved by visualizing the concept lattice when FCA is the chosen clustering algorithm. In the following we discuss some of the systems/tools introduced for visualizing, navigating and querying a concept lattice. We are going to discuss the properties of each system one by one and then finally give the comparison between all the tools.

1.6.1 Tree-Folder Display

CREDO Carpineto and Romano (2004b) is one of the first web clustering engines and it provides visualization of clustered web search results in the form of tree-folders. Figure 1.11 shows a screen grab of CREDO displaying the result of the query in the form of tree-folder display. The user can navigate upwards and downwards to retrieve interesting results. Other web clustering engines such as CreChainDo Nauer and Toussaint (2007) and JBrainDead Cigarrán et al. (2004) also follows the tree-folder structure. However, Search Sleuth Dau et al. (2008) follows a slightly different approach to display the results. It first computes the search concepts and then computes the upper and lower neighbors. Then, it computes the sibling concepts. *FooCA* Koester (2006) is also a web clustering engines which allows the user to refine the search by showing a context and allow the user to refine the search by limiting the number of objects and attributes (see Figure 1.13). The final lattice obtained only keeps the information that the user wants to see.

1.6.2 Systems using Hasse Diagram

This section discusses the traits of the lattice visualization tools which allow navigation, querying and information browsing by taking advantage of the Hasse Diagram of a concept lattice.

ULYSSES Carpineto and Romano (1995) is the tool for lattice visualization allowing several operations such as browsing, querying and bounding. To control the size of lattice to be viewed it generates the concept lattice one level at a time. The number of nodes to be shown is further restrained by a constant. It also contains the feature of query and bound. The query can be formulated in two ways, either the user specifies the new terms from scratch or the user modifies the current query. The querying allows the user to make jumps to regions of interest. The last feature is bounding which allows the user to change the regions of the concept lattice from which she is retrieving the answers. If c is the admissible class and c_1 is a particular class then there are four ways to impose constraints: $c \geqslant c_1$, $c \leqslant c_1$, $c\neg \geqslant c_1$ and $c\neg \leqslant c_1$. Figure 1.15 shows these modes of bounding. In Carpineto and Romano (1996b), the authors further use the similar tool for the concept lattices generated using background knowledge.

Email Manager. In Cole and Eklund (1999), the authors discuss a system which helps in analyzing the e-mails. It extracts the meta-data about an e-mail using regular expressions and stores it as inverted file index. A hierarchy is defined by set of subsumption rules defined by

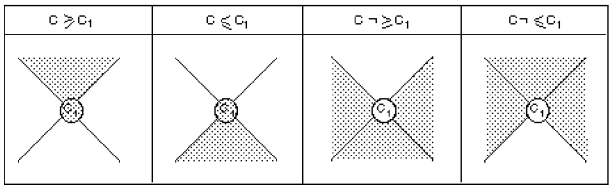


Figure 1.15: Pictorial representation of user constraints in ULYSSES.

the user. After the taxonomy of attributes is defined the user is allowed to choose a subset of attributes. To locate the needed information a text search is implemented. A conceptual scale is a subset of attributes selected by the user and is displayed in the left panel. Finally a concept lattice is built and shown to the user.

Another e-mail manager called CEM introduced in Cole and Stumme (2000). In this system, the e-mails are the objects and the attributes are the catchwords extracted from the body of the e-mail. A hierarchy of catchwords is built using Formal Concept Analysis. The tool shows the user a view of this hierarchy as a tree-widget. The program allows the user to perform 4 operations over the hierarchy of catchwords i.e., insert catchword, remove catchword, insert ordering and remove ordering. The e-mails organized in the hierarchy of catchwords are navigable through a tree display. Moreover, on selection it also shows the user a sub-lattice of the interesting part.

An improvement of CEM called as Mail Sleuth was introduced in Eklund et al. (2004). It guides the novice user to read the Hasse Diagram of a concept lattice. It introduces a user centered design and evaluation of a concept lattice. Human evaluation was performed by having one-to-one interview and the people selected for this task were from different backgrounds who did not have any knowledge of FCA. The system was scored based on its look and feel and ease-of-use. Figure 1.16 shows the display strategy of a concept lattice.

Conexp[4] and Galicia[5] are two tools for visualizing small lattices. Galicia also supports the visualization of Relational Concept Analysis. Conexp allows the user to compute implications and perform attribute exploration over a given context. However, these two functionalities are not implemented in Galicia.

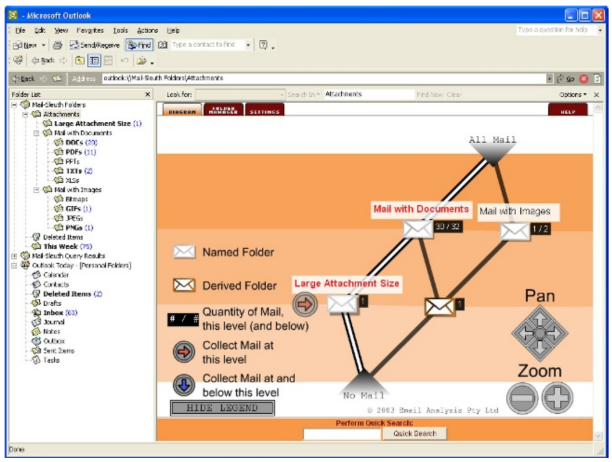


Figure 1.16: Tree as well as lattice display support by Mail Sleuth.

[4] http://conexp.sourceforge.net/
[5] http://www.iro.umontreal.ca/~galicia/

1.6.3 User Friendly Interface.

In Wray et al. (2013); Wray and Eklund (2014), the authors present an iPad application *"A Place for Art"* which allows user to explore an art collection with the help of links generated using FCA. It visualizes a collection of contemporary and Australian works from the University of Wollongong's Art Collection. It allows the user to browse through the concept lattice without actually seeing the lattice structure. It also uses implications to extract several information pieces from the the data and shows it to the user in a very intuitive way.

Image Sleuth. Ducrou et al. (2006) is a tool for browsing and searching annotated collections of images. The set of objects are the images and their annotated features are the set of attributes. The thumbnails of the images are the extent of the concept. It allows the user to restrict the set of attributes, move to upper and lower neighbours, search for similar objects and similar concepts. The concept lattice is displayed with the help of a tree display just for ensuring user readability. The Hasse diagram is only displayed for the neighbourhood of the selected concept. An extension of Image Sleuth is DVD Sleuth Ducrou (2007) which was applied to the information space built from the dynamic DVD collection in amazon.com.

Figure 1.17: Browsing images using Image Sleuth.

Camelis. In Ferré (2009), the authors present a Logical Information System called Camelis using Logical Concept Analysis introduced in Ferré and Ridoux (2000). The difference between FCA and LCA is that LCA allows the use of logical properties which are partially ordered by subsumption relation. After obtaining the concept lattices the system provides a support for navigation over the properties. First kind of navigation involves downward navigation. For example, with respect to the location property the downward navigation will be going from country to the sub-location city. Upward navigation allows the user to generalise the properties e.g., from city navigate back to country. Like web browsers, it also provides backward and forward navigation which includes keeping the history of navigation. The final form of navigation provided is the sideward navigation which is a combination of upward-downward navigation or downward-upward navigation. Moreover, Camelis also allows querying by formulas as well as examples. Querying by formulas refer to posing queries again the concept lattice through logical formulas based on the intent of the concept lattice. Query by examples refers to querying the concept lattice w.r.t. the extent of a concept in the concept lattice. It also uses a tree-folder display as a visualization tool.

1.6.4 Querying

BR-Explorer. In Messai et al. (2006), the authors introduce an algorithm for querying a concept lattice. It generates a formal concept representing the user query called as *query concept*

and then adds this concept to the concept lattice. More precisely, the user gives an attribute as a query and a concept is created where this given attribute is the intent and a dummy object is the extent of that concept. It then inserts this pair in the context which leads to two possibilities, i.e., create a new lattice from scratch or incrementally insert this concept in the existing concept lattice. In this study, the authors choose to add the concept incrementally in the concept lattice. Then it builds the query result step by step with the help of a "pivot concept". For a user query $Q(\{x\}, \{x\}')$ the pivot concept is given as: $P = (\{x\}'', \{x\}')$. The set of objects which are in $\{x\}''$ and in the intent of the super-concepts of P are assigned to the result set.

1.6.5 Categorization of Lattice Visualisation Tools

In Ferré (2014b), Sébastien Ferré distinguished these systems into three categories: 1) Query Languages, 2) Navigation Structures and 3) Interactive Views. The first category includes the systems based on some Query Language, where a query is posed by a user and an answer is returned by the system. The second category includes the systems which allow navigation over the lattice structure through links which in terms of FCA is introduced by partially ordered relation. Systems following tree-folder display or hasse diagram fall under this category. The third category includes the tools where user can interact with system to obtain the elements of interest such as Faceted Search.

According to the systems described in this chapter, BR-Explorer comes under the first category. While, CREDO and JBrainDead come under the second category which provides lattice navigation. CreChainDO, ImageSleuth, DVD-Sleuth belong to second and third category. FooCA comes under only third category and finally, ULYSSES and Camelis pertains to all three categories.

1.7 Discussion

This chapter gives a detailed introduction of the methodologies used in this thesis i.e., FCA. It summarizes the idea behind web clustering engines and explains the framework how the interactive exploration is applied to the web clustering engines. Then it discusses various web clustering engines based on FCA as a clustering algorithm (e.g., CREDO etc.) and allow interactive exploration of these results e.g., CreChainDo, FooCA etc. Finally, it discusses the functionalities of the tools developed for visualizing concept lattices as visualization is the essential part of web clustering engines.

Chapter 2

Semantic Web

Contents

This chapter moves from "Web of Documents" to "Web of Data" and discusses the fundamentals of Semantic Web (see section 2.1). It introduces the basic principles and features of Linked Data in section 2.2. Furthermore, it also details some of the systems based on semantic web data which follow the architecture of the web clustering engines (see section 2.3). Finally in section 2.4, it discusses some of the related studies which use Formal Concept Analysis for dealing with RDF data.

2.1 From Web of Documents to Semantic Web

Recently, there has been a substantial growth in data available on-line in the form of textual resources or HTML documents. These documents are accessible through web browsers. This web of documents allows the navigation from one document to another document with the help of hyperlinks. Web of documents is more easily consumed by human-agents. However with the recent efforts of Semantic Web Community the web of document is turning into web of data. Many technologies have been offered for publishing machine-processable data on web. In the following we discuss some of the schemas introduced for semantic web.

2.1.1 Schema.org

Schema.org is one such initiative by world leading search engines i.e., Bing, Yahoo, Google introduced on 2^{nd} June, 2011. It defines a common set of schema vocabularies for structured

data markup on web pages. The proposal was to mark up the contents of the website with its own metadata using `Schema.org` vocabulary with the Microdata, RDFa, or JSON-LD formats. Then the search engine spiders can identify these markups for accessing the meaning the web site. The terms defined in `Schema.org` can be mapped to RDF (expressed in RDF Schema)[6]. A list of predefined types in `Scehma.org` is also available online[7]. Some of these schema markups such as Organization and Person are used in the results associated to Google's *Knowledge Graphs*. The *Knowledge Graph* was added to Google's search engine in 2012. It is a knowledge base for enhancing the search results with semantic-search information gathered from several sources. It aims at allowing the user to obtain answer to their query without the need to traverse through all the links and manually gather the needed information. Knowledge Graph gathers data from many resources such as Freebase [8] (which going to be replaced by Wikidata in 2015) and Wikipedia. The Knowledge Graphs contain the features of answer engine such as Wolfram Alpha[9] which answers factual queries by computing the answers from curated data instead of provide a list of web pages that may or may not contain the answer required by the user. Other such efforts are Linked Data Bizer et al. (2009a) and DBpedia Bizer et al. (2009b). There have been many large-scale knowledge bases such as YAGO Suchanek et al. (2007), Freebase etc., but still these resources are far from completion. A successor of Knowledge Graph is Knowledge Vault Dong et al. (2014) created by Google in 2014. It is a Web-scale probabilistic knowledge base which extracts context from several web sources by analyzing text, tabular data, human annotations etc. It also uses the prior knowledge present on several repositories such as YAGO, DBpedia. It uses machine learning algorithms to merge all these distinct information resources. The database of Knowledge Vault contains over 1.6 billion triples.

2.1.2 Linked Data

Another emerging source of such data are published in the form of Linked Data (LD) cloud Bizer et al. (2009a). Linked Data is called Linked Open Data (LOD) when the resource is open source. It follows a predefined principals of publishing data over distributed resources, where any individual can publish data in RDF format. The whole LOD cloud contains data sets from many domains such as biomedical, government, publications etc (see Figure 2.1). LD was first introduced by Tim Berners Lee in 2006[10]. He proposed a general guidance for publishing Linked Data following which many data publishers are sharing the data. It consists of the annotations based on the documents. Instead of following links between HTML pages, It allows navigation by following RDF links. This allows the user to move through potentially large web of data just as the web of documents. The Web of Data can be crawled by following RDF links, one such crawler is LDSpider Isele et al. (2010). Working on the crawled data, search engines can provide sophisticated query capabilities, similar to those provided by conventional relational databases. Because the query results themselves are structured data, not just links to HTML pages, they can be immediately processed, thus enabling a new class of applications based on the Web of Data.

These RDF resources are interlinked with each other to form a cloud and are accessible via SPARQL queries, RDF data dumps or crawling. In some cases, queries in natural language against standard search engines can be simple to use but sometimes the queries are complex

[6] `http://schema.rdfs.org/`
[7] `https://schema.org/docs/full.html`
[8] `https://www.freebase.com/`
[9] `http://www.wolframalpha.com/`
[10] `http://www.w3.org/DesignIssues/LinkedData.html`

and may require integration of data sources. Then the standard search engines will not be able to easily answer these queries, e.g., *Currencies of all G8 countries* as it returns a resource which contains the list of G8 countries and another resource listing all the countries with their currencies. Such a complex query can be formalized as a SPARQL query over data sources present in LOD cloud through SPARQL endpoints for retrieving answers and possibly create user defined applications.

2.2 Principles of Linked Data

Linked Open Data (LOD) Bizer et al. (2009a) is the way of publishing structured data for data sharing purposes. Linked Data is built over the standard web technologies i.e., HTTP, RDF and URIs. Some of the subsections are based on Arenas et al. (2009). The principles of Linked Data Heath and Bizer (2011) are as follows:

Universal Resource Identifier (URI): Linked data uses URIs as names for things. A URI is a string of characters used to identify a name of a resource. Such identification enables interaction with representations of the resource over a network.

HTTP URIs: They define a web of information objects, where information objects carry some sort of message, and can be represented, to a greater or lesser authenticity, in bits. HTTP URIs identify the dimensions such as:

- *time*, the contents can vary with revision,
- *Context-type* in which the bits are encoded
- *Natural Language* is the language used to write a human-readable document
- *Machine language* is the language used to write a machine-processable document

Resource Description Framework (RDF) is a data model based on directed labelled graphs. Coupled with URIs, it acts as a core component of the Semantic Web. The atomic piece of knowledge in RDF is the triple, a (subject, predicate, object) construct.

Resource Description Framework Schema (RDFS) is a language for defining RDF vocabularies. RDFS provides the notion of class and property for a resource, along with the domain and the range of a relationship. An RDFS vocabulary can contain subclasses and sub-properties.

Web Ontology Language (OWL): If RDFS expressivity is not sufficient, Web of Data vocabularies can be designed with OWL, a formal ontology language for the Semantic Web. OWL supports features such as class intersection, union and cardinality restriction.

SPARQL is the Query and Update language for the Web of Data.

2.2.1 Resource Description Framework (RDF)

LD represents RDF data in the form of node-and-arc-labeled directed graphs. This representation helps in the connection between several resources through their schema. RDF[11] allows the

[11] `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/`

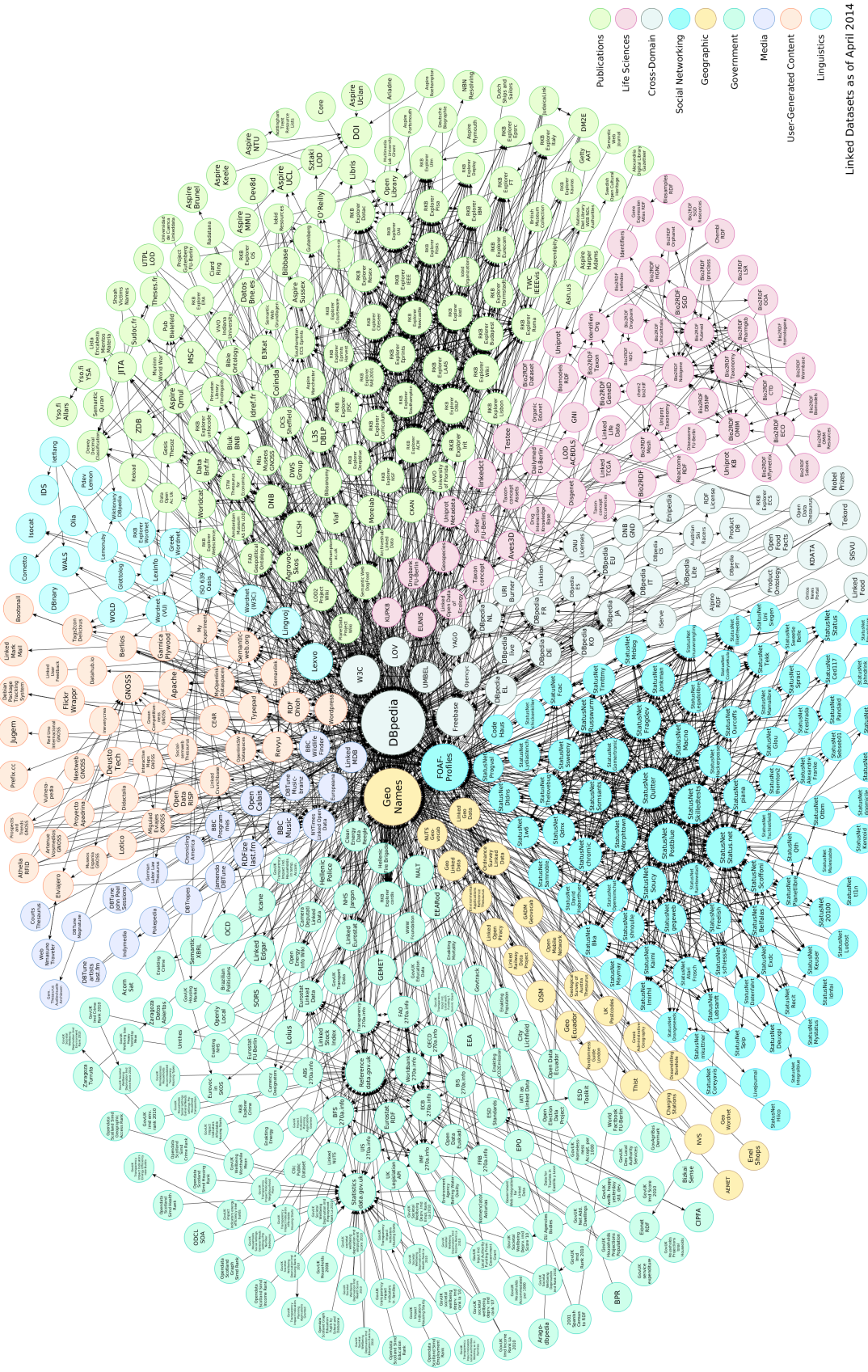


Figure 2.1: Linked Open Data Cloud 2014. Different colors represent different domains. The central hub of the LOD cloud is DBpedia.

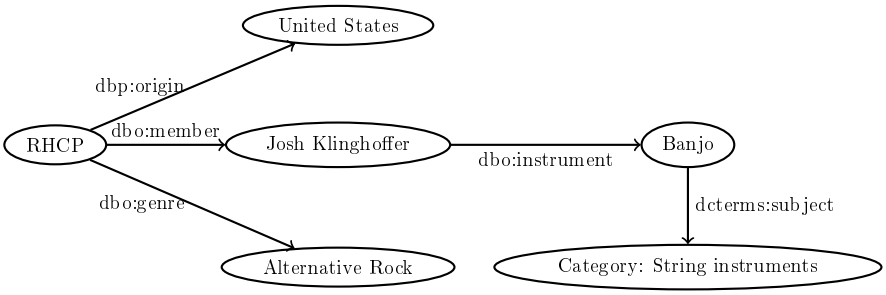


Figure 2.2: Sample RDF Graph for Musical Instruments from DBpedia. The prefixes dbo:, db:, dbp: stand for dbpedia-owl:, dbpedia: and dbpprop: respectively.

specification of the named entities with the help of URIs which are further re-used across the Web. These entities are further grouped into named classes which are related to each other through named relations. The attributes related to each entity is defined with the help of literal values.

The fundamental concepts of RDF include resources, properties and statements.

Resource: A *resource* is an object or a thing such as book, author etc.

Statement: A statement is an object-attribute-value triple, consisting of a resource, a property and a value. Value can either be resources or literals.

Literals: Literal are the values which can be a string or integer.

Blank Nodes: A blank node is a node in an RDF graph which represents a resource with a missing URI or literal.

Using the above fundamental concepts RDF triple and RDF graph are defined as follows:

Definition 14 (RDF Triple)**.** *Given a set of URIs U, blank nodes B and literals L, an RDF triple is represented as $t = (s, p, o) \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{B} \cup \mathbf{L})$, where s is a subject, p is a predicate and o is an object.*

Definition 15 (RDF Graph)**.** *A finite set of RDF triples is called as RDF Graph $\mathcal{G}$ such that $\mathcal{G} = (V, E)$, where V is a set of vertices and E is a set of labeled edges.*

Each pair of vertices connected through a labeled edge keeps the information of a statement. Each statement is represented as $\langle subject, predicate, object \rangle$ referred to as an RDF Triple. In $\mathcal{G} = (V, E)$, V includes *subject* and *object* while E includes the *predicate*. Let us consider the Musical Instrument domain. The RDF triple store (i.e., set of triples) related to this domain are shown in Table 2.1. A graphical representation of this domain is shown in Figure 2.2 These triples belong to DBpedia Bizer et al. (2009c), a central hub of LOD which extracts data from Wikipedia info boxes and makes it available in the structured format. The current version of DBpedia 3.9 contains 2.46 billion RDF triples. Out of these triples 470 million are extracted from Wikipedia in English language, 1.98 billion are in other languages and about 45 million triples link DBpedia to the external data sets.

	Triples About Stringed Musical Instruments
t1	db:RHCP[12] rdf:type dbo:Band
t2	db:Disturbed rdf:type dbo:Band
t3	db:RHCP dbp:origin db:United States
t4	db:Disturbed dbp:origin db:United States
t5	db:RHCP dbo:bandMember db:Josh_Klinghoffer
t6	db:Disturbed dbo:bandMember db:John_Moyer
t7	db:Disturbed dbo:bandMember db:Dan_Donegan
t8	db:John_Moyer dbo:instrument db:Bass_Guitar
t9	db:Dan_Donegan dbo:instrument db:Electronic_Keyboard
t10	db:Josh_Klinghoffer dbo:intsrument db:Banjo
t11	db:Josh_Klinghoffer dbo:intsrument db:Accordion
t12	db:Banjo dcterms:subject Category:Stringed_Instrument
t13	db:Bass_Guitar dcterms:subject Category:Stringed_Instrument
t14	db:Accordion dcterms:subject Category:Keyboard_Instrument
t15	db:Electronic_Keyboard dcterms:subject Category:Keyboard_Instrument

Table 2.1: RDF Triples for Musical Instruments from DBpedia. The prefixes dbo:, db:, dbp: stand for dbpedia-owl:, dbpedia: and dbpprop: respectively.

Resource Description Framework Schema (RDFS). RDF Schema[13] extends RDF data model by providing a set of classes along with certain properties, which provide structure to the RDF resources by describing basic ontology, otherwise called RDF vocabularies. A class can be thought of as a set of resources where each individual object belonging to a class is referred to as an *instance* of a class. The relationship between instance and a class can be defined in RDF with the help of the property `rdf:type`. Once we have classes, it is important to define the notion of relationship between the classes. A *subclass* relation defines a hierarchy of classes and is written as `rdfs:subClassOf`. More generally, A is a subclass of B if every instance of class A is also an instance of class B. This hierarchy may or may not be in the form of trees because one class can have multiple superclasses. If A is a subclass of both B_1 and B_2 then it means that all the instances of class A are also an instance of both the classes B_1 and B_2. Similarly, a hierarchical order can also be defined over properties referred to as property hierarchy.

Ontology Web Language (OWL). The expressivity of RDF is more or less limited to binary ground predicates and RDF schema is limited to subclass hierarchy and property hierarchy with the definitions of domain and range. On the other hand, Web Ontology Language[14] Staab and Studer (2009) provides much more expressivity as compared to RDF and RDF Schema. One of the use is to allow people to reason about knowledge. The reasoning can be about class membership, equivalence classes, consistency checking and classification. There three major types of OWL, OWL-Full, OWL-DL and OWL-Lite.

2.2.2 SPARQL

SPARQL[15] is the standard query language for RDF. This introduction is based on Arenas et al. (2009). A SPARQL query is basically composed of three parts, namely *pattern matching*, *solution modifiers* and *output*. The *pattern matching* part takes into account pattern matching features used by the graphs such as union of patterns, filtering of values etc. The *solution modifiers* include the operations which are applied after the output from the pattern matching part is obtained such as limit, distinct (other clauses like Group By, Having are also included in this part). The third part is the *output* which is of many types such as boolean queries returning yes/no answers

[13] `http://www.w3.org/TR/rdf-schema/`
[14] `http://www.w3.org/TR/owl-guide/`
[15] `http://www.w3.org/TR/rdf-sparql-query/`

(ASK keyword), selection of values of the variables matching the patterns, construction of new RDF data (through CONSTRUCT clause) and descriptions of resources (DESCRIBE keyword).

A SPARQL query can also be represented in the form of head ← body, where body includes the RDF graph patterns including conjunction disjunction and other constraints also containing variables. While the head of the query organizes the construction of answer of the query. A SPARQL query `Q` is matched against a graph $\mathcal{G}$ to obtain a set of values bound to the variables in the body. These values are then processed based on the information given in the head of `Q` to produce answers.

In the current work we will focus more on the type of queries whose output performs value selection over the variables matching the patterns. Such queries contain $SELECT$ clause with the projection over set of variables while evaluating a SPARQL query.

Now let us assume that there exists a set of variables V disjoint from U in the above definition of RDF, then $(U \cup V) \times (U \cup V) \times (U \cup V)$ is a graph pattern called a triple pattern. Let us consider a variable $?X \in V$ and $?X = c$ then $c \in U$. Given U and V a mapping μ is a partial function $\mu : V \rightarrow U$. If t is the triple pattern then $\mu(t)$ would be the triple obtained by replacing variables in t with respect to μ.

$[\![.]\!]_G$ takes an expression of patterns and returns a set of mappings. Given a mapping $\mu : V \rightarrow U$ and a set of variables $W \subseteq V$, μ is represented as $\mu_{|W}$, which is described as a mapping such that $dom(\mu_{|W}) = dom(\mu) \cap W$ and $\mu_{|W}(?X) = \mu(?X)$ for every $?X \in dom(\mu) \cap W$. Finally, the SELECT SPARQL query is defined as follows:

Definition 16. *A SPARQL SELECT query is a tuple (W, P), where P is a graph pattern and W is a set of variables such that $W \subseteq var(P)$. The answer of (W, P) over an RDF graph G, denoted by $[\![(W, P)]\!]_G$, is the set of mappings:*

$$[\![(W, P)]\!]_G = \{\mu_{|W} | \mu \in [\![P]\!]_G\}$$

In the above definition $var(P)$ is the set of variables in pattern P where as W represents the variables in the SELECT clause of the SPARQL query. In the rest of the thesis we denote W as V to avoid overlap between the attribute values W in many-valued context and variables W in SELECT clause of SPARQL query. Further details on the formalization and foundations of RDF databases are discussed in Arenas et al. (2009).

Consider a query `Q` *all the bands which play different stringed instruments along with their origin.* `Q` can not be answered by standard search engines as it generates a separate list of bands and stringed instruments requiring multiple resources to be integrated. However, `Q` can be answered by SPARQL queries over LOD. For example, let us consider the SPARQL query `Q` over DBpedia[16] shown in Listing 2.1. This query retrieves all the bands with the instruments played by their band members along with the origin of the band.

Listing 2.1: SPARQL Query Q

```
1 SELECT ?band ?instrument ?origin WHERE {
2       ?band rdf:type dbpedia-owl:Band.
3       ?band dbpprop:origin ?origin.
4       ?band dbpedia-owl:bandMember ?member.
5       ?member dbpedia-owl:instrument ?instrument .
6       ?instrument dcterms:subject dbpedia:Category:String_instruments .}
7 GROUP BY ?instrument ?origin
```

[16] `http://dbpedia.org/sparql`

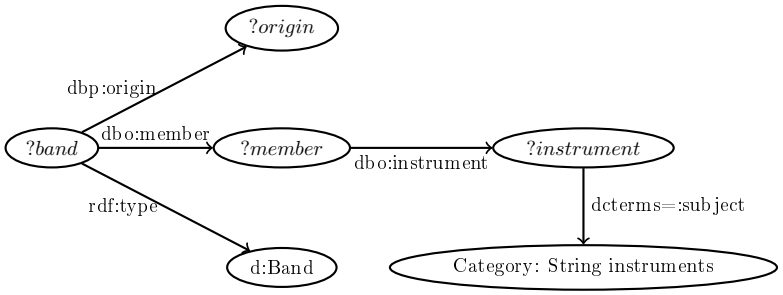


Figure 2.3: Basic Graph Pattern for query in Listing 2.1 from DBpedia matched against the RDF graph in Figure 2.2. The prefixes dbo:, db:, dbp: stand for dbpedia-owl:, dbpedia: and dbpprop: respectively.

?band	?instrument	?origin
dbpedia:RHCP	dbpedia:Banjo	dbpedia:US
dbpedia:Disturbed	dbpedia:Bass_Guitar	dbpedia:US
dbpedia:The_Solution	dbpedia:Banjo	dbpedia:Sweden

Table 2.2: Small section of results obtained as an answer to query `Q`.

The BGP for the query in listing 2.1 is shown in Figure 2.3. Here the head of the query contains `SELECT` clause, $V = \{?\texttt{band}, ?\texttt{instrument}, ?\texttt{origin}\}$ The pattern in line 2 of `Q` matches the triples {t1,t2}, line 3 select the triples {t3,t4} while line 4 and 5 select {t5,t6,t7} and {t8,t9,t10,t11} respectively. However, line 6 reduces the number of triples selected by line 3 and 4 with the help of the constraint that the instruments to be selected should be strictly stringed instruments. Finally, the selected triples by line 4, 5 and 6 are {t5,t6,t8,t7}. Moreover, line 4 and line 5 integrate the answer variable `?band` with the answer variable `?instrument` with the help of the free variable `?member`. The above SPARQL query returns a set of tuples representing a list of bands along with the instruments they play and their origin as an answer. An excerpt of the answers is shown in Table 2.2.

2.2.3 Accessing and Consuming Linked Data

Heath and Bizer (2011) mentions three different strategies for accessing the Web of Data from applications. Such techniques are compared and evaluated by Hartig and Langegger in Hartig and Langegger (2010) according to the number of data sources, the required data freshness, response time, and the need for runtime data discovery. Applications might access the Web of Data with the following strategies:

Crawling. Applications using crawling strategy traverse the desired RDF links. Afterwards, a local dump of the crawled web of data is created. Major advantage of crawling is that it improves the response time as the application works on the local dump of the data. However, this way the application works on stale and replicated data.

On-The-Fly Dereferencing. As soon as an application needs the data the URIs are dereferenced at runtime. This way the applications always operate on up-to-date and non-replicated data. One of the major limitation is that this strategy suffers from long response time and a significant network usage.

Query Federation. It relies on SPARQL queries instead of HTTP access. Applications run

complex queries over the desired SPARQL endpoint without needing to replicate the data locally. The drawback of query federation is to find an efficient query execution plan for join queries over huge number of data sources.

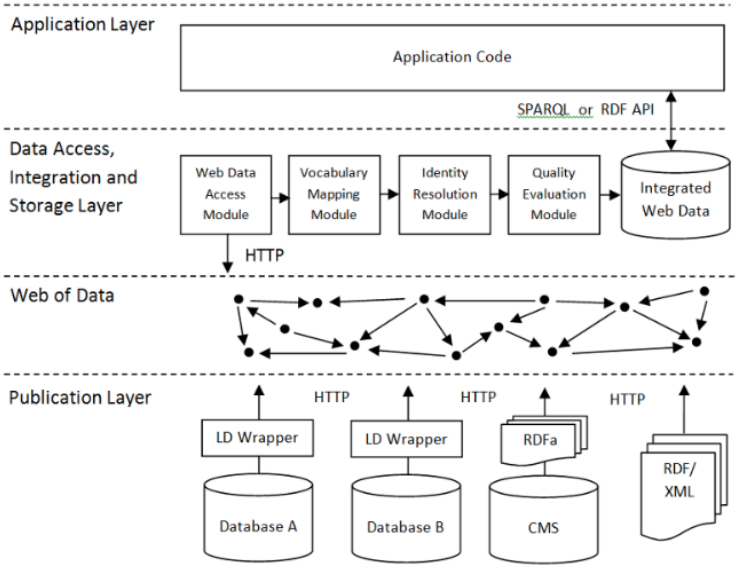


Figure 2.4: An overview of the architecture of a Linked Data application

Figure 2.4 (taken from Heath and Bizer (2011)) illustrates the architecture of a Linked Data application which implements crawling. All the data published using the Linked Data principles, becomes part of a global graph depicted as the Web of Data Layer in Figure 2.4. Applications implementing any of the above defined access strategies (i.e., crawling, on-the-fly dereferencing and query federation) usually implement the modules shown in the Data Access, Integration and Storage Layer. The functionalities of each of these modules are given below:

Accessing the Web of Data. The basic means of accessing data as described before are crawling and on-the-fly dereferencing. Additionally, data can also be retrieved by using Linked Data search engines. Queries over SPARQL endpoint and RDF data dumps are also used as a way of accessing Web of Data.

Vocabulary Mapping. Different resources over Linked Data use different RDF vocabularies for similar information. For understanding this data, the applications transform these terms from different vocabularies into a single target schema. The terms which are unknown to the application search for the mappings over the web and apply the resulting mappings for transformation.

Identity Resolution. Different resources use different URIs to identify the same entity. Some data sources provide `owl:sameAs` links which point to the data about same entity present on other resources. In case if these links are not provided identity resolution can be applied by the applications to discover such links.

Provenance Tracking. For efficient processing of data, it is often cached locally by the applications and these data may belong to several data sources. It is important to keep track of data provenance in case if the original data source is to be visited again and to assess the quality of the data.

Data Quality Assessment. As the Web is open, the Web data should be considered as information provided by different sources rather than as facts. If the application uses only small,

known and authentic data sets then quality is not an issue. However, if the applications pull data from the open Web, a data quality assessment method needs to be employed by the applications to reject the untrustworthy resources.

Using the Data in the Application Context. After completing the above described tasks, the application has integrated and cleansed Web data to an extent that is required for more sophisticated processing. This processing may involve simply displaying the data to the user for other purposes such as application of KDD process or provide user interaction using various visualizations.

2.3 Clustering SPARQL Query Answers.

Like Web Clustering Engines, a similar problem of huge amount of results obtained by search results is encountered while querying the Semantic Web data. Semantic Web data is accessible through SPARQL query language. Even though semantic web is structured, the problem arises that user may not know the granularity of the data in advance and the user will pose a general query. In this case, she will face the problem of huge amount of results returned by some SPARQL queries. Even if the GROUP BY clause is used for a variable in the SPARQL query, the problem is that if there are too many answers to this variable then the query will generate many small groups. Moreover, these answers are shown in the form of lines. Such kind of answers are not easily interpretable by users. In order to make these answers more observable several studies have been proposed which cluster the answers obtained by SPARQL query, that we discuss in this section.

In Figure 2.5, we propose an overall architecture of how this problem can be solved by applying the architecture of web clustering engines on RDF data. As a first step, the user sends a SPARQL query with the solution modifier. This SPARQL query is then used to acquire the answers. This query with the solution modifier and the query answers are then fed for preprocessing. During this process the answers are organized to be processed by the suitable clustering algorithm. Then the clustering algorithm generates clusters with or without background knowledge. The background knowledge shown in the figure is in dotted box because it is optional. Afterwards, the clusters are visualized most commonly using tree-folder structure which allows the user to visualize and interpret the results. Following these lines, this section discusses various studies conducted for targeting this problem following the described architecture.

CLUSTER BY Clause. In Lawrynowicz (2009b), the author proposes a declarative way of invoking the query answer clustering. The authors discuss the overall architecture of how the clustering is performed and how the answers are allocated to the clusters. It extends the SPARQL query specification with a solution modifier i.e., `CLUSTER BY` clause. This clause is placed after the pattern matching part of the SPARQL query. It takes the answers generated by the query and applies the clustering algorithm. Afterwards, a new clustered output is shown to the user. This study implements several clustering algorithms such as hierarchical clustering, K-means etc. It allows the user to either specify the clustering algorithm to be used or leave it unspecified to allow the system to use the default clustering algorithm. For doing so, a new clause `USING` is introduced which is followed by the identifier of the clustering method or a list of identifiers.

Conceptual Clustering of SPARQL Query Answers. In Lawrynowicz (2009a) the authors improve the previous study by providing conceptual clustering over query answers. It takes into

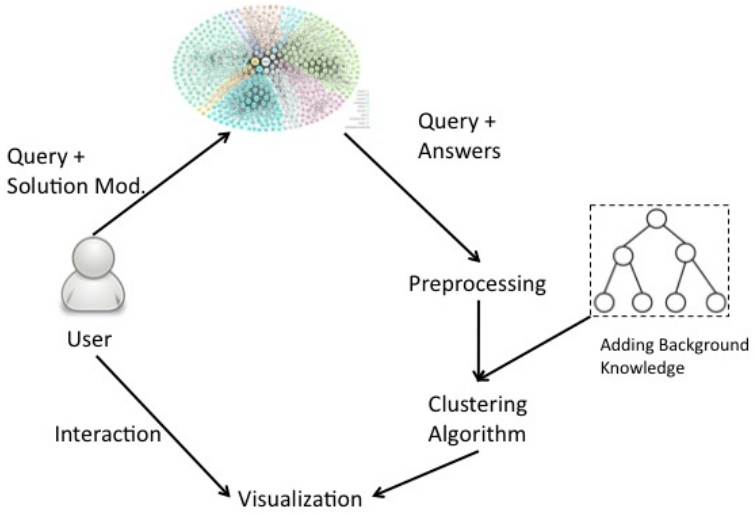


Figure 2.5: Overall Architecture for Clustering SPARQL Query Results.

account the answers generated by conjunctive queries posed over Knowledge Bases (KB) represented as OWL. It uses incremental conceptual clustering algorithm such as COBWEB Fisher (1987). COBWEB generates cluster description while generating the cluster. The COBWEB algorithm takes a set of items represented as set of attribute value pairs as input. Based on this representation it generates hierarchical classification tree over the given set of items. As COBWEB builds the classification tree incrementally, it starts from the root node and descends down the tree by applying one of the four operators, i.e., *adding* the item to an existing class, *creating* a new singleton class for the item, *merging* two sibling classes and *splitting* a class into several classes.

Now the problem is, how these attribute-value pairs can be obtained to be used by the COBWEB algorithm. For doing so, the authors construct the feature sets reflecting the similarity (or dissimilarity) of an item referred to as *semantic* features. For designing the feature-vector representation for the COBWEB algorithm the authors introduce a feature function. This feature function f maps (a, C), where a denotes an individual, C denotes a concept from description logic knowledge base, to values from $\{true, false\}$, $f : (a, C) \rightarrow \{true, false\}$. Now this feature function constructs feature vector for each individual. Let $\mathbf{C} = \{C_1, C_2, \ldots, C_m\}$ be a vector of concepts in the knowledge base then a feature vector for an individual a is computed as $f(a, C_i)$ where $i \in \{1, \ldots, m\}$. Here $f(a, C_i) = true$ is the individual $C_i(a)$ is entailed by the knowledge base KB otherwise $f(a, C_i) = false$.

As the authors are using the `CLUSTER BY` clause they do not have to extract features for all the individuals. They only extract the features for the individuals which are solution to the variable in the CLUSTER BY clause.

CATEGORIZE BY Clause. In d'Amato et al. (2010) the authors introduce a novel way of grouping query answers (semantic grouping) where grouping is done on the ground of a knowledge base of reference. The proposed technique aggregates query results consisting on a dynamic generation of a navigable hierarchy on top of the retrieved results and that is based on their semantics. Such a hierarchy constitutes a multi-valued classification of the results that may be seen as a novel approach for generating a dynamic faceted classification over retrieved results, enabling further faceted search/browsing over them. The authors introduce a new clause

`Categorize By` for enabling the semantic grouping of the answers of conjunctive queries.

A conjunctive query with a semantic aggregate subgoal is of the form *categorize_by* ($[X_1, X_2, \ldots, X_m]$) : $Q(x, y)$ where $X_1, X_2, \ldots, X_m$ is a grouping list of variables in x which is a distinguished variable and y is a non-distinguished variable. Following the line, the *semantic category* is given as follows: Given a query Q = *categorize_by* ($[X_1, X_2, \ldots, X_m]$) : $Q(x, y)$, a semantic category is a tuple of concepts $C_1, C_2, \ldots, C_m$, where each C_i corresponds to X_i in the grouping variables list of Q. Then, the operational semantics for categorize by clause is to first create $\mathcal{H}$, a partially ordered set of semantic categories, based on inferred semantic types of grouping variables, and then to partition the input relation to groups with equal values for the same semantic categories. A concept corresponding to each query variable included in the CATEGORIZE BY clause is derived. Then it is classified in the subsumption hierarchy of the concepts in the knowledge base. The authors focus on subtrees of concepts, generated for each grouping variable. For dealing with multiple variables in the `Categorize By` clause, the authors take a tree product of the extracted hierarchy.

In case of single variable, this may not be enough to obtain meaningful hierarchy of groups. Consider for example the situation, where the concept C_i is a leaf in the classified subsumption hierarchy. Another case where this basic approach would not be of help is when all the retrieved results for variable v_i fall under the same type. To overcome this issue, a refinement operator is introduced in Lawrynowicz et al. (2010).

A Refinement Operator Based Method for Grouping Query Results. To overcome the above problems, in Lawrynowicz et al. (2010), the authors propose a new method for generation of semantic groups. For each grouping variable X_i, the algorithm first tries to infer a type of a variable by exploiting the information from the query atoms. Firstly, the explicit typing represented by those concepts explicitly mentioned in the query atoms $C(X_i)$ is considered. Additionally, the implicit types inferred by the role atoms $R(X_i, \Delta)$ or $R(\Delta, X_i)$, respectively the domain and range of role R are added. The concept determined for each query variable is $C_i := \bigsqcap_{C_p^i \in \mathcal{B}_i} C_p^i$. If any C_i is a top concept, then the algorithm performs one scan on the results to derive the variable typing. In the next step, a KB is classified, and each inferred concept C_i is placed in the proper place in the subsumption hierarchy. If a hierarchy for some C_i has less levels than a user-specified MAXDEPTH threshold then the refinement operator is applied to further expand the hierarchy. Then, all the hierarchies obtained for grouping variables are fed into a tree product operator. A tree product is a binary operation on trees, which takes two trees T_1 and T_2 as an input, and produces a tree T whose vertex set is a subset of the Cartesian product, and two product vertices (u_1, u_2) and (v_1, v_2) are connected in T , iff u_1, u_2, v_1, v_2 satisfy conditions of a certain type in T_1 and T_2 . Multiple variables are handled by an iterative application of a tree product. Finally, the results (tuples) are assigned to the most specific semantic categories (populate).

A system with the name ASPARAGUS Lawrynowicz et al. (2011) was developed which implements `Cluster By` and `Categorize By` clause with all the above defined grouping functionalities. This system takes the SPARQL endpoint, SPARQL query and the Ontology as input and generates the clustered SPARQL query results. Figure 2.6 shows the system ASPARAGUS. (1) takes the SPARQL query with the new solution modifier, (4) takes the SPARQL endpoint and keeps details of the clustering algorithms used, (2) shows the hierarchy of DL-concept generated after executing the query and applying the clustering algorithm and finally, (3) keeps the instances related to each of the concept.

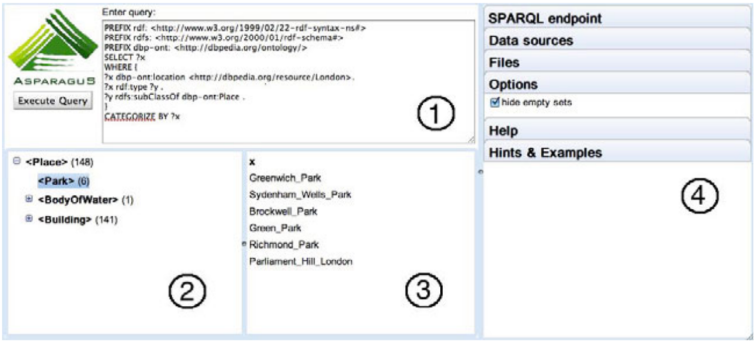


Figure 2.6: ASPARAGUS System for clustering SPARQL query answers.

2.4 RDF Graphs and Formal Concept Analysis

Conceptual Navigation of RDF Graphs. Ferré (2010) introduces LIS (Logical Information System) which allows conceptual navigation for browsing of RDF graphs, where concepts are accessed through SPARQL-like queries. The user interface gives a local view of the concept lattice, centered on a concept called the focus. The local view is made of three parts: (1) the query, (2) the extent, and (3) the index. The query is a logical formula. The extent is the set of objects that are matched by the query and identifies the focus concept. Finally, the index is a finite subset of the logic that is restricted to formulas that match at least one object in the extent. The index plays the role of a summary of the extent, showing which kinds of objects there are, and how many of each kind there are.

Several navigation modes are provided by LIS are zoom-in, naming and reversal. These navigation link are consistent and complete. A navigation link is *consistent* w.r.t. a dataset iff the query does not return empty results. A navigation graph is *consistent* iff its links are all consistent. A navigation graph is *complete* iff for every query whose extent is not empty, there exists a finite sequence of navigation links.Figure 2.7 shows the interface of Camelis2.

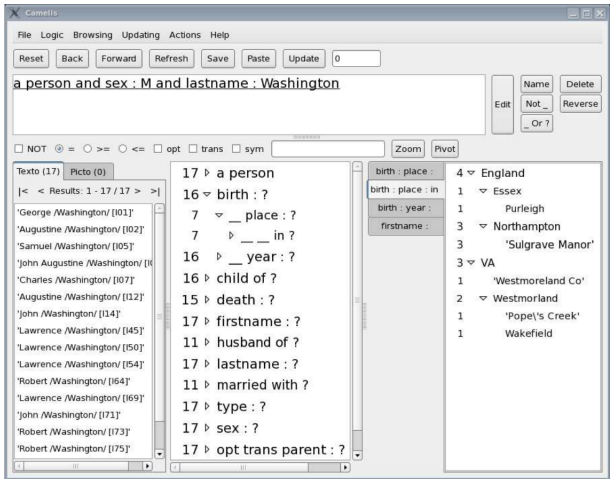


Figure 2.7: Conceptual Navigation of RDF Graphs using (Sewelis) Camelis 2.

SPARKLIS. Ferré (2014a,c) is a tool for exploring SPARQL endpoints which guides the user on each step in building question and answers through interaction. It combines faceted

search, expressivity of SPARQL and readability of natural language. The goal of SPARKLIS is to provide a tool for making the semantic web data more accessible and enable the user to define and send SPARQL queries to endpoints. It guides the user step-by-step in selection process for creating SPARQL queries . Each selection leads to a set of suggestions to refine the current selection. User only pick one of the suggestions according to her preference. It provides overview and feedback during this the search process hence supporting the exploratory search. In other words, SPARKLIS can be called as a SPARQL query builder.

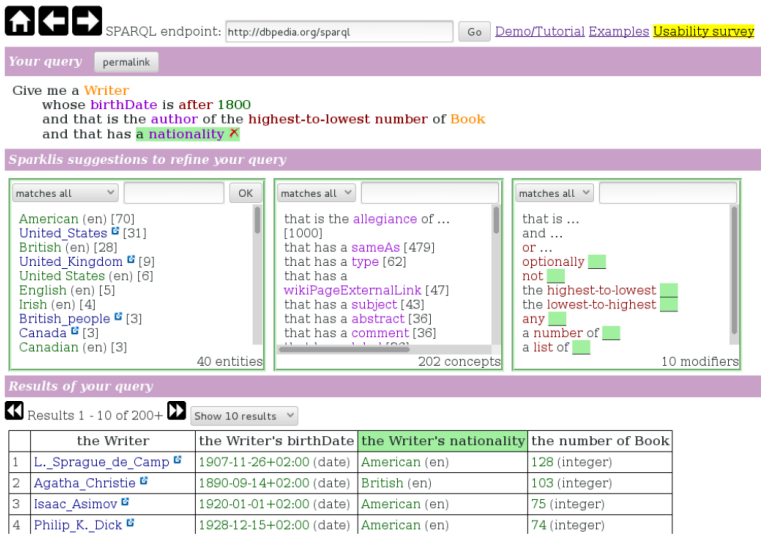


Figure 2.8: SPARKLIS Query Builder.

2.5 Discussion

In this chapter, we move from web of documents to the web of data and we discuss how the architecture of web clustering engines is applied to RDF data and discusses some of the previous works. Finally, it gives several FCA-based frameworks introduced to deal with RDF graphs. Table 2.3 gives comparison between the interfaces of different tools for navigating concept lattice detailed in Chapter 1 and 2. It also distinguishes the systems which are explicitly web clustering engines. The first property *Tree-Folder Display* groups the systems using this layout for visualization. *Hasse Diagram* refers to those systems which display concept lattice in its original form. The group *Lattice in the Background* keeps those systems where a user friendly interface is shown to the user and she does not know that a concept lattice is at work in the background. Some of the systems allow *Interactive Lattice Creation* meaning that the user can edit the context or provide feedback through the concept lattice, which can then be refined. The property *Querying Lattices* means that the systems contain the functionality of querying the lattices. The third last category keeps the systems dealing with RDF-Graphs. *non-FCA* groups the systems which are not based on FCA. Finally, *Web Clustering Eng.* classifies web clustering engines. In the rest of this thesis, we provide interactive exploration mechanisms over RDF graphs by classifying RDF triples and allowing the user to navigate through these classes to obtain useful information.

	Tree of concepts	Hasse Diagram	Local View	Interactive Lattice Creation	Querying	RDF Graphs	non-FCA	Web Clustering Engine
Camelis Ferré (2009)	✓		✓		✓			
CreChainDo Nauer and Toussaint (2007)	✓			✓				✓
CREDO Carpineto and Romano (2004b)	✓							✓
JBrainDead Cigarrán et al. (2004)	✓							✓
ASPARAGUS Lawrynowicz et al. (2011)	✓					✓	✓	
CEM Cole and Stumme (2000)	✓							
Cole and Eklund (1999)	✓							
Search Sleuth Dau et al. (2008)	✓							✓
FooCA Koester (2006)		✓	✓	✓				✓
Carpineto and Romano (1996b)		✓	✓					
ULYSSES Carpineto and Romano (1995)		✓	✓					
Conexp		✓	✓					
Galicia		✓	✓					
Museum Collection Wray et al. (2013)			✓					
Image Sleuth Ducrou et al. (2006)			✓					
DVD Sleuth Ducrou (2007)			✓					
Eklund et al. (2004)								
SPARKLIS Ferré (2014a)			✓			✓		
BR-Explorer Messai et al. (2006)		✓			✓			
RV-Explorer		✓				✓		✓
Sewelis Ferré (2010)		✓	✓		✓	✓		

Table 2.3: Comparison between different tools developed for retrieving specific information from a concept lattice.

Chapter 3

Lattice-Based View Access

Contents

Like queries across search engines, SPARQL queries over semantic web data usually produce list of tuples as answers that may be hard to understand and interpret. Accordingly, this chapter focuses on Lattice-Based View Access (LBVA), a framework based on FCA. This framework provides a classification of the answers of SPARQL queries based on a concept lattice, that can be navigated for retrieving or mining specific patterns in query results. In this way, the concept lattice can be considered as a materialized view of the data resulting from a SPARQL query. In this chapter we also give implementation details about the tool named as RV-Xplorer (Rdf View eXplorer) for navigating/exploring the view resulting from SPARQL query enhanced by solution modifier `View By`. We discuss the support provided to the expert for answering certain questions through the navigation strategies provided by RV-Xplorer. We also compare our system with the existing state of the art approaches. The description of Lattice-Based View Access is based on Alam and Napoli (2014a,b) and the description of the tool RV-Xplorer is based on Alam et al. (2015d).

3.1 Introduction

At present, Web has become a potentially large repository of knowledge, which is becoming main stream for querying and extracting useful information. In particular, Linked Open Data (LOD) Bizer et al. (2009a) provides a method for publishing structured data in the form of RDF resources. These RDF resources are interlinked with each other to form a cloud. SPARQL queries are used in order to make these resources usable, i.e., queried. In some cases, queries in natural language against standard search engines can be simple to use but sometimes they are complex and may require integration of data sources. Then the standard search engines will not be able to easily answer these queries, e.g., *Currencies of all G8 countries*. Such a complex query can be formalized as a SPARQL query over data sources present in LOD cloud through SPARQL endpoints for retrieving answers. Moreover, users may sometimes execute queries which generate huge amount of results giving rise to the problem of information overload d'Amato et al. (2010). A typical example is given by the answers retrieved by search engines, which mix between several meanings of one keyword. In case of huge results, user will have to go through a lot of results to find the interesting ones, which can be overwhelming without any specific navigation tool. Same is the case with the answers obtained by SPARQL queries, which are huge in number and it may be harder to extract the most interesting patterns. This problem of information overload raises new challenges for data access, information retrieval and knowledge discovery w.r.t web querying.

Accordingly, this chapter proposes a new approach based on Formal Concept Analysis (FCA Ganter and Wille (1999))s. It describes a lattice-based classification of the results obtained by SPARQL queries by introducing a new clause `VIEW BY` in SPARQL query. This framework, called Lattice-Based View Access (LBVA), allows the classification of SPARQL query results into a concept lattice, referred to as a *view*, for data analysis, navigation, knowledge discovery and information retrieval purposes. This new clause `VIEW BY` which enhances the functionality of already existing `GROUP BY` clause in SPARQL query by adding sophisticated classification and Knowledge Discovery aspects. Here after, we describe how a lattice-based view can be designed from a SPARQL query. Afterwards, a view is accessed for analysis and interpretation purposes which are totally supported by the concept lattice. In case of large data only a part of the lattice Stumme et al. (2001) can be considered for the analysis. In this way, this chapter investigates also the capabilities of FCA to deal with semantic web data.

The intuition of classifying results obtained by SPARQL queries is inspired by web clustering

engines Carpineto et al. (2009) such as Carrot2[17]. The general idea behind web clustering engines is to group the results obtained by query posed by the user based on the different meanings of the terms related to a query. Such systems deal with unstructured textual data on web. By contrast, there are some studies conducted to deal with structured RDF data. One such system is ASPARAGUS Lawrynowicz et al. (2011) which deductively groups the results by taking into account the subsumption hierarchy defined in knowledge bases. It takes SPARQL endpoint and background knowledge along with a SPARQL query as an input from the user and generates clusters of the results obtained by the given SPARQL query based on the given background knowledge. In d'Amato et al. (2010), the authors introduce a clause `Categorize By` to target the problem of managing large amounts of results obtained by conjunctive queries with the help of subsumption hierarchy present in the knowledge base. By contrast, the `VIEW BY` clause generates lattice-based views which provide a mathematically well-founded classification based on formal concepts and an associated concept lattice. Moreover, it also paves way for navigation or information retrieval by traversing the concept lattice and for data analysis by allowing the extraction of association rules from the lattice. Such data analysis operations allow discovery of new knowledge. Additionally, unlike `Categorize By`, `VIEW BY` can deal with data that has no schema (which is often the case with linked data). Moreover, `VIEW BY` has been evaluated over very large set of answers (roughly 100,000 results) obtained over real datasets. In case of larger number of answers, `Categorize By` does not provide any pruning mechanism while this chapter describes how the views can be pruned using iceberg lattices.

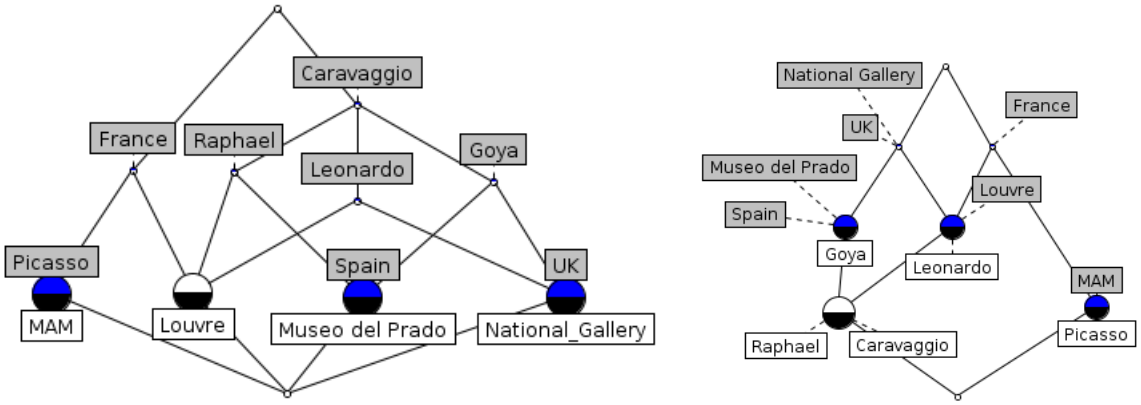
3.1.1 Motivation

In this section we introduce a motivating example focusing on why LOD should be queried and why the SPARQL query results need classification. We will continue this scenario to describe the proposed framework. Let us consider that a query `Q` searching for *museums where the exhibition of some famous artists is taking place along with the location of the museum.* Here, we do not discuss the interface aspects and we will assume that SPARQL queries are provided. A standard query engine is not adequate for answering such kind of questions and a direct query over LOD will give better results. One of the ways to obtain such an information is to query LOD through its SPARQL endpoint. This query will generate a huge amount of results, which will need further manual work to group the interesting links. According to the scenario, a SPARQL query can be given as follows:

```
1 SELECT ?museum ?country ?artist WHERE {
   2 ?museum rdf:type dbpedia-owl:Museum .
   3 ?museum dbpedia-owl:location ?city .
   4 ?city dbpedia-owl:country ?country .
   5 ?painting dbpedia-owl:museum ?museum .
   6 ?painting dbpprop:artist ?artist}
7 GROUP BY ?country ?artist
```

This query retrieves the list of museums along with the artists whose work is exhibited in a museum along with the location of a museum. `Lines 5 and 6` retrieve information about the artists whose work is displayed in some museum. More precisely, the page containing the information on a museum (`?museum`) is connected to the page of the artists (`?artist`) through a page

[17] `http://project.carrot2.org/index.html`



(a) Classes of Museums w.r.t Artists and Countries, e.g., the concept on the top left corner with the attribute `France` contains all the French Museums, i.e., Musee du Louvre (Louvre) and Musee d'Art Moderne (MAM). (`VIEW BY ?museum`)

(b) Classes of Artists w.r.t Museums and Countries. (`VIEW BY ?artist`)

Figure 3.1: Lattice-Based Views w.r.t Museum's and Artist's Perspective .

on the work of artist (`?painting`) displayed in the museum. In order to integrate these three resources, two predicates were used `dbpedia-owl:museum` and `dbpprop:artist`. An excerpt of the answers obtained by `Group by` clause is shown below:

```
Pablo_Picasso       Musee_d'Art_Moderne   France
Leonardo_Da_Vinci   Musee_du_Louvre       France
Raphael             Museo_del_Prado       Spain
```

The problem encountered while browsing such an answer is that there are too many statements to navigate through. Even after using the `GROUP BY` clause the answers are not organized in any ordered structure. By contrast, the clause `VIEW BY` activates the LBVA framework, where the user will obtain a classification of the statements as a concept lattice where statements are partially ordered (see Figure 3.1a). To obtain the museums in `UK` displaying the work of `Goya`, all the museums displaying the work of `Goya` can be retrieved and then the specific concept containing `Goya and UK` is obtained by navigation. The answer obtained is `National Gallery` in the example.

3.2 Lattice-Based View Access

3.2.1 SPARQL Queries with Classification Capabilities

The idea of introducing a `VIEW BY` clause is to provide classification of the results and add a knowledge discovery aspect to the results w.r.t the variables appearing in `VIEW BY` clause. Let `Q` be a SPARQL query of the form `Q = SELECT ?X ?Y ?Z WHERE {pattern P} VIEW BY ?X` then the set of variables $V = \{?X, ?Y, ?Z\}$ [18]. According to the definition 16 the answer of the tuple (V, P) is represented as $[\![(\{?X, ?Y, ?Z\}, P)]\!] = \mu_i$ where $i \in \{1, \ldots, k\}$ and k is the number of mappings obtained for the query `Q`. For the sake of simplicity, $\mu_{|W}$ is given as μ. Here, $dom(\mu_i) = \{?X, ?Y, ?Z\}$ which means that $\mu(?X) = X_i$, $\mu(?Y) = Y_i$ and $\mu(?Z) = Z_i$. Finally, a complete set of mappings can be given as $\{\{?X \rightarrow X_i, ?Y \rightarrow Y_i, ?Z \rightarrow Z_i\}\}$.

[18] As W represents set of attribute values in the definition of a many-valued formal context, we represent the variables in select clause as V to avoid confusion.

	?museum	?artist	?country
μ_1	Musee_d'Art_Moderne	Pablo_Picasso	France
μ_2	Museo_del_Prado	Raphael	Spain
⋮	⋮	⋮	⋮

Table 3.1: Generated Mappings for SPARQL Query `Q`

The variable appearing in the `VIEW BY` clause is referred to as object variable[19] and is denoted as Ov such that $Ov \in V$. In the current scenario $Ov = \{?X\}$. The remaining variables are referred to as attribute variables and are denoted as Av where $Av \in V$ such that $Ov \cup Av = V$ and $Ov \cap Av = \varnothing$, so, $Av = \{?Y, ?Z\}$.

Example 10. *Following the example in section 3.1.1, an alternate query with the* `VIEW BY` *clause can be given as:*

```
SELECT ?museum ?artist ?country WHERE {
    ?museum rdf:type dbpedia-owl:Museum .
    ?museum dbpedia-owl:location ?city .
    ?city dbpedia-owl:country ?country .
    ?painting dbpedia-owl:museum ?museum .
    ?painting dbpprop:artist ?artist}
VIEW BY ?museum
```

Here, V={`?museum, ?artist, ?country`}* and P is the conjunction of patterns in the* `WHERE` *clause then the evaluation of $[\![(\{?museum, ?artist, ?country\}, P)]\!]$ will generate the mappings shown in Table 3.1. Accordingly, $dom(\mu_i) = \{?museum, ?artist, ?country\}$. Here, $\mu_1(?museum) = Musee\_d'Art\_Moderne$, $\mu_1(?artist) = Pablo\_Picasso$ and $\mu_1(?country) = France$. We have Ov = {*`?museum`*}* because it appears in the `VIEW BY` *clause and Av = {*`?artist`*, `?country`*}. Figure 3.1a shows the generated view when Ov = {*`?museum`*}* and in *Figure 3.1b, we have; Ov = {*`?artist`*}* and Av = {*`?museum`*, `?country`*}.

3.2.2 Designing a Formal Context of Answer Tuples

The results obtained by the query are in the form of set of tuples, which are then organized as a many-valued context.

Obtaining a Many-Valued Context (G, M, W, I): As described previously, we have $Ov = \{?X\}$ then $\mu(?X) = \{X_i\}_{i\in\{1,\ldots,k\}}$, where X_i denote the values obtained for the object variable and the corresponding mapping is given as $\{\{?X \rightarrow X_i\}\}$. Finally, $G = \mu(?X) = \{X_i\}_{i\in\{1,\ldots,k\}}$. Let $Av = \{?Y, ?Z\}$ then $M = Av$ and the attribute values $W = \{\mu(?Y), \mu(?Z)\} = \{\{Y_i\}, \{Z_i\}\}_{i\in\{1,\ldots,k\}}$. The corresponding mapping for attribute variables are $\{\{?Y \rightarrow Y_i, ?Z \rightarrow Z_i\}\}$. In order to obtain a ternary relation, let us consider an object value $g_i \in G$ and an attribute value $w_i \in W$ then we have $(g_i, \text{“}?Y'', w_i) \in I$ iff $?Y(g_i) = w_i$, i.e., the value of g_i for attribute $?Y$ is w_i, $i \in \{1, \ldots, k\}$ as we have k values for $?Y$.

Obtaining Binary Context (G, M, I): Afterwards, a conceptual scaling used for binarizing the many-valued context, in the form of (G, M, I). Finally, we have $G = \{X_i\}_{i\in\{1,\ldots,k\}}$, M =

[19] The object here refers to the object in FCA.

Museum	*Artist*	*Country*
Musee du Louvre	{Raphael, Leonardo Da Vinci, Caravaggio}	{France}
Musee d'Art Moderne	{Pablo Picasso}	{France}
Museo del Prado	{Raphael, Caravaggio, Francisco Goya}	{Spain}
National Gallery	{Leonardo Da Vinci, Caravaggio, Francisco Goya}	{UK}

Table 3.2: Many-Valued Context (Museum).

	Artist					*Country*		
Museum	Raphael	Da Vinci	Picasso	Caravaggio	Goya	France	Spain	UK
Musee du Louvre	×	×		×		×		
Musee d'Art Moderne			×			×		
Museo del Prado	×			×	×		×	
National Gallery		×		×	×			×

Table 3.3: Formal Context $\mathcal{K}_{tuple}$ w.r.t $?museum$.

Museum	Raphael	Da Vinci	Picasso	Caravaggio	Goya
Musee du Louvre	×	×		×	
Musee d'Art Moderne			×		
Museo del Prado	×			×	×
National Gallery		×		×	×

Table 3.4: S_{artist}.

Museum	France	Spain	UK
Musee du Louvre	×		
Musee d'Art Moderne	×		
Museo del Prado		×	
National Gallery			×

Table 3.5: $S_{country}$.

$\{Y_i\} \cup \{Z_i\}$ where $i \in \{1, \ldots, k\}$ for object variable $Ov = \{?X\}$. The binary context obtained after applying the above transformations to the SPARQL query answers w.r.t to object variable is called the *formal context of answer tuples* and is denoted by $\mathcal{K}_{tuple}$.

Example 11. *In the example* $Ov = \{?\texttt{museum}\}$, $Av = \{?\texttt{artist}, ?\texttt{country}\}$. *The answers obtained by this query are organized into a many-valued context as follows: the distinct values of the object variable* `?museum` *are kept as a set of objects, so* $G = \{MuseeduLouvre, MuseodelPrado, \ldots\}$, *attribute variables provide* $M = \{artist, country\}$, $W_1 = \{Raphael, LeonardoDaVinci, \ldots\}$ *and* $W_2 = \{France, Spain, UK, \ldots\}$ *in a many-valued context. The obtained many-valued context is shown in Table 3.2. Following the above defined procedure a many-valued context is conceptually scaled to obtain a binary context shown in Table 3.3. Table 3.4 and Table 3.5 show the scales* $S_{instrument}$ *and* S_{origin} *for the attributes instrument and origin respectively. The corresponding concept lattice is shown in Figure 3.1(a).*

The organization of the concept lattice is depending on the choice of object variable and the attribute variables. Then, to group the artists w.r.t the museums where their work is displayed and the location of the museums, the object variable would be `?artist` and the attribute variables will be `?museum` and `?country`. Then, the scaling can be performed for obtaining a formal context. In order to complete the set of attribute, domain knowledge can also be taken into account, such as the the ontology related to the type of artists or museums. This domain knowledge can be added with the help of pattern structures, an approach linked to FCA, on top of many-valued context without having to perform scaling. For the sake of simplicity, we do not discuss it in this study.

3.2.3 Building a Concept Lattice

Once the context is designed, the concept lattice can be built using an FCA algorithm.There are some very efficient algorithms that can be used Ganter and Wille (1999); van der Merwe et al. (2004). However, in the current implementation we use AddIntent van der Merwe et al. (2004) which is an incremental concept lattice construction algorithm. In case of large data iceberg lattices can be considered Stumme et al. (2001). The use of `VIEW BY` clause activates the process of LBVA, which transforms the SPARQL query answers (tuples) to a formal context $\mathcal{K}_{tuples}$ through which a concept lattice is obtained which is referred to as a *Lattice-Based View*. A view on SPARQL query in section 3.1.1, i.e, a concept lattice corresponding to Table 3.3 is shown in Figure 3.1a.

3.2.4 Interpretation Operations over Lattice-Based Views

A formal context effectively takes into account the relations by keeping the inherent structure of the relationships present in LOD as object-attribute relation. When we build a concept lattice, each concept keeps a group of terms sharing some attribute (i.e., the relationship with other terms). This concept lattice can be navigated for searching and accessing particular LOD elements through the corresponding concepts within the lattice. It can be drilled down from general to specific concepts or rolled up to obtain the general ones which can be further interpreted by the domain experts. For example, in order to search for the museums where there is an exhibition of the paintings of `Caravaggio`, the concept lattice in Figure 3.1(a) is explored levelwise. It can be seen that the paintings of `Caravaggio` are displayed in `Musee du Louvre, Museo del Prado and National Gallery`. Now it can be further filtered by country, i.e., look for `French` museums displaying `Caravaggio`. The same lattice can be drilled down and `Musee du Louvre` as an answer can be retrieved. Next, to check the museums located in `France` and `Spain`, the roll up operation from the French Museums to the general concept containing all the museums with Caravaggio's painting can be applied and then the drill down operation to Museums in `France` or `Spain` displaying `Caravaggio` can be performed. The answer obtained will be `Musee du Louvre` and `Museo del Prado`.

A different perspective on the same set of answers can also be retrieved, meaning that the group of artists w.r.t museums and country. For selecting French museums according to the artists they display, the object variable will be $Ov = \{?artist\}$ and attribute variables will be $Av = \{?museum, ?country\}$. The lattice obtained in this case will be from Artist's perspective (see Figure 3.1b). Now, it is possible to retrieve `Musee du Louvre` and `Musee d'Art Moderne`, which are the French museums and to obtain a specific French museum displaying the work of `Leonardo Da Vinci` a specific concept can be selected which gives the answer `Musee du Louvre`.

FCA provides a powerful means for data analysis and knowledge discovery. `VIEW BY` can be seen as a clause that engulfs the original SPARQL query and enhances it's capabilities by providing *views* which can be reduced using iceberg concept lattices. Iceberg lattices provide the top most part of the lattice filtering out only general concepts. The concept lattice is still explored levelwise depending on a given threshold. Then, only concepts whose extent is sufficiently large are explored, i.e., the support of a concept corresponds to the cardinal of the extent. If further specific concepts are required the support threshold of the iceberg lattices can be lowered and the resulting concept lattice can be explored levelwise.

Knowledge Discovery: Among the means provided by FCA for knowledge discovery, the Duquenne-Guigues basis of implications takes into account a minimal set of implications which

represent all the implications (i.e., association rules with confidence 1) that can be obtained by accessing the *view* i.e., a concept lattice. For example, implications according to Figure 3.1(a) state that all the museums in the current context which display `Leonardo Da Vinci` also display `Caravaggio` (rule: `Leonardo Da Vinci` → `Caravaggio`). It also says that only the museums which display the work of `Caravaggio` display the work of `Leonardo Da Vinci` Such a rule can be interesting if the museums which display the work of both `Leonardo Da Vinci` and `Caravaggio` are to be retrieved. The rule `Goya, Raphael, Caravaggio` → `Spain` suggests that there exists a museum which have works of `Goya, Raphael, Caravaggio` only in Spain, more precisely `Museo Del Prado`. (These rules are generated from only the part of SPARQL query answers shown as a context in Table 3.3).

3.3 Tool for Interaction with SPARQL Query Answers

In this section we continue Lattice-Based View Access (LBVA) Alam and Napoli (2014a), which provides a view over RDF graphs through SPARQL queries to give complete understanding of a part of RDF graph that expert wants to analyze with the help of Formal Concept Analysis. LBVA takes the SPARQL query and returns a concept lattice called as view instead of the results of the SPARQL query. These views created by LBVA are machine as well as human processable. Accordingly, RV-Xplorer (Rdf View eXplorer) exploits the powerful mathematical structure of these concept lattices thus making it interpretable by human. It also allows human agents to interact with the concept lattice and perform navigation. The expert can answer various questions while navigating the concept lattice. RV-Xplorer provides several ways to guide the expert during this navigation process.

3.3.1 Motivating Example

For detailing all the functionalities of the tool effectively, we use the scenario based on scientific publications. Consider a scenario where an expert wants to pose following questions based on articles published in conferences or journals from a team working on data mining. In the current study, we extract the papers published in "Orpailleur Team" in LORIA, Nancy, France. Following are the questions in which an expert may be interested in:

- What are the main research topics in the team and the key researchers w.r.t. these topics, for example, researchers involved in most of the papers in a prominent topic?
- What is the major area of the research of the leader of the team and various key persons?
- Can the diversity of the team leader and key persons be detected?
- Given a paper is it possible to retrieve similar papers published in the team?
- Who are the groups of persons working together?
- What are the research tendencies and possibly the forthcoming and new research topics (for example, single and recent topics which are not in the continuation of the present topics)?

Such kind of questions can not be answered by standard search engines. We want to answer such kind of questions through lattice navigation supported by RV-Xplorer which is built from an initial query and then is explored by the expert according to her preferences.

3.4 The RV-Xplorer

RV-Xplorer (Rdf View eXplorer) is a tool for navigating concept lattices generated by the answers of SPARQL queries over part of RDF graphs using Lattice-Based View Access. Accordingly, this tool provides navigation to the expert over the classification of SPARQL query answers for analyzing the data, finding hidden regularities and answering several questions. On each navigation step it guides the expert in decision making and performing selection to avoid unnecessary selections. It also allows the user to change her point of view while navigating i.e., navigation by extent. Moreover, it also allows the expert to only focus on the specific and interesting part of the concept lattice by allowing her to hide the part of lattice which is not interesting for her.

RV-Xplorer is a web-based tool for building concept lattices. On the client side it uses D3.js which stands for Data-Driven Documents and is based on Javascript for developing interactive data visualizations in modern web browsers. It also uses model-view-controller (MVC) which separates presentation, data and logical components. On the server side we use PHP and MySQL for computing and storing the data. Generally, data can be a graph or pattern generated by pattern mining algorithms etc. Currently, this tool is not publicly available.

Figure 3.2 shows the overall interface of RV-Xplorer (Rdf View eXplorer) which consists of three parts: (1) the middle part is called *local view* which shows detailed description of the selected concept allowing interaction, navigation and level-wise navigation, (2) the left panel is referred to as *Spy* showing the global view of the concept lattice and (3) the lower left is the summarization index for guiding the expert in making decision about which node to choose in the next level by showing the statistics of the next level. For the running scenario, the concept lattice is also available on-line[20].

3.4.1 Local View

Each selected node in the concept lattice is shown in the middle part of the interface displaying complete information. Let c be the selected concept such that $c \in C$ where C is the set of concepts in the complete lattice $\mathcal{L} = (C, \leqslant)$ then a local view shows the complete information about this concept i.e., the extent, intent and the links to the super-concept and the sub-concepts. The set of super and sub-concepts are linked to the selected node where each link represents the partially ordered relation $\leqslant$. By default, the top node is the selected node and is shown in *local view*.

Figure 3.2 (below) shows the selected concept, the orange part defines the label of the selected node which is the entry point for the concept, the pink and yellow parts give the labels of the super-concepts and sub-concepts connected to the selected concept respectively. The green and blue part give the information about the intent and the extent respectively.

3.4.2 Spy

A global view in left panel shows the map of the complete lattice $\mathcal{L} = (C, \leqslant)$ for a particular SPARQL query over an RDF Graph. It tracks the position of the expert in the concept lattice and the path followed by the expert to reach the current concept. It also helps in several navigation tasks such as direct navigation, changing navigation space and navigation between point-of-views. All of these navigation modes are discussed in section 3.5.

[20] `http://rv-xplorer.loria.fr/#/graph/orpailleur_paper/1/`
This tool has been developed with Matthieu Osmuk who was an engineer in LORIA and Amedeo Napoli.

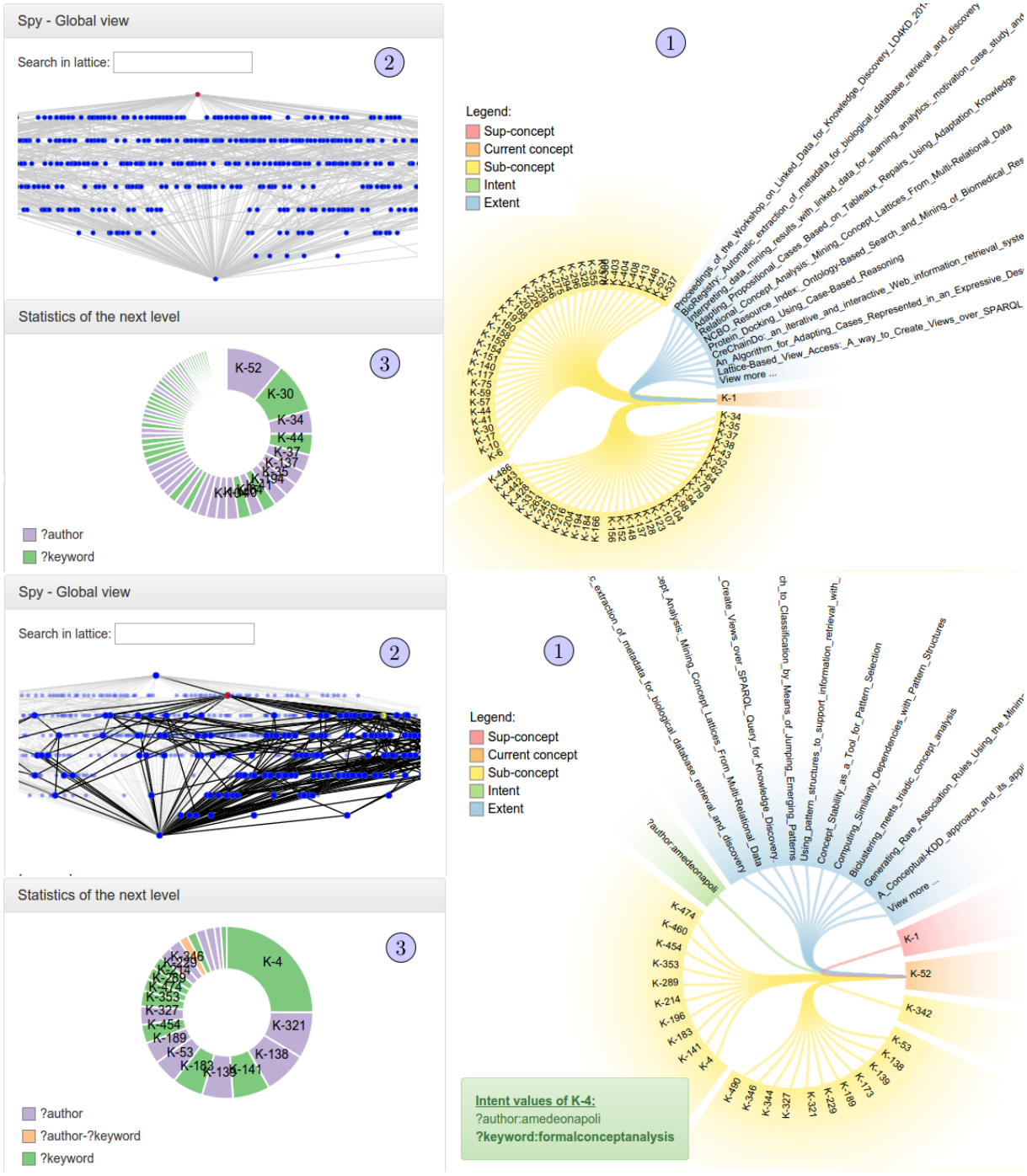


Figure 3.2: Figure above shows the basic interface of RV-Xplorer displaying the top concept. The Figure below shows the local view of $K\#52$, the concept containing all the papers authored by Amedeo Napoli.

3.4.3 Statistics about the next level

The statistics about the next level are computed with the help of a summarization index which depicts the information about the distribution of the objects in the extent of the selected concept in the linked sub-concepts i.e., concepts in the next level of the concept lattice. Let c_i be a concept in the next level where $i \in \{1, \ldots, n\}$ and n is the number of concepts in the next level. $ext(c_i)$ is the extent of the concept then $|ext(c_i)|$ is the size of the extent. Finally, the statistics about the next level are computed with the help of summarization index.

$$summarization\ index = \frac{|ext(c_i)|}{\sum_{j=\{1,\ldots,n\}} |ext(c_j)|} \times 100 \tag{3.1}$$

Here, $\sum_{j=\{1,\ldots,n\}} |ext(c_j)|$ is the sum of extent size of all the concepts in the next level. The sum of summarization index for all the sub-concept adds to 100%. In Figure 3.2, the percentages are represented in the form of a pie-chart which shows the distribution. The sub-concept containing the most elements in the extent has the highest percentage and hence has the biggest part in the pie chart.

3.5 Navigation Operations

In this section we detail some of the classical Carpineto and Romano (1996b) as well as advanced navigation operations that are implemented in RV-Xplorer. Navigation can be done locally with a parallel operation which is shown globally through local and global views. Navigation operations allow the expert to locate particular pieces of information which helps in obtaining several answers of the expert questions as well as analysis of the data at hand. Initially, the selected concept is the top concept which contains all the objects.

3.5.1 Guided Downward (Drill down)/ Upward Navigation (Roll-up):

The local view provides expert with the drilling down operation which is achieved by selecting the sub-concepts given in yellow part of local view. RV-Xplorer guides the expert in drilling down the concept lattice by showing contents of the sub-concept to the expert before selecting the node on mouse over. Another added guidance provided to the expert is with the help of the summarization index which gives the statistics about the next level. This way the expert can avoid the attributes or the navigation path which may lead to uninteresting results. The local view also allows the expert to roll-up from the specific concept to the general concept. A super-concept can be selected following the link given in the view.

Consider the running scenario discussed in section 3.3.1 where the expert wants to know *who are researchers having main influences in the team?* by analyzing the publications of this particular team. Initially, the selected concept in the local view is the top concept (see Figure 3.2 (above)). Now it can be seen from the summarization index that most of the papers are contained in $K\#52$. On mouse over on $K\#52$ it shows that this concept keeps all the papers published by Amedeo Napoli. From here it can be safely concluded that Amedeo Napoli is the leader of the team. Similarly, several key team members can be identified on the same level such as supervisors etc. If the expert wants to view the papers published by Amedeo Napoli, a downward navigation is performed by selecting concept $K\#52$. With the help of the summarization index another question can be answered i.e., *what are the main research topics of these researchers?*. Again by consulting the index it can be seen that $K\#4$ keeps the largest percentage of papers published by Amedeo Napoli (see Figure 3.2 (below)) and the keyword in this concept is Formal

Concept Analysis meaning that the main area of research of Amedeo Napoli is Formal Concept Analysis. However, there are many other areas of research on which he has worked, which shows the diversity of authors based on the area of research he has published in. Moreover, the sub-lattice connected to this concept keeps information about the community of authors with who she publishes the most and about which topic and what variants of formal concept analysis. Now, if the expert wants to retrieve all the papers published by Amedeo Napoli then she can go back to $K\#52$.

3.5.2 Direct Navigation

The spy on the left part of the RV-Xplorer (see Figure 3.2) allows the expert for direct navigation. If an expert has navigated too deep in the view while performing multiple drill-down operations then the spy, which keeps track of the current position of the expert, shows all the paths from the selected concept to the top concept and allows the expert to directly jump from one concept to another linked concept without performing level-wise navigation. Unlike drill-down and roll-up, direct navigation allows the expert to skip two or more hops and select the more general or specific concept.

These three navigation modes are very common and are repeatedly discussed in many of the navigational tools built for concept lattice such as Camelis Ferré (2009) and CREDO Carpineto and Romano (2004b) which may or not may not be for a specific purpose. The main difference between RV-Xplorer and the two approaches and most of the navigational tools is that they use folder-tree display. As a contrast we manage to keep the original structure of a concept lattice. An added advantage of RV-Xplorer is that these navigation modes are guided at each step meaning that the interface shows the expert with what is contained in the next node as well as the statistics about the next level. This way the interface guides the expert in choosing the nodes interesting for her by reducing the chance of performing unnecessary navigation and backtracking to see the details unnecessarily.

3.5.3 Navigating Across Point-of-Views

The current interface allows the expert to toggle between points-of-view, i.e., at any point an expert can start exploring the lattice with respect to the objects (extent) in the concept lattice. Let c be the selected concept and the expert is interested in $g_1 \in ext(c)$ where $ext(c)$ is the extent of the selected concept. Then if the expert hovers her mouse over this extent in the local view, the Spy highlights all the concepts where this object is present along with the object concept of g_1 which is highlighted in red.

For instance, the selected concept contains keyword *data dependencies* in the intent and she is interested in the paper *Computing Similarity Dependencies with Pattern Structures* and she wants to retrieve all the related or similar papers then on mouse hover it highlights all the concepts containing this paper. Then she selects the concept highlighted in red i.e., the object concept of this paper. The right side of Figure 3.3 shows the highlighted object concept of *Computing Similarity Dependencies with Pattern Structures* in RV-Xplorer. After this concept is selected. The spy highlights all the paths from this concept until bottom and the top which actually is the sub-lattice associated to this paper. All the objects contained in the extent of the concepts in this sub-lattice are similar to the paper at hand i.e., papers sharing some properties with the paper *Computing Similarity Dependencies with Pattern Structures*.

If we consider the folder-tree display as discussed in most of the navigational tools such as CREDO Carpineto and Romano (2004b) and CEM Cole and Stumme (2000), such kind of

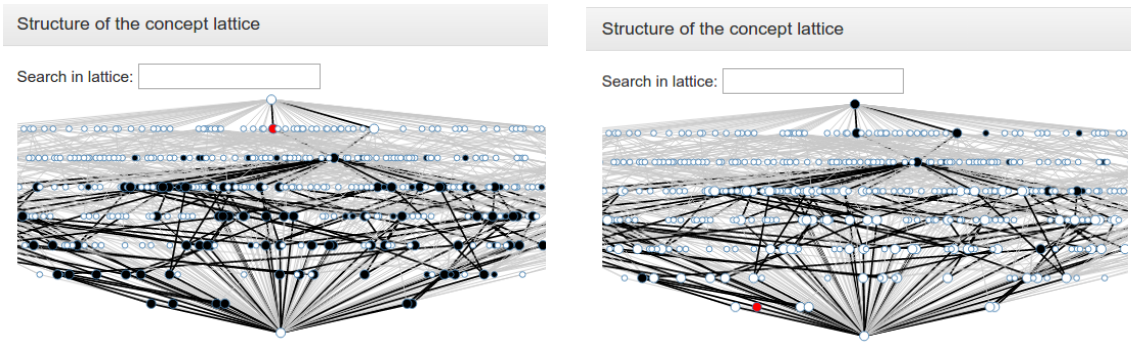


Figure 3.3: Left Figure shows the Attribute Concept of FCA and Right Figure shows the Object Concept of *Computing Similarity Dependencies with Pattern Structures*.

navigation is not possible because it only allows navigation w.r.t. intent and extent is considered as the answers of the navigations. In case of RV-Xplorer, it is possible to obtain the sub-lattice related to a certain interesting object and this way the whole sub-lattice connected to the object concept of the object of interest can be navigated to retrieve similar objects i.e., sharing at least one attribute with the object of interest.

3.5.4 Altering Navigation Space

The navigation space can be changed when the selected concept is deep-down in the concept lattice without the effort to start the navigation all over again from the top concept. Let c be the selected concept such that m_1 *and* $m_2 \in int(c)$ (int(c) is the intent of the selected concept) and the expert has navigated downwards from the concept whose intent only contains m_1. Now the expert wants to navigate the lattice w.r.t. m_2, on mouse hover the interface highlights all the concepts where the given attribute exists and further highlights the attribute concept in red. The attribute concept of m_2 can be selected. In the running example, if the expert has navigated the lattice w.r.t. the author Amedeo Napoli and she finds some papers on FCA authored by Amedeo Napoli. Now she wants to navigate the concept lattice w.r.t. the keyword *FCA* then she can easily locate the attribute concept of the keyword FCA and navigate to get specific information. The left side of Figure 3.3 shows the highlighted attribute concept of FCA in RV-Xplorer.

In tree-folder display altering navigation space w.r.t. intent needs the expert to locate the attribute concept by herself by manually checking each of the branches because it represents the concept lattice as a tree. The problem with such a display is that it is not easy to alter the browsing space quickly or change the navigation point of view. Moreover, the sub-lattice connected to a selected concept can not be seen because of the restrictions posed by tree display.

3.5.5 Area Expansion

Area expansion allows the expert to select several concepts at one time scattered over the concept lattice and gives the overall view of what these concepts contains. These concepts are not necessarily a part of navigation path that the expert is following. It allows the expert to have an overall view of other concepts without starting the navigation process again.

This idea was first put-forth in Messai et al. (2010), where they allow the expert to move from one concept lattice to another concept lattice based on the granularity level w.r.t. a taxonomy and a similarity threshold. The concepts in the concept lattice with higher threshold contains

more detailed information as compared to the concept lattice built using lesser threshold. One drawback of such kind of zooming operation is that it requires the computation of several concept lattices. In case of RV-Xplorer, we are dealing with simple concept lattice instead of the one created after using hierarchies meaning that all such kind of information needs to be scaled to obtain a binary context. As we are dealing with concept lattices built from binary contexts, we bend this functionality to suit the needs. It does not require computation of many concept lattices as well as no re-computation is required.

3.5.6 Hiding Non-Interesting Parts of the View

One of the most interesting characteristic of RV-Xplorer is that it allows the expert to hide the non-interesting part of the lattice. Let us consider that expert selects a concept c and it contains an attribute which is not interesting for her. She can at any point right click on the concept and select hide sub-lattice. One of the most interesting characteristic of a concept lattice is that if one concept contains some attribute in an intent then all the sub-concepts inherit this attribute. This way if the expert considers one concept as un-interesting then the whole sub-lattice will be considered as uninteresting and hence will be hidden from the expert while navigation. Such kind of functionality enables expert to reduce her navigational space and at the end the concept lattice contains only those concepts which are interesting for the expert.

Similar functionality was first introduced in CreChainDo system Nauer and Toussaint (2007). Similar to CREDO Carpineto and Romano (2004b), CreChainDo allows the expert to pose a query against the standard search engine which returns some results. These results are then organized in the form of a concept lattice and displayed to the expert in the form of folder-tree display. An added advantage of CreChainDo over CREDO is that the former allows expert interaction i.e., the expert can mark the concepts as relevant or irrelevant based on her priorities. After the expert has marked the concept irrelevant the sub-lattice linked to that concept is deleted. Meaning that, it reduces the context based on this feedback and the concept lattice is computed again using the reduced context. In case of RV-Xplorer, the concept lattice is built on top of RDF graphs. Moreover, we do not recompute the lattice or remove anything from the concept lattice. We only hide the non-interesting part of the lattice to reduce the navigation space of the expert. This way a reduction in the navigation space is performed without re-computing a concept lattice.

3.5.7 Other Functionalities

RV-Xplorer also allows the user to set the *support* and *stability* thresholds to reduce the navigation space of the user. Moreover, RV-Xplorer also displays *implications* generated directly from the concept lattice. It generates a complete basis of implications as well as it also displays level-wise implications for knowledge discovery purposes.

3.6 Experimentation

The experiments were conducted on real dataset. Our algorithm is implemented in Java using Jena[21] platform and the experiments were conducted on a laptop with 2.60 GHz Intel core i5 processor, 3.7 GB RAM running Ubuntu 12.04. We extracted the information about the movie with their genre and location using SPARQL query enhanced with `VIEW BY` clause. The

[21] `https://jena.apache.org/`

experiment shows that even though the background knowledge (ontological information) was not extracted the views reveal the hidden hierarchical information contained in the SPARQL query answers and can be navigated accordingly. Moreover, it also shows that useful knowledge is extracted from the answers through the the views using $\mathcal{DG}-$Basis of implications. We also performed quantitative analysis where we discussed about the sparsity of the semantic web data. We also tested how our method scales with growing number of results. The number of answers obtained by YAGO were 100,000. The resulting view kept the classes of movies with respect to genre and location.

3.6.1 YAGO

The construction of YAGO ontology Suchanek et al. (2007) is based on the extraction of instances and hierarchical information from Wikipedia and Wordnet. In the current experiment, we sent a query to YAGO with the `VIEW BY` clause.

```
PREFIX rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX yago:  http://yago-knowledge.org/resource/
SELECT ?movie ?genre ?location WHERE {
    ?movie rdf:type yago:wordnet_movie_106613686 .
    ?movie yago:isLocatedIn ?location .
    ?movie rdf:type ?genre .  }
VIEW BY ?movie
```

While querying YAGO it was observed that the genre and location information was also given in the ontology. The first level of the obtained view over the SPARQL query results over YAGO kept the groups of movies with respect to their languages. e.g., the movies with genre `Spanish Language Films`. However, as we further drill down in the concept lattice we get more specific categories which include the values from the location variable such as `Spain, Argentina` and `Mexico`. There were separate classes obtained for movies based on novels which were then further specialized by the introduction of the country attribute as we drill down the concept lattice. Finally with the help of lattice-based views, it can be concluded that the answers obtained by querying YAGO provides a clean categorization of movies by making use of the partially ordered relation between the concepts present in the concept lattice.

3.6.2 DBpedia

DBpedia is currently comprised of a huge amount of RDF triples in many different languages. It reflects the state of Wikipedia Bizer et al. (2009b); Lehmann et al. (2014). Due to information extraction from crowd-sourced web site, triples present on DBpedia may contain incorrect information. Even if Wikipedia contains correct information, a parser may pick up wrong information Wienand and Paulheim (2014a). Due to the above described reasons some of the properties may not be used uniformly. In the current experiment, we extracted the information about movies with their genre and location. It was also observed that most of the movies had a missing type (`rdf:type`) i.e., dbpedia-owl:Film and were not retrieved by the SPARQL query.

```
SELECT ?movie ?genre ?country WHERE {
?movie rdf:type dbpedia-owl:Film .
```

ID	Supp.	Intent
C#1	17	`Hard Rock`
C#2	15	`Contemporary R&B`
C#3	18	`Jazz`
C#4	732	`United States en`
C#5	2	`United States`
C#6	16	`USA en`
C#7	1225	`India en`
C#8	6	`France`

Table 3.6: Some Concepts from $\mathcal{L}_D Bpedia$

```
?movie dbpprop:genre ?genre .
?movie dbpprop:country ?country .}
VIEW BY ?movie
```

The obtained concept lattice contained 1395 concepts. Out of which 201 concepts on the first level were evaluated manually for correctness of the information about the movie genre. 60 concepts contained the distinct classes related to the country of the movies. The other 141 concepts kept the genre information about the movie. Out of these 141 concepts 45% of the concepts contained wrong genre information as its intent. In Table 3.6, first three concepts contain wrong information about the music genre. In such a case, the generated lattice-based views helps in separating music genre from the movie genre and further guide in introducing a new relation such as `soundtrackGenre` and adding new triples to the knowledge base, for example, $dbpedia : The\_Scorpion\_King, dbpedia - owl : soundtrackGenre, dbpedia : Hard\_Rock$.

Moreover, If we observe the obtained view, it can be seen that there are too few movies from countries other than United States and India. For example, C#4,C#5,C#6 and C#7 are the classes for movies from United States and India, where there are 1225 movies from India in DBpedia and 750 movies from United States. The movies from France are very few, meaning that there are some incompletions present in DBpedia. For example, there are some of the movies which are from France should have `rdf:type Film` and the property `hasCountry France`. Unfortunately, many of the movies are missing such kind of information. Same is the case with the data belonging to domains other than Film. We will target this issue in detail in next chapter.

In C#5 and C#6, the *en* suffix represents a literal for the country United States and in C#4 the intent united States represents a URI, which is a widely found error in DBpedia. Which means that such classes can be merged into one by taking the union of the extent of C#4,C#5 and C#6. Finally, it can be concluded that the information present on DBpedia still needs to be corrected and completed.

The concept lattice can help in obtaining classes of movies w.r.t countries also. As this approach provides an added value to the already existing `Group By` clause, it is possible to find movies which are made in collaboration with several countries. For example, `The Scorpion King` was made in collaboration with `United States, Germany` and `Belgium`. However, one of the problems encountered for obtaining such results is that the support and stability of such concepts is very low.

$\mathcal{DG}$-Basis of Implications: $\mathcal{DG}$-Basis of Implications for YAGO were calculated. The implications were filtered in three ways. Firstly, pruning was performed naively with respect to

Impl. ID	Supp.	Implication
YAGO		
1.	96	`wikicategory RKO Pictures films → United States`
2.	46	`wikicategory Oriya language films → India`
3.	64	`wikicategory Film remakes → wordnet remake`
DBpedia		
4.	3	`Historical fiction → United Kingdom@en`
5.	3	`Adventure fiction, Action fiction → Science fiction`

Table 3.7: Some implications from $\mathcal{DG}$-Basis of Implication (YAGO)

support threshold. For DBpedia, the number of rules obtained were 64 for a support threshold of 11%. Around 200 rules were extracted on support threshold of 2%. In order, to make the rules observable, the second type of filtering based on number of elements in the body of the rules was applied. All the implications which contained one item set in the body were selected. However, if there still are large number of implications to be observed then a third type of pruning can be applied which involved the selection of implications with different attribute type in head and body, e.g., in rule#1 head contains United States which is of type country and body contains the wikicategory. Such kind of pruning helps in finding attribute-attribute relations.

Table 3.7 contains some of the implications. Calculating $\mathcal{DG}-Basis$ of implications is actually useful in finding regularities in the SPARQL query answers which can not be discovered from the raw tuples obtained. For example, rule#1 states that `RKO picture films` is an American film production and distribution company as all the movies produced and distributed by them are from United States. Moreover, rule#2 says that all the movies in `Oriya language` are from `India`. This actually points to the fact that Oriya is one of many languages that is spoken in India. This rule also tells that Oriya language is only spoken in India. Rule#3 shows a link between a category from Wikipedia and Wordnet, which clearly says that the `wikicategory` is more specific than the `wordnet` category as `remake` is more general than `Film remakes`. On the other hand the rules obtained from DBpedia may be a little bit vague. For example, rule#4 states the strange fact that all the `historical fiction` movies are from `United Kingdom`. Same is the case with rule#3 which states that all the movies which are `Adventure fiction` and `Action fiction` are also `Science Fiction`, which may not actually be the case.

3.6.3 Evaluation

Besides the qualitative evaluation of LBVA, we performed an empirical evaluation. The characteristics of the datasets DBpedia and YAGO are shown in Table 3.8 and Table 3.9. These concepts were pruned with the help of iceberg lattices and stability for qualitative analysis.

The plots for the experimentation are shown in Figure 3.4. Figure 3.4(a) and 3.4(c) show a comparison between the number of tuples obtained and the density of the formal context. The density of the formal context is the proportion of pairs in I w.r.t the size $G \times M$. It has very low range for both the experiments, i.e., it ranges from 0.14% to 0.28% and from 0.33% to 0.95% for DBpedia. This means in particular that the semantic web data is very sparse when considered in a formal context and deviates from the datasets usually considered for FCA (as they are dense). Here we can see that as the number of tuples increases the density of the formal context is decreasing which means that sparsity of the data also increases.

We also tested how our method scales with growing number of results. The number of answers

No. of Tuples	$\|G\|$	$\|M\|$	No. of Concepts
20%	530	200	291
40%	1091	326	550
60%	1574	411	856
80%	2037	495	1149
100%	2619	584	1395

Table 3.8: Characteristics of Datasets (DBpedia)

No. of Tuples	$\|G\|$	$\|M\|$	No. of Concepts
20%	3657	2198	7885
40%	6783	3328	19019
60%	9830	4012	31264
80%	12960	4533	43510
100%	15272	4895	55357

Table 3.9: Characteristics of Datasets (YAGO)

for SPARQL query over DBpedia were 4000 (this is the limit posed by the SPARQL endpoint for DBpedia) and for YAGO were 100,000 (this is the limit posed by the SPARQL endpoint for YAGO). Figure 3.4(b) and 3.4(d) illustrate the execution time for building the concept lattice w.r.t the number of tuples obtained. The execution time ranges from 20 to 100 seconds for YAGO and 0 to 0.2 seconds for DBpedia, it means that the the concept lattices were built in an efficient way and large data can be considered for these kinds of experiments. Usually the computation time for building concept lattices depends on the density of the formal context but in the case of semantic web data, as the density is not more than 1%, the computation completely depends on the number of objects obtained which definitely increases with the increase in the number of tuples (see Table 3.8 and 3.9).

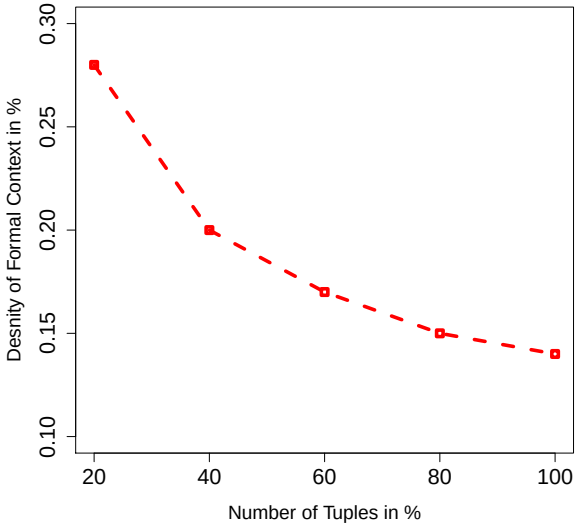
3.6.4 Application to Biomedical Data

This experiment is performed to evaluate that LBVA is an application independent framework i.e., it can be applied to any dataset. It also discusses how the view and the $\mathcal{DG}$-Basis of implications can be used for accessing useful knowledge. It also gives an overall idea of how LBVA can be used to extract knowledge from biomedical datasets present in the LOD cloud.

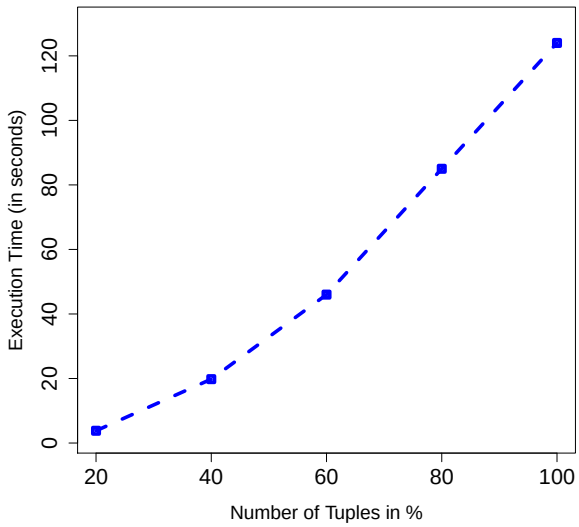
It considers a SPARQL query with four variables. The query is run on *Sider Database*[22] for extracting drugs with their side effects and on *Drug Bank*[23] for extracting drugs with their categories and the proteins they target for the largest drug set *Cardiovascular Agents (CVA)*. In this scenario, one objective is to check the validity of prescriptions drug-diseases and second objective is to check the side effects of some drugs. Following is the query:

```
SELECT ?drugname ?sideeffect ?protein ?category
WHERE {
   ?drug rdf:type drugbank:drugs .
   ?drug rdfs:label ?drug_name .
```

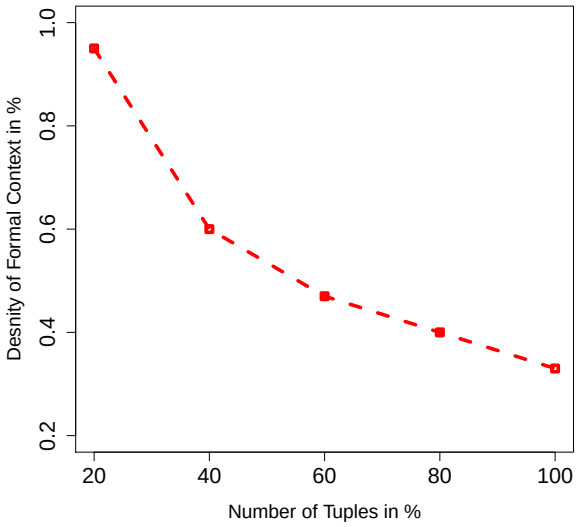
[22] http://sideeffects.embl.de/
[23] http://www.drugbank.ca/



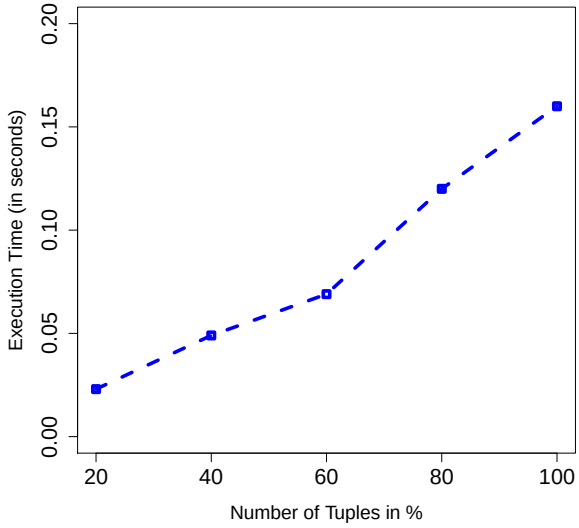
(a) Density of $\mathcal{K}_{YAGO}$



(b) Runtime for Building $\mathcal{L}_{YAGO}$



(c) Density of $\mathcal{K}_{DBpedia}$



(d) Runtime for Building $\mathcal{L}_{DBpedia}$

Figure 3.4: Experimental Results.

ID	Supp.	Intent
C#1	39	`se:Jaundice`
C#2	24	`se:Infection, se:Shock`
C#3	13	`p08588`
C#4	12	`p07550, p08588`
C#5	15	`se:Acute coronary syndrome, cat:Antihypertensive Agents`
C#6	12	`se:Tachycardia, cat:Anti-Arrhythmia Agents`

Table 3.10: Some Concepts from the Iceberg Lattice

ID	Supp.	Implication
1.	22	`se:Stevens-Johnson syndrome` → `se:Erythema multiforme`
2.	8	`se:Acute coronary syndrome, se:Arthralgia, se:Infection, se:Pyrexia, se:Tachycardia` → `se:Cerebrovascular accident`
3.	7	`cat:Vasoconstrictor Agents` → `se:Acute coronary syndrome`
4.	8	`p43700` → `se:Vision blurred`

Table 3.11: Some implications from $\mathcal{DG}$-Basis of Implication

```
?drug drugbank:sideEffect ?sideeffect .
?drug drugbank:hasCategory ?category .
?drug drugbank:target ?protein .
filter regex(?category, 'Cardiovascular Agents') }
VIEW BY ?drug
```

The obtained result set contain 4-tuples, *i.e., drug name, side effect, protein id (UniProt ID) and category*. In this experiment the total number of tuples obtained is `4843`. with `89` drugs belonging to `10` distinct sub categories of `CVA`. These drugs target `161 proteins` and have `72 distinct side effects`. The objects count in the formal context is `89` and the attribute count is `243`. The concept lattice includes some interesting concepts where sets of drugs constitute the extent while combination of side effects constitute the intents.

In the following, we discuss the possible interpretation of some concepts in collaboration with some domain experts.

Navigation and Information Retrieval: For this an Iceberg lattice with the support threshold of `12%` gives 360 concepts (the maximum support is `46`). Some of the concepts are shown in Table 3.10. The prefixes `se:` for side effect, `cat:` for category help in differentiating categories from side effects while interpreting. The maximum support `39` was for side effect `Jaundice` (C#1). This confirms that many drugs which are `CVA` cause `Jaundice`. The concept lattice generated also contain several interesting combinations of side effects caused by groups of drugs. For example, `Infection` along with `Shock` are caused by 24 drugs. *C#3* provides the following explanation: having some abnormality (mutation) in in the protein `Beta-1 Adrenergic Receptor` forbids the use of these 13 drugs. Moreover, 12 drugs target two proteins `(p07550,p08588)` (C#4). A subgroup of `CVA` (C#5) used for hypertension cause `Acute Coronary Syndrome`. The most eye catching result concern 12 drugs (in C#6) which are used for the treatment of `Arrhythmia` (irregular heart beat) and cause the same kind of side effect (as its indication) `Tachycardia`.

$\mathcal{DG}$-Basis of Implication: Some of the rules obtained in $\mathcal{DG}$-Basis of implications for the current formal context are shown in Table 3.11. The first rule states that all the drugs which have side effect `Stevens-johnson Syndrome` also have side effect `Erythema multiforme` (support of 22). This explains that if these drugs are prescribed and they cause `Stevens-johnson Syndrome` then a patient is most likely to have `Erythema multiforme`. Rule#2 depicts that all the drugs which have side effects `Acute coronary syndrome, Arthralgia, Infection, Pyrexia and Tachycardia` have the side effect `Cerebrovascular accident`. `Cerebrovascular accident` is the medical name for stroke. Thus if the drugs are causing the above mentioned side effects then it is most likely that a patient may have this side effect also. Similarly, rule#3 says that all the drugs which are used for the treatment of `Vasoconstriction` (narrowing blood vessels) have the side effect `Acute Coronary Syndrome`. Finally, rule#4 mentions that when the drugs target protein `p43700` they cause `blurred vision`.

All these results show that a careful interpretation of some concepts in the lattice may provide very useful explanation on some observations. These explanations can be reused by human agents and software agents as well.

3.7 Discussion

In this chapter, we introduce a classification framework based on FCA for the set of tuples obtained as a result of SPARQL queries over LOD. In this way, a view is organized as a concept lattice built through the use of `VIEW BY` clause that can be navigated where information retrieval and knowledge discovery can be performed. Several experiments show that LBVA is rather tractable and can be applied to large data. We also introduced a new navigational tool for concept lattices called as RV-Xplorer which provides exploration over SPARQL query answers. With the help of guided navigation implemented in RV-Xplorer we were able to answer all the questions posed initially in the scenario. However, this tool is not designed for only specific purpose any kind of concept lattice can be visualized and data from any domain can be analyzed using this tool. Finally, several experiments were discussed performed on several domains. One of the experiments reveal the incompletion in the RDF data more specifically DBpedia.

Most of the resources containing RDF data are generated automatically or are obtained by converting the crowd-sourced web resource into RDF such as Wikipedia. Due to these reasons these data may have some missing information which are revealed while navigating and exploring RDF data. In the next chapter we discuss how we can use association rule mining to complete this missing information by reasoning.

Chapter 4

Mining definitions from RDF annotations using Formal Concept Analysis

Contents

The popularization and quick growth of Linked Open Data (LOD) has led to challenging aspects regarding quality assessment and data exploration of the RDF triples that shape the LOD cloud. Particularly, we are interested in the completeness of data and its potential to provide concept definitions in terms of necessary and sufficient conditions. In this work we propose a novel technique based on Formal Concept Analysis which organizes RDF data into a concept lattice. This allows data exploration as well as the discovery of implications, which are used to automatically detect missing information and then to complete RDF data. Moreover, this is a way of reconciling syntax and semantics in the LOD cloud. Finally, experiments on the DBpedia knowledge base show that the approach is well-founded and effective. This work has been done in collaboration with Aleksey Buzmakov, Victor Codocedo and Amedeo Napoli Alam et al. (2015b,a).

4.1 Introduction

World Wide Web has tried to overcome the barrier of data sharing by converging data publication into Linked Open Data (LOD) Bizer et al. (2009a). The LOD cloud stores data in the form of *subject-predicate-object* triples based on the RDF language[24], a standard formalism for information description of web resources. In this context, DBpedia is the largest reservoir of linked data in the world currently containing more than 4 billion triples. All of the information stored in DBpedia is obtained by parsing Wikipedia, the largest open Encyclopedia created by the collaborative effort of thousands of people with different levels of knowledge in several and diverse domains.

More specifically, DBpedia content is obtained from semi-structured sources of information in Wikipedia, namely *infoboxes* and *categories*. Infoboxes are used to standardize entries of a given type in Wikipedia. For example, the infobox for "automobile" has entries for an image depicting the car, the name of the car, the manufacturer, the engine, etc. These *attributes* are mapped by the DBpedia parser to a set of "properties" defined in an emerging ontology[25] Benz et al. (2010) (infobox dataset) or mapped through a hand-crafted lookup table to what is called the DBPedia Ontology (mapped-based ontology). Categories are another important tool in Wikipedia used to organize information. Users can freely assign a category name to an article relating it to other articles in the same category. Example of categories for cars are "Category:2010s automobiles", "Category:Sports cars" or "Category:Flagship vehicles". While we can see categories in Wikipedia as an emerging "folksonomy", the fact that they are curated and "edited" make them closer to a controlled vocabulary. DBpedia exploits the Wikipedia category system to "annotate"[26] objects using a taxonomy-like notation. Thus, it is possible to query DBpedia by using *annotations* (e.g. all cars annotated as "Sport cars"). While categorical information in DBpedia is very valuable, it is not possible to use a category as one could expect, i.e. as a definition of a class of elements that are instances of the class or, alternatively, that are "described" by the category. In this sense, such a category violates the actual spirit of semantic Web.

Let us explain this with an example. The Web site of DBpedia in its section of "Online access" contains some query examples using the SPARQL query language. The first query has the description "People who were born in Berlin before 1900" which actually translates into a graph-based search of entities of the type "Person", which have the property "birthPlace" pointing to the entity representing the "city of Berlin" and another property named "birthDate" with a value less than 1900. We can see here linked data working at "its purest", i.e. the form of the query provides the right-hand side of a definition for "People who were born in Berlin before 1900". Nevertheless, the fourth query named "French films" does not work in the same way. While we could expect also a graph-based search of objects of the type "Film" with maybe a property called "hasCountry" pointing to the entity representing "France", we have a much rougher approach. The actual SPARQL query asks for objects (of any type) annotated as "French films".

In general, categorization systems express "information needs" allowing human entities to quickly access data. French films are annotated as such because there is a need to find them by these keywords. However, for a machine agent this information need is better expressed through a *definition*, like that provided for the first query (i.e. "People who were born in Berlin before 1900"). Currently, DBPedia mixes these two paradigms of data access in an effort to profit from the structured nature of categories, nevertheless further steps have to be developed to ensure

[24] Resource Description Framework - `http://www.w3.org/RDF/`

[25] Emerging in the sense of "dynamic" or "in progress".

[26] Notice that in DBPedia the property used to link entities and categories is called "subject". We use "annotation" instead of "subject" to avoid confusions with the "subject" in an RDF triple.

coherence and completeness in data.

Accordingly, in this work we describe an approach to bridge the gap between the current syntactic nature of categorical annotations with their semantic correspondent in the form of a concept definition. We achieve this by mining patterns derived from entities annotated by a given category, e.g. All entities annotated as "Lamborghini cars" are of "type automobile" and "manufactured by Lamborghini", or all entities annotated as "French films" are of "type film" and of "French nationality". We describe how these category-pattern equivalences can be described as "definitions" according to *implication rules* among attributes which can be mined using Formal Concept Analysis (FCA Ganter and Wille (1999)). The method considers the analysis of heterogeneous complex data (not necessarily binary data) through the use of "pattern structures" Ganter and Kuznetsov (2001b), which is an extension of FCA able to process complex data descriptions. A concept lattice can be built from the data and then used for discovering *implication rules* (i.e. association rules whose confidence is 100%) which provide a basis for "subject definition" in terms of necessary and sufficient conditions. An RDF triple is represented as $U \times U \times (U \cup L)$. For convenience, in the following we denote the set of predicate names as P and the set of object names as O. For example, let us consider the SPARQL query given in Listing 4.1, for all the entities of type Automobile manufactured by *Lamborghini*, annotated as "Sport_cars" and as "Lamborghini_vehicles",

```
SELECT ?s WHERE {
?s dc:subject dbpc:Sports_cars .
?s dc:subject dbpc:Lamborghini_vehicles .
?s rdf:type dbo:Automobile .
?s dbo:manufacturer dbp:Lamborghini }
```

Listing 4.1: SPARQL for the formal context in Figure 4.1. Prefixes are defined in Table 4.1.

Consider the formal context in Figure 4.1 where $G = U$[27], $M = (P \times O)$ and $(u, (p, o)) \in I \iff \langle u, p, o \rangle \in \mathcal{G}$, i.e. $\langle u, p, o \rangle$ is a triple built from different triples manually extracted from DBpedia about nine different Lamborghini cars (35 RDF triples in total). Given a subject-predicate-object triple, the formal context contains subjects in rows, the pairs predicate-object in columns and a cross in the cell where the triple subject in row and predicate-object in column exists. Figure 4.1 depicts the concept lattice in reduced notation calculated for this formal context and contains 12 formal concepts. Consider the first five cars (*subjects*) in the table for

[27] In this chapter we call G "entities" to avoid confusions with "objects" as defined for RDF triples.

Predicates		Objects	
Index	URI	Index	URI
A	dc:subject	a	dbpc:Sport_Cars
		b	dbpc:Lamborghini_vehicles
B	dbp:manufacturer	c	dbp:Lamborghini
C	rdf:type	d	dbo:Automobile
D	dbp:assembly	e	dbp:Italy
E	dbo:layout	f	dbp:Four-wheel_drive
		g	dbp:Front-engine

Namespaces:	
dc:	http://purl.org/dc/terms/
dbo:	http://dbpedia.org/ontology/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
dbp:	http://dbpedia.org/resource/
dbpc:	http://dbpedia.org/resource/Category:

Table 4.1: Index of pairs predicate-object and namespaces.

	A		B	C	D		E
	a	b	c	d	e	f	g
Reventon	×	×	×	×	×	×	
Countach	×	×	×	×	×		
350GT	×	×	×	×			×
400GT	×	×	×	×			
Islero	×	×	×	×			
Veneno	×	×					
Aventador Roadster	×	×					
Estoque			×	×		×	×
Gallardo			×	×	×		

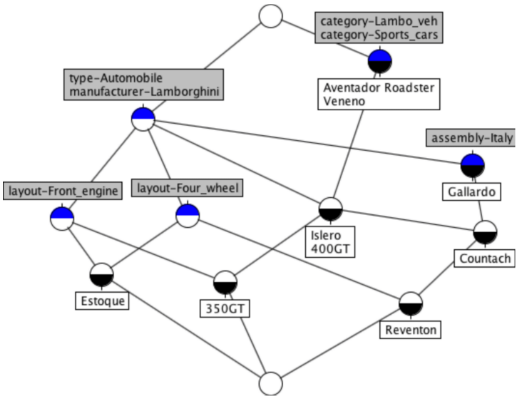


Figure 4.1: The formal context shown on the left is built after scaling from DBpedia data given in Table 4.1. Each cross (×) corresponds to a triple subject-predicate-object. On the right the corresponding concept lattice is shown.

which the maximal set of attributes they share is given by the first four *predicate-object* pairs. Actually, they form a formal concept depicted by the gray cells in Figure 4.1 and labelled as "Islero, 400GT" in Figure 4.1 (actually, the extent of this concept is "Islero, 400GT, 350GT, Reventon").

Given a concept lattice, rules can be extracted from the intents of concepts which are comparable. For instance, consider the association rule $e \rightarrow a, b, c, d$. Its support is given by $|\{a, b, c, d\}' \cap \{e\}'|$ which is the same as $|\{Reventon, Countach\}| = 2$. Since $|\{e\}'| = 3$, we have that the confidence of this association rule is $2/3 \approx 0.67$. A rule $X \Longrightarrow Y$ of confidence 1 (when $|X' \cap Y'| = |X'|$ or $X' \subseteq Y'$) is called an *implication*. When $X \Longrightarrow Y$ and $Y \Longrightarrow X$ are implications, we say that $X \Longleftrightarrow Y$ is an equivalence or a *definition*. This can happen when two attributes have the same "attribute concept", e.g. *type-Automobile* and *manufacturer-Lamborghini* in the concept lattice of Figure 4.1.

4.2 Improving DBpedia with FCA

4.2.1 Problem context

Consider the following fictional scenario. You are a bookkeeper in a library of books written in a language you do not understand. A customer arrives and asks you for a book about "Cars". Since you do not know what the books are about (because you cannot read them), you ask the customer to browse the collection on his own. After he finds a book he is interested to read, you will mark the symbol ⋆ on that book for future references. Then, in an empty page you will write (⋆ - *Cars*). After several cases like this, you will probably end up with a page full of symbols representing different topics or categories of your books, among them (⊖ - *Sports*), (⋄ - *Football*) and (○ - *History*). Now you can even combine symbols when customers ask you for "Sport Cars" which you translate into ⋆⊖. Actually, the demand for books about "Sport Cars" is so high that you create a new symbol † just for it. So doing, you have created your own categorization system of a collection of books you do not understand.

In general, given a topic, you are able to retrieve books without many troubles, however since you do not understand the books, you are restricted to the set of symbols you have for doing this. Furthermore, if you are not careful some problems start to arise, such as books marked with ⋄

Rule	Confidence	Support	Meaning
d $\Longrightarrow$ c	100%	7	Every automobile is manufactured by Lamborghini.
c $\Longrightarrow$ d	100%	7	Everything manufactured by Lamborghini is an automobile.
e $\Longrightarrow$ b,c	100%	3	All the entities assembled in Italy are Lamborghini automobiles.
c,d $\Longrightarrow$ a,b	71%	7	71% of the Lamborghini automobiles are categorized as "sport cars" and "Lamborghini vehicles"

Table 4.2: Association rules extracted from formal context in Figure 4.1.

and without $\ominus$. Finally, people do not get books marked with † when they look for "Cars", since they only search for the symbol $\ominus$.

It is easy to stablish an analogy on how DBpedia profits from Wikipedia's categorization system and the above scenario. DBpedia is able to retrieve entities when queried with an annotation (as the example of "French films"), however any information need not initially provided as a category is unavailable for retrieval (such as "French films about the Art Nouveau era"). Incoherences in categorical annotations are quite frequent in DBpedia, for example there are over 200 entities annotated as "French films" which are not typed as "Films". Finally, DBpedia is not able to provide inferencing. For example, in Figure 4.1, the entities Veneno and Aventador, even though they are annotated as "Lamborghini vehicles", cannot be retrieved when queried simply by "vehicles". In such a way, it is exactly as if they were marked with a symbol such as †.

4.2.2 The completion of DBpedia data

Our main concern in this case lies in two aspects. Firstly, are we able to complete data using logical inferences? For example, can we *complete* the information in the dataset by indicating that the entities "Estoque" and "Gallardo" should be categorized as "Lamborghini vehicles" and "Sport cars"? Secondly, are we able to *complete* the descriptions of a given type? For example, DBpedia does not specify that an "Automobile" should have a "manufacturer". In the following, we try to answer these two questions using implications and association rules.

Consider rules provided in Table 4.2. Of course, the first three implications are only true in our dataset. This is due to the fact that we use the "closed world" assumption, meaning that our rules only apply in "our world of data" where all cars are of "Lamborghini" brand, i.e. all other information about cars that we do not know can be assumed as false Fürber and Hepp (2011). While these implications are trivial, they provide a good insight of the capabilities of our model. For instance, including a larger number of triples in our dataset would allow discovering that, while not all automobiles are manufactured by Lamborghini, they are manufactured by either a Company, an Organization or an Agent. These three *classes*[28] are types of the entity Lamborghini in DBpedia. Such a rule would allow providing a *domain* characterization to the otherwise empty description of the predicate "dbo:manufacturer" in the DBpedia schema.

The association rule given in the fourth row in Table 4.2 shows the fact that 29% of the subjects of type "Automobile" and manufactured by "Lamborghini" should be categorized by "Sports cars" and "Lamborghini vehicles" to complete the data. This actually corresponds to the entities "Estoque" and "Gallardo" in Figure 4.1. Based on this fact, we can use association rules also to create new triples that allow the completion of the information included in DBpedia.

[28] *In the OWL language sense.*

Entity	*dbo:productionStartYear*
Reventon	2008
Countach	1974
350GT	1963
400GT	1965
Islero	1967
Veneno	2012
Aventador Roadster	-
Estoque	-
Gallardo	-
Interval Pattern Concepts	
Reventon, Veneno	$\langle[2008, 2012]\rangle$
Countach,	$\langle[1974, 1974]\rangle$
350GT,400GT,Islero	$\langle[1963, 1967]\rangle$

Table 4.3: Upper table shows values of predicate *dbp:productionStartYear* for entities in Figure 4.1. The symbol - indicates that there are no values present in DBpedia for those subjects. Lower table shows the derived interval pattern concepts .

4.2.3 Pattern structures for the completion process

The aforementioned models to support linked data using FCA are adequate for small datasets as the example provided. Actually, LOD do not always consists of triples of resources (identified by their URIs) but contains a diversity of *data types* and structures including dates, numbers, collections, strings and others making the process of data processing much more complex. This calls for a formalism able to deal with this diversity of complex and heterogeneous data.

Accordingly, pattern structures are an extension of FCA which enables the analysis of complex data, such as numerical values, graphs, partitions, etc. In a nutshell, pattern structures provide the necessary definitions to apply FCA to entities with complex descriptions. The basics of pattern structures are introduced in Ganter and Kuznetsov (2001b). Below, we provide a brief introduction using interval pattern structures Kaytoue et al. (2011a).

Let us consider Table 4.3 showing the predicate *dbo:productionStartYear* for the subjects in Figure 4.1. In such a case we would like to extract a pattern in the year of production of a subset of cars. Contrasting a formal context as introduced in Chapter 1, instead of having a set M of attributes, interval pattern structures use a semi-lattice of interval descriptions ordered by a subsumption relation and denoted by $(D, \sqsubseteq)$[29]. Pattern Structures along with interval pattern structures are already detailed in Chapter 1. Using interval pattern concepts, we can extract and classify the actual pattern (and pattern concepts) representing the years of production of the cars. Some of them are presented in the lower part of Table 4.3. We can appreciate that cars can be divided in three main periods of time of production given by the intent of the interval pattern concepts.

4.2.4 Heterogeneous pattern structures

Different instances of the pattern structure framework have been proposed to deal with different kinds of data, e.g. graph, sequences, interval, partitions, etc. For linked data we propose to use the approach called "heterogeneous pattern structure" framework introduced in Codocedo and Napoli (2014) as a way to describe objects in a heterogeneous space, i.e. where there are relational, multi-valued and binary attributes. It is easy to observe that this is actually the case for linked data where the set of literals L greatly varies in nature depending on the predicate. For

[29]It can be noticed that standard FCA uses a semi-lattice of set descriptions ordered by inclusion, i.e. $(M, \subseteq)$.

the sake of simplicity we provide only the most important details of the model used for working with linked data.

When the range of a predicate (hereafter referred to as "relation") $p \in P$ is such that $range(p) \subseteq U$, we call p an "object relation". Analogously, when the range is such that $range(p) \subseteq L$, p is a "literal relation". For any given relation (object or literal), we define the pattern structure $\mathcal{K}_p = (G, (D_p, \sqcap), \delta_p)$, where $(D_p, \sqsubseteq)$ is an ordered set of descriptions defined for the elements in $range(p)$, and δ_p maps entities $g \in G$ to their descriptions in D_p. Based on that, the triple (G, H, Δ) is called a "heterogeneous pattern structure", where $H = \times D_p(p \in P)$ is the Cartesian product of all the descriptions sets D_p, and Δ maps an entity $g \in G$ to a tuple where each component corresponds to a description in a set D_p.

For an "object relation", the order in $(D_p, \sqsubseteq)$ is given by standard set inclusion and thus, the pattern structure $\mathcal{K}_p$ is just a formal context. Regarding "literal relations", such as numerical properties, the pattern structure may vary according to what is more appropriate to deal with that specific kind of data. For example, considering the predicate *dbo:productionStartYear* discussed in the previous section, $\mathcal{K}_{\text{dbo:productionStartYear}}$ should be modelled as an interval pattern structure. For the running example, the heterogeneous pattern structure is presented in Table 4.4. Cells in grey mark a *heterogeneous pattern concept* the extent of which contains cars "350GT, 400GT, Islero". The intent of this heterogeneous pattern concept is given by the tuple $(\{a, b\}, \{c\}, \{d\}, \langle[1963, 1967]\rangle)$, i.e. *"Automobiles manufactured by Lamborghini between 1963 and 1967"*. The model of heterogeneous pattern structures is the basis of the experiments which are presented in the next section.

4.3 Experimentation

To evaluate our model, four datasets were created from DBpedia, namely "Cars", "Videogames", "Smartphones" and "Countries" (see characteristics of the datasets in Table 4.5). Each dataset was created using a single SPARQL query with a unique restriction (either a fixed subject or a fixed type). A dataset consists of a set of triples whose predicate is given by the properties in Table 4.5. The heterogeneous aspect of data is illustrated by the fact that in two of the four datasets there are properties with numerical ranges.

For each dataset we calculated the set of all implications derived from the heterogeneous pattern concept lattice. Each rule of the form $X \implies Y$ was ranked according to the confidence of the rule $Y \implies X$ (the latter is referred as the "inverted rule" of the former). Thus, given that implications have always a confidence of 100%, the confidence of the inverted rule tells us how close we are from a definition, i.e. $X \iff Y$ or both rules are implications. Having an implication or not leads to the decision whether a set of RDF triples should be completed or not. For example, the following implication from the Cars dataset has an inverted rule of 92% of confidence:

$$\textit{rdf:type-dbo:MaseratiVehicles} \implies \textit{dbo:manufacturer-dbp:Maserati}$$

Accordingly, we can make of this implication a definition stating that the remainder 8% of the entities manufactured by *Maserati* should also be "typed" as *MaseratiVehicles* (recall in here that we have constructed our "world of data" by taking all "Sport Cars" from DBpedia, thus things built by Maserati which are not vehicles do not belong in our data). Of course, there are cases in which this will not be true. For example with a 90% of confidence in the opposite direction we have the implication:

$$\textit{dbo:layout-dbp:Quattro} \implies \textit{dbo:manufacturer-dbp:Audi}$$

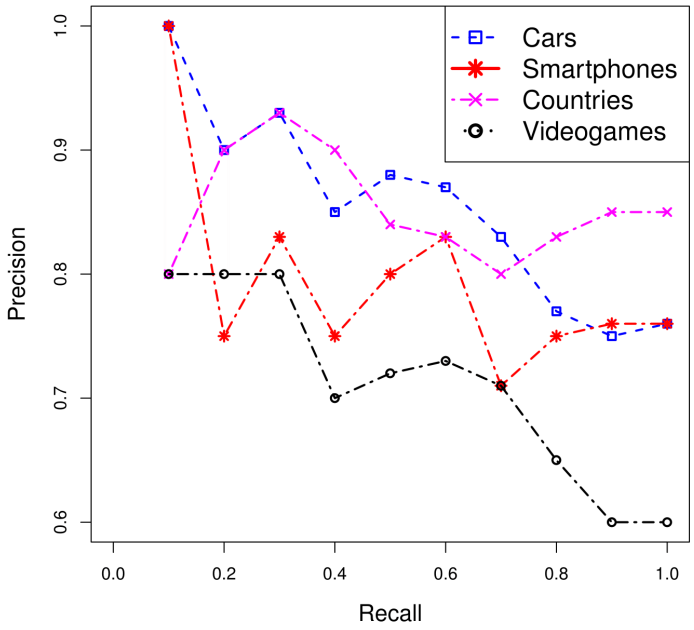


Figure 4.2: Precision at 11-points for each dataset (P@11p)

The creation of a definition from this rule (i.e. making the remainder 10% of the cars manufactured by Audi have a layout 4x4) would be wrong. While we expect that the high confidence of the opposite association rule distinguish the case when a definition should be made, a human ruling to include background information will always be needed.

4.3.1 Evaluation

Considering that there is no ground truth for any of the datasets, the reported results are given for assessing the feasibility of our approach. For each of the ranked basis of implications in the experimentation we performed a human evaluation. With the help of DBpedia, it was evaluated if an implication was likely to become a definition. The answer provided for each of the rule was binary (yes or no). For instance, in the previous examples the first implication would render a "yes" answer, while the second, a "no". Here, we use precision and recall as the evaluation measure over the ranked basis of implications. These evaluation measures are defined below based on Manning et al. (2008).

Definition 17 (Precision). *Precision (P) is the fraction of retrieved documents that are relevant and is given as follows:*

$$Precision = \frac{\#(relevantitemsretrieved)}{\#(retrieveditems)} \tag{4.1}$$

Definition 18 (Recall). *Recall (R) is the fraction of relevant documents that are retrieved and is given as follows:*

$$Recall = \frac{\#(relevantitemsretrieved)}{\#(relevantitems)} \tag{4.2}$$

They are computed using unordered sets of documents. We need to extend these measures (or to define new measures) if we are to evaluate the ranked retrieval results that are now standard with search engines. In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top k retrieved documents. Precision can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called *precision at n* or *Pn*. During this study, we measured the precision for the first 20 ranked implications (P@20) as the proportion of the rules that were likely to become a definition (those evaluated as yes) over 20 (the total number of rules taken). Actually, the precision value works as an indicator of how likely implications are useful for RDF data completion (see Table 4.5).

By contrast, since we do not have a ground truth of all the triples that should be added to each dataset, we are not able to report on recall. Nevertheless, to complement this evaluation, we provide the values of the precision at 11 points (P@11p) Manning et al. (2008). In ranked retrieval, precision represents how accurate the system is in the documents it returns (1.0 means it only returned relevant documents) while recall is the percentage of the relevant documents the system found (1.0 means system found them all). The recall of the system can be adjusted by returning more and more documents, but there is a tradeoff with precision, because it is highly likely that more errors are committed. An "interpolated precision" is where a recall level r is picked and for all recall levels $r' \geqslant r$, it is the best precision that can be achieved. Then in 11-pt interpolated average precision, there are 11 recall levels (0.0, 0.1, 0.2, ..., 1.0) and interpolated precision is found at each point. The average of these scores determine the score of the developed system.

In this study, we consider each human evaluation as the ground truth for its respective dataset and thus, each list of implications has a 100% recall. Precision at 11 points provides a notion on how well distributed are the answers in the ranking. Figure 4.2 contains the curves for each of the datasets. Values for precision at 20 points are high for all datasets and particularly for the dataset "Countries". This may be due to the fact that Countries was the only dataset built for resources with a fixed type.

A precision of 0.9 indicates that 9 out of 10 implications can be transformed into definitions by creating RDF triples that would complete the entities descriptions. Precision at 11-points shows that confidence is a good indicator on the usefulness of implications for data completion. For example, regarding the worst result i.e. the Videogames dataset, when the evaluator provides the last "yes" answer for an implication, he/she has also given a "yes" to 6 out 10 (from a total of 46). For our best result (Countries dataset) it is over 8 out of 10. Results show that confidence is a good indicator for the selection of implications in terms of data completion.

Further experimentation should be performed to assess if the triples being created are "correct" or not. As already mentioned, we assume that resources being completed are correctly linked to the implication. While this may not always be true, our approach is still useful under those circumstances given that it would allow discovering such "incorrectly" annotated entities. Finally, regarding execution times, Table 4.5 shows that even for the larger dataset, the execution time is less than a minute, and this time is perfectly acceptable for the analysis of implications.

4.4 Discussion

To conclude, in the current study we introduce a mechanism based on association rule mining for the completion of the RDF dataset. Moreover, we use heterogeneous pattern structures to deal with heterogeneity in LOD. Several experiments have been conducted over four datasets and an evaluation was conducted for each of the experiments. This study shows the capabilities of FCA

for completing complex RDF structures.

Now, let us consider the case where similar information related to one domain is present on several RDF resources and the background knowledge is present in the form of taxonomy in another RDF resource. In such a case we need to be able to directly process RDF triples. As FCA deals with only binary data, it may not be able to handle RDF statements along with RDF Schema (reference taxonomy) directly without going through some scaling for converting such information to binary format. In order to deal with such an information we use the generalization of FCA i.e., pattern structures. More specifically, we revisit pattern structures for structured attributes and then wee apply this algorithm to provide interactive exploration over RDF triples as well as RDF Schema.

	$\mathcal{K}_A$ a	b	$\mathcal{K}_B$ c	$\mathcal{K}_C$ d	$\mathcal{K}_D$ e	$\mathcal{K}_E$ f	g	$\mathcal{K}_{\text{dbo:productionStartYear}}$
Reventon	×	×	×	×	×	×		$\langle[2008, 2008]\rangle$
Countach	×	×	×	×	×			$\langle[1974, 1974]\rangle$
350GT	×	×	×	×			×	$\langle[1963, 1963]\rangle$
400GT	×	×	×	×				$\langle[1965, 1965]\rangle$
Islero	×	×	×	×				$\langle[1967, 1967]\rangle$
Veneno	×	×						$\langle[2012, 2012]\rangle$
Aventador Roadster	×	×						-
Estoque			×	×		×	×	-
Gallardo			×	×	×			-

Table 4.4: Heterogeneous pattern structure for the running example. Indexes for properties are shown in Table 4.1.

Dataset	**Cars**	**Videogames**	**Smartphones**	**Countries**
Dataset building conditions				
Restriction	dc:subject dbpc:Sports_cars	dc:subject dbpc:FPS†	dc:subject dbpc:Smartphones	rdf:type Country
Predicates	rdf:type dc:subject bodyStyle transmission assembly designer layout	rdf:type dc:subject cp‡ developer requirement genre releaseDate	rdf:type dc:subject manufacturer operativeSystem developer cpu	rdf:type dc:subject language govenmentType leaderType foundingDate gdpPppRank
Dataset Characteristics				
# Subjects	529	655	363	3,153
# Objects	1,291	3,265	495	8,315
# Triples	12,519	20,146	4,710	50,000
# Concepts	14,657	31,031	1,232	13,754
Exec. time [s]	17.32	17.14	0.7	59.82
Results				
Rules Eval.	19	46	47	50
P@20	0.85	0.7	0.79	0.9

† Front_Person_Shooters
‡ computerPlatform

Table 4.5: Summary table of experimental procedures. Upper table shows the predicates used to construct each datasets. Properties without a prefix have the default namespace "dbo:". Underlined properties have numerical ranges. Middle table show each dataset characteristics. Lower table shows experimental results. P@20 stands for "Precision at the first 20 implication".

Chapter 5

Revisiting Pattern Structures for Structured Attribute Sets

Contents

In this chapter, we revisit an original proposition on pattern structures for structured sets of attributes. There are several reasons for carrying out this kind of research work. The original proposition does not give many details on the whole framework, and especially on the possible ways of implementing the similarity operation. There exists an alternative definition without any reference to pattern structures, and we would like to make a parallel between two points of view. Moreover we discuss an efficient implementation of the intersection operation in the corresponding pattern structure. Finally, we discovered that pattern structures for structured attribute sets are very well adapted to the classification and the analysis of RDF data. We finish this chapter by an experimental section where it is shown that the provided implementation of pattern structures for structured attribute sets is quite efficient. This work has been done in collaboration with Aleksey Buzmakov, Alibek Sailanbayev and Amedeo Napoli Alam et al. (2015c).

5.1 Introduction

In this chapter, we want to make precise and develop a section of Ganter and Kuznetsov (2001a) related to pattern structures and structured sets of attributes. There are several reasons for carrying out this kind of research work. Firstly, the the pattern structures, the similarity operator $\sqcap$ and the associated subsumption operator $\sqsubseteq$ for structured sets of attributes are based on antichains and rather briefly sketched in the original paper. Secondly, there is an alternative and a more "qualitative" point of view on the same subject in **??** without any reference to pattern structures, and we would like to make a parallel between these two points of view. Finally, faced

to the problem of classifying RDF triples in the analysis of the content of Linked Open Data (LOD), we discovered that actually pattern structures for structured sets of attributes are very well adapted to solve this problem Alam and Napoli (2015a). Moreover, the classification of RDF triples provides a very good and practical example for illustrating the use of such a pattern structure and helps to reconcile the two above points of view.

Accordingly, in this chapter, we will go back to the two original definitions and show how they are related. For completing the history, it is worth mentioning that antichains, whose intersection is the basis of the similarity operation in the pattern structure for structured attribute sets, our study, are studied in the book Caspard et al. (2012). Moreover, this book cites as an application of antichain intersection an older paper from 1994 Pichon et al. (1994), written in French, about the decomposition of total orderings and its application to knowledge discovery.

Then, we proceed in presenting a way of efficiently working with antichains and intersection of antichains, which can be very useful, especially in case of large sets of data. The last section is related to a series of experiments where it is shown that pattern structures can be implemented with an efficient intersection operation and that they have a generally better behavior than scaled contexts.

5.2 Pattern Structures for Structured Attributes

5.2.1 Two original propositions on structured attribute sets

We briefly recall two original propositions supporting the present study. The first work is firstly published by Carpineto & Romano in Carpineto and Romano (1996b) and then developed in Carpineto and Romano (2004a). The second work is related to the definition of pattern structures by Ganter & Kuznetsov in Ganter and Kuznetsov (2001a).

In Carpineto and Romano (1996b, 2004a), the authors consider a formal context (G, M, I) and an extended set of attributes $M^* \supset M$ where attributes are organized within a subsumption hierarchy according to a partial ordering denoted by $\leqslant_{M^*}$. The following condition should be satisfied:

$\forall g \in G, m_1 \in M, m_2 \in M^* : [(g, m_1) \in I, m_1 \leqslant_{M^*} m_2] \Longrightarrow (g, m_2) \in I$

The subsumption hierarchy can be either a tree or an acyclic graph with a unique maximal element, as this is the case of attributes lying in a thesaurus for example. Then the building of a concept lattice from such a context can be done in two main ways. A first is to use a scaling and to complete the description of an object with all attributes implied by the original attributes. We discuss this scaling operation in detail later. The problem would be the space necessary to store the scaled context, especially in case of big data. A second way is to use an "extended intersection operation" between sets of attributes which is defined as follows. The intersection of two sets of attributes Y_1 and Y_2 is obtained by finding for each pair $(m_1, m_2), m_1 \in Y_1, m_2 \in Y_2$, the most specific attributes in M^* that are more general than m_1 and m_2, and then retaining only the most specific elements of the set of attributes generated in this way. Then if (X_1, Y_1) and (X_2, Y_2) are two concepts, we have:

$(X_1, Y_1) \leqslant (X_2, Y_2) \Longleftrightarrow \forall m_2 \in Y_2, \exists m_1 \in Y_1, m_1 \leqslant_{M^*} m_2$

In other words, this intersection operation corresponds to the intersection of two antichains as this is explained in Ganter and Kuznetsov (2001a), where the authors define the formalism of pattern structures and take as an instantiation structured attribute sets. More formally, it is assumed that the attribute set $(M, \leqslant_M)$ is finite and partially ordered, and that all attribute combinations that can occur must be order ideals (downsets) of this order. Then, any order ideal O can be described by the set of its maximal elements; $O = \{x | \exists y \in M,\ x \leqslant y\}$. *It should be*

noticed that the order considered on the attribute sets in Ganter and Kuznetsov (2001a) is reversed with respect to the order considered in Carpineto and Romano (1996b, 2004a). However, we keep the original definitions used in Ganter and Kuznetsov (2001a) in the present paragraph. These maximal elements form an antichain, and conversely, each antichain is the set of maximal elements of some order ideal. Thus, the semilattice $(D, \sqcap)$ of patterns in the pattern structure consists of all antichains of the ordered attribute set. In addition, it is isomorphic to the lattice of all order ideals of the ordered set, and thus isomorphic to the concept lattice of the context $(P, P, \not\ni)$. For two antichains AC_1 and AC_2, the infimum $AC_1 \sqcap AC_2$ consists of all maximal elements of the order ideal:

$\{m \in P \mid \exists ac_1 \in AC_1, \exists ac_2 \in AC_2, m \leqslant ac_1 \text{ and } m \leqslant ac_2\}$.

There is a "canonical representation context" (or an associated scaling operator) for the pattern structure $(G, (D, \sqcap), \delta)$ related to structured attribute sets, which is defined by the set of "principal ideals $\downarrow p$" as follows: (G, P, I) with $(g, p) \in I \iff p \leqslant \delta(g)$.

In the next section, we make precise and discuss the pattern structure for structured attribute sets by taking the point of view of filters and not of ideals in agreement with the order from Carpineto and Romano (1996b, 2004a), with the most general attributes above.

5.2.2 From Structured Attributes to Tree-shaped Attributes

An important case of structured attributes is "tree-shaped attributes", i.e., when the attributes are organized within a partial order corresponding to a rooted tree. If it is the case, then the root of the tree, denoted by $\top$, can be matched against the description of any object, while the leaves of this tree are the most detailed descriptions. For example, the root can correspond to the attribute 'Animal' and a leaf can correspond to the attribute 'Cat'; somewhere in between there could be attribute 'Mammal'.

An example of such kind of data naturally appears in the domain of semantic web data. For example, Figure 5.1 gives a small part of ACCS[30]. This attribute tree will be used as a running example and should be read as follows. If an object belongs to class C_1 (and probably to some other classes), then it necessarily belongs to classes C_{10}, C_{12}, and $\top$, e.g., if an object is a cat, then it is a mammal and an animal. Accordingly, the description of an object can include several classes, e.g., classes C_1, C_5 and C_8. Thus, some of the tree-shaped attributes can be omitted from the description of an object. However, they should be always taken into account when computing the intersection between descriptions. Thus, in order to avoid redundancy in the descriptions we can allow only antichains of the tree as possible elements in the set D of descriptions, and then, accordingly compute the intersection of antichains.

An efficient way of computing intersection of antichains is explained in the next section. Here it is important to notice that although it is a hard task to efficiently compute intersection of antichains in an arbitrary partial order of attributes, the intersection of antichains in a tree can help in computing this more general intersection. Indeed, in a partial order of attributes, we can add an artificial attribute $\top$ that can be matched against any description. Then, instead of considering an intersection of antichains in an arbitrary poset we can take a spanning tree of it with $\top$ taken as the root. Although we have lost some relations between attributes, and, thus, the size of the antichains is probably larger, we can apply the efficient intersection of antichains of tree discussed below.

[30] `https://www.acm.org/about/class/2012`

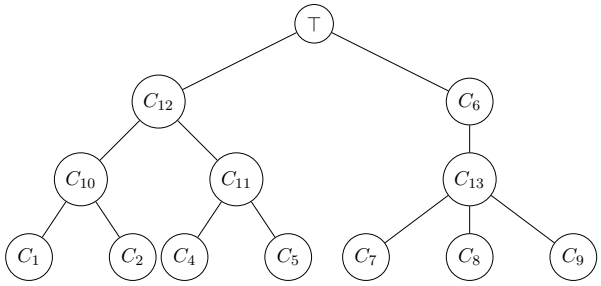


Figure 5.1: A small part from ACM Computing Classification System (ACCS).

5.2.3 On Computing Intersection of Antichains in a Tree

In this subsection we show how to efficiently solve the problem of intersection of antichains in a tree. The problem is formalized as follows. A partial order is described by the Hasse diagram corresponding to the tree. The root is denoted by $\top$ and it is larger w.r.t. the partial order than any other element in the tree. Given a rooted tree $\mathcal{T}$ and two antichains X and Y, we should find an antichain Z such that (1) for all $x \in X \cup Y$ there is $z \in Z$ such that $x \leqslant z$ and (2) no $z \in Z$ can be removed or changed to $\tilde{z} < z$ without violating requirement (1).

If the cardinality of antichains X and Y is 1 then this task is reduced to the well-known problem of a Least Common Ancestor (LCA). In 1984 it was already shown that the LCA problem can be reduced to Range Minimum Query (RMQ) problem Gabow et al. (1984). Later several simpler approaches were introduced for solving the LCA problem. Here we briefly introduce the reduction of LCA to RMQ in accordance with Bender et al. (2005).

Reduction of LCA to RMQ.

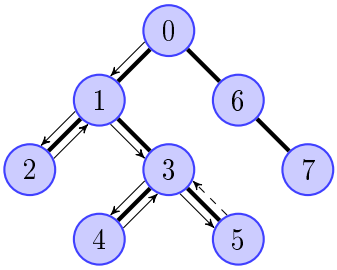
Given an array of numbers, the RMQ problem consists in efficient answering queries on the position of the minimal value in a given range (interval) of positions for this array. For example, given an array

Array	[	2	1	0	3	2]
Positions		1	2	3	4	5

where the first value is in position 1 and the last value is in position 5, the answer to the query on the position of the minimal number in the range 2–4, i.e., the corresponding part of array is [1;0;3], is 3 (the value of the 3rd element in the array is 0 and it is the minimal value in this range). Accordingly, the position of the minimal number in the range 1–2 (the part of the array is [2;1]) is 2. The good point about this problem is that it can be solved in $O(n)$ preprocessing computational time and in $O(1)$ computational time per one query Bender et al. (2005), where n is the number of elements in the array.

In order to introduce the reduction of LCA to RMQ we need to know what is the depth of a tree vertex. The *depth* of a vertex in a rooted tree is the number of edges in the shortest path from that vertex to the root of the tree.

We create the array of depths of the vertices in the tree that is used as an input array for RMQ. We build this array in the following way. We traverse the tree in depth first order (see Figure 5.2). Every time the algorithm considers a vertex, i.e., the first visit or a return to the vertex, we should put the depth of that vertex at the end of the depth array $\mathcal{D}$. We also keep



Depth array $\mathcal{D}$	[	0	1	2	1	**2**	3	**2**	3	**2**	1	0	**1**	2	**1**	0	]
Corresponding vertex		v0	v1	v2	v1	**v3**	v4	**v3**	v5	**v3**	v1	v0	**v6**	v7	**v6**	v0	
Positions		1	2	3	4	**5**	6	**7**	8	**9**	10	11	**12**	13	**14**	15	

Figure 5.2: Reducing RMQ task to LCA. Arrows show the depth first order traversal. The depth array $\mathcal{D}$ is accompanied by the corresponding vertices and positions.

track of a vertex corresponding to each depth in $\mathcal{D}$. The depth array $\mathcal{D}$ has $2|\mathcal{T}| - 1$ values, where $|\mathcal{T}|$ is the number of vertices in the tree.

Now for any value in $\mathcal{D}$ we know the corresponding vertex of the tree and any vertex of the tree is associated with several positions in $\mathcal{D}$. For example, in Figure 5.2 the value in the first position of $\mathcal{D}$, i.e., $\mathcal{D}[1]$, is 0, corresponding to the root of the tree. If we take vertex 3, then the associated values of $\mathcal{D}$ are on positions 5, 7, and 9.

Given two vertices $A, B \in \mathcal{T}$, let a be one of the positions in $\mathcal{D}$ corresponding to vertex A, let b be one of the positions in $\mathcal{D}$ corresponding to B. Then it can be shown that the vertex corresponding to the minimal value in $\mathcal{D}$ in the range a–b is the least common ancestor of A and B. For example, to find LCA between vertices 3 and 6 in Figure 5.2, one should first take two positions in $\mathcal{D}$ corresponding to vertices 3 and 6. Positions 5,7, and 9 in array $\mathcal{D}$ correspond to vertex 3, positions 12 and 14 correspond to vertex 6. Thus, we can query RMQ for ranges 5–14, 7–14, 7–12, etc. The minimal value in $\mathcal{D}$ for all these ranges is 0 located at position 11 in $\mathcal{D}$, i.e., $\texttt{RMQ}(5, 14) = 11$. Thus, the vertex corresponding to position 11, i.e., vertex 0, is the least common ancestor for vertices 3 and 6.

Let us notice that if $A \in \mathcal{T}$ is an ancestor of $B \in \mathcal{T}$ and a and b are two positions corresponding to the vertices A and B, then the position $\texttt{RMQ}(a, b)$ in $\mathcal{D}$ always corresponds to the vertex A, in most of the cases $\texttt{RMQ}(a, b) = a$. Thus we are also able to check if a vertex of $\mathcal{T}$ is an ancestor of another vertex of $\mathcal{T}$.

Now we know how to solve the LCA problem in $O(|\mathcal{T}|)$ preprocessing computational time and $O(1)$ computational time per query. Let us return to the problem of intersecting antichains of a tree.

Antichain intersection problem.

Let us first discuss the naive approach to this problem. Given two antichains $A, B \subset \mathcal{T}$, one can compute the set $\{\texttt{LCA}(a, b) \mid \forall a \in A \text{ and } \forall b \in B\}$. Then this set should be filtered for removing the comparable elements in order to get an antichain. It is easy to see that the result is the intersection of A and B but it requires at least $|A| \cdot |B|$ operations.

Let us reformulate this naive approach in terms of RMQ. Given a depth array $\mathcal{D}$ and two sets of indices $A, B \subseteq \mathbb{N}_{|\mathcal{D}|}$ forming an antichain, we should compute the set $Z = \{\texttt{RMQ}(a, b) \mid \forall a \in A \text{ and } \forall b \in B\}$ and then remove all elements $z \in Z$ such that there is $x \in Z \backslash \{z\}$ with the position $\texttt{RMQ}(z, x)$ corresponding to the same vertex as z, i.e., elements z corresponding to an

$\mathcal{D}$	[	0	1	2	3	2	3	2	1	2	3	2	3	2	1	0	1	2	3	2	3	2	3	2	1	0]
		$\top$	C_{12}	C_{10}	C_1	C_{10}	C_2	C_{10}	C_{12}	C_{11}	C_4	C_{11}	C_5	C_{11}	C_{12}	$\top$	C_6	C_{13}	C_7	C_{13}	C_8	C_{13}	C_9	C_{13}	C_6	$\top$
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Figure 5.3: Depth array, the corresponding vertices, and indices for the tree in Figure 5.1.

ancestor of another element from Z.

Let us consider for example the tree $\mathcal{T}$ given in Figure 5.1. Figure 5.3 shows the depth array, the corresponding vertices, and indices of this array. Let us show how to compute the intersection of $A = \{C_1, C_5, C_8\}$ and $B = \{C_1, C_7, C_9\}$. The expected result is $\{C_1, C_{13}\}$. First we translate the sets A and B to the indices in array $\mathcal{D}$ for RMQ, i.e., $A = \{4, 12, 20\}$ and $B = \{4, 18, 22\}$. Then we compute RMQ for all pairs from A and B:

$$\{\texttt{RMQ}(4,4) = 4, \texttt{RMQ}(4,18) = 15, \texttt{RMQ}(4,22) = 15, \cdots, \texttt{RMQ}(20,18) = 19, \cdots\}.$$

Now we should remove positions corresponding to ancestors in the tree, e.g., $\texttt{RMQ}(4,15) = 15$ and, hence, 15 should be removed. The result is $\{4, 13\}$ representing exactly $\{C_1, C_{13}\}$.

Let us discuss two points that help us to reduce the complexity of the naive approach. Consider the positions $i \leqslant l \leqslant m \leqslant j$ and $k = \texttt{RMQ}(i,j)$, $n = \texttt{RMQ}(l,m)$. Then the depth in the position k is not larger than the depth in the position n, $\mathcal{D}[k] \leqslant \mathcal{D}[n]$. Hence the position $\texttt{RMQ}(k,n)$ corresponds to the same vertex as position k. For example, in Figure 5.3 $\texttt{RMQ}(4,6) = 5$ and $\texttt{RMQ}(2,7) = 2$. The value in position 5 in the array $\mathcal{D}$ is $\mathcal{D}[5] = 2$. It is larger than the value in position 2, $\mathcal{D}[2] = 1$. Thus, the value in position returned by RMQ for the larger range is smaller than the value in position returned by RMQ for the smaller range.

Thus, given two sets of indices $A, B \subseteq \mathbb{N}_{|\mathcal{D}|}$ corresponding to antichains, we can modify the naive algorithm by ordering the set $A \cup B$ and computing RMQ only for consecutive elements from different sets, rather then for all pairs from different sets. For example, for intersecting $A = \{4, 12, 20\}$ and $B = \{4, 18, 22\}$, we join them to the set $Z = \{4_A, 4_B, 12_A, 18_B, 20_A, 22_B\}$. Then, we compute RMQ only for consecutive elements from different sets, i.e., $\texttt{RMQ}(4,4) = 4$, $\texttt{RMQ}(4,12) = 8$, $\texttt{RMQ}(12,18) = 15$, $\texttt{RMQ}(18,20) = 19$, and $\texttt{RMQ}(20,22) = 21$. The cardinality of $A \cup B$ is less then $|A| + |B|$, hence, the number of the consecutive elements is $O(|A| + |B|)$, and, thus, the number of RMQs of consecutive elements is $O(|A| + |B|)$.

However, the set Z of RMQs of consecutive elements does not not necessarily correspond to an antichain in $\mathcal{T}$. Thus we should filter this set, in order to remove all ancestors of another elements form Z. Accordingly, it is clear that to filter the set Z it is enough to check only consecutive elements of Z. For example, the intersection of $A = \{4, 12, 20\}$ and $B = \{4, 18, 22\}$ gives us the following set $Z = \{4, 8, 15, 19, 21\}$. Let us now check the RMQs of consecutive elements. $\texttt{RMQ}(4,8) = 8$, thus, 8 is an ancestor of 4 and 8 can be removed. Since 8 is removed, we compare $\texttt{RMQ}(4,15) = 15$, thus, 15 should be also removed. Then we compute $\texttt{RMQ}(4,19) = 15$, i.e., the indices 4 and 19 are not ancestors and both are kept. Now we compute $\texttt{RMQ}(19,21) = 19$ and, thus, 19 should be removed (actually positions 19 and 21 correspond to the same vertex C_{13} and one of them should be removed). Thus, the result of intersecting A and B is $\{4, 21\}$ corresponding to the antichain $\{C_1, C_{13}\}$.

Since the number of elements in the set Z is $O(|A| + |B|)$, then overall complexity of computing intersection for two antichains $A, B \subset \mathcal{T}$ of a tree $\mathcal{T}$ is $O(|A| + |B|)$ or, taking into account that the cardinality of an antichain in a tree is less then the number of leaves (vertices having no descendants) in this tree, the complexity of computing intersection of two antichains is $O(|\texttt{Leaves}(\mathcal{T})|)$.

Antichain intersection by scaling.

An equivalent approach for computing intersection of antichains is to scale the antichains to the corresponding filters. A *filter corresponding to an antichain* in a poset is the set of all elements of the poset that are larger then at least one element from the antichain. For example, let us consider a tree-shaped poset in Figure 5.1. A filter corresponding to the antichain $\{C_1, C_5, C_8\}$ is the set of all ancestors of all elements from the antichain, i.e., it is equal to $\{C_1, C_{10}, C_{12}, \top, C_5, C_{11}, C_8, C_{13}, C_6\}$.

The set-intersection of filters corresponding to the given antichains is a filter corresponding to the antichain resulting from intersection of the antichains. However this approach has a higher complexity. Indeed, the size of a filter is $O(|\mathcal{T}|)$ and, thus, the computational complexity of intersecting two antichains by means of a scaling is $O(|\mathcal{T}|)$ which is harder then $O(|\texttt{Leaves}(\mathcal{T})|)$ for intersecting antichains directly. Indeed, the number of leaves in a tree can be dramatically smaller than the number of vertices in this tree. For example, the number of vertices Figure 5.1 is 13, while the number of leaves is only 7. Thus, the direct intersection of antichains is more efficient than the intersection by means of a scaling procedure.

Relation to intersection of antichains in partially ordered sets of attributes.

As it was mentioned in the previous section, the intersection of antichains in arbitrary posets can be reduced to the intersection of antichains in a tree. However, the size of the antichain representing a description of an object can increase. Indeed, since we have reduced a poset to a tree, some relations have been lost, and thus the attributes that are subsumed in the poset for a given antichain A are no more subsumed in the tree for A, and hence should be added to A. However, the reduction is still more computationally efficient than computing the intersection of antichains in a poset by means of a scaling as it is discussed in the previous paragraph. However, for the reduction it could be interesting to find the spanning tree with the minimal number of leaves. Unfortunately, this is an NP-complete task and it thus cannot be applied for increasing the computational efficiency Salamon and Wiener (2008). We should notice here that there is some work that solves the LCA problem for more general cases, e.g., lattices Aït-Kaci et al. (1989) or partially ordered sets Bender et al. (2005). However, it is an open question whether these works can help to efficiently compute intersection of antichains in the corresponding structures.

5.3 Experiments and Discussion

Several experiments are conducted using publicly available data on a MacBook with a 1.3GHz Intel Core i5, 4GB of RAM running OS X Yosemite 10.3. We have used `FCAPS`[31] software developed in C++ for dealing with different kinds of pattern structures. It can build a concept lattice starting from a standard formal context or from object descriptions given as antichains of a given tree. The last one is based on the similarity operation that is discussed above.

We performed our experiments on two datasets from different domains i.e., DBLP and biomedical data. In these datasets, object descriptions are given as subsets of attributes. A taxonomy of the attributes is already known based on domain knowledge. We compute a concept lattice in two different ways. In the first one, we directly compute the concept lattice from the antichains in a taxonomy. In the second one we scale every description to the corresponding filter of the taxonomy. After this we do not rely on the taxonomy and process the scaled context with standard FCA.

[31] `FCAPS` will be available soon.

Table 5.1: Results of the experiments with different kind of data.

#objects is the number of objects in the corresponding dataset. #attributes is the number of numerical attributes before scaling. $|G|$ is the number of objects used for building the lattice. $|\mathcal{T}|$ is the size of the attribute tree and the number of attributes in the scaled context $|M|$. $\texttt{Leaves}(\mathcal{T})$ is the number of leaves in the attribute tree. $|\mathcal{L}|$ is the size of the concept lattice for the corresponding data. $t_{\mathcal{T}}$ is the computational time for data represented as a set of antichains in the attribute tree. $t_{\mathbb{K}}$ is the computational time represented by a scaled context, i.e., by a set of filters in the attribute tree; '*' shows that the we are not able to build the whole lattice. t_{num} is the computational time for numerical data represented by an interval pattern structure.

(a) Real data experiments.

Dataset	$\lvert G\rvert$	$\lvert\mathcal{T}\rvert$	$\texttt{Leaves}(\mathcal{T})$	$\lvert\mathcal{L}\rvert$	$t_{\mathcal{T}}$	$t_{\mathbb{K}}$
DBLP	5293	33207	33198	10134	45 sec	21 sec
Biomedical Data	63	1490	933	1725582	145 sec	162 sec

(b) Numerical data experiments.

Dataset	#objects	#attributes	$\lvert G\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\texttt{Leaves}(\mathcal{T})\rvert$	$\lvert\mathcal{L}\rvert$	$t_{\mathcal{T}}$	$t_{\mathbb{K}}$	t_{num}
BK	96	5	35	626	10	840897	37 sec	42 sec*	19 sec
LO	16	7	16	224	26	1875	0.043 sec	0.088 sec	0.024 sec
NT	131	3	131	140	6	128624	3.6 sec	6.8 sec	3.1 sec
PO	60	16	22	1236	58	416837	49 sec	57 sec*	10.7 sec
PT	5000	49	22	4084	60	452316	50 sec	38 sec*	15 sec
PW	200	11	94	436	21	1148656	60 sec	49 sec*	48 sec
PY	74	28	36	340	53	771569	46 sec	40 sec*	21 sec
QU	2178	4	44	8212	8	783013	28 sec	30 sec*	15.4 sec
TZ	186	61	31	626	88	650041	58 sec	43 sec*	22 sec
VY	52	4	52	202	15	202666	5.9 sec	11.6 sec	3 sec

The first data set is DBLP, from which we extracted a subset of papers with their keywords published in conferences in Machine Learning domain. The taxonomy used for classifying such kind of triples is ACM Computing Classification System (ACCS)[32].

The second data set belongs to the domain of life sciences. It contains information about drugs, their side effects (SIDER[33]), and their categories (DrugBank[34]). The taxonomies related to this dataset are MedDRA [35] for side effects and MeSH[36] for drug categories.

The parameters of the datasets and the computational results are shown in Table 5.1a. It can be noticed that for DBLP the context consists of 5293 objects and 33207 attributes, in the taxonomy of the attributes we have 33198 leaves meaning that most of attributes are mutually incomparable. It took 45 seconds to produce a lattice having 10134 concepts directly from the descriptions given by antichains of the taxonomy. To produce the same lattice starting from a scaled context the program only takes 21 seconds. However, if we consider the biomedical data,

[32] `https://www.acm.org/about/class/2012`
[33] `http://sideeffects.embl.de/`
[34] `http://www.drugbank.ca/`
[35] `http://meddra.org/`
[36] `http://www.ncbi.nlm.nih.gov/mesh/`

the approach based on antichains is better. Indeed, it takes 145 seconds, while the computation starting from the scaled contexts takes 162 seconds. In this case, the dataset contains 1490 attributes with 933 leaves. Thus, the direct approach works faster if the number of leaves is significantly smaller than the number of vertices. It is worth noticing that the size of antichains is significantly smaller than the size of the filters, and thus our approach is more efficient. However, when the number of leaves is comparable to the number of vertices, the our approach is slower. Although in this case our approach has the same computational complexity as the scaling approach, the antichain intersection problem requires more efforts than the set intersection.

Since the efficiency of the antichain approach is high for the trees with a low number of leaves, we can use this method to increase efficiency of standard FCA for special kind of contexts. In a context (G, M, I) an attribute m_1 can be considered as an ancestor of another attribute m_2 if any object containing the attribute m_2 also contains the attribute m_1. Accordingly we can construct an attribute tree $\mathcal{T}$ and rely on it for computing intersection operation. In this case the set of attributes M and the set of vertices of $\mathcal{T}$ are the same and $|M| = |\mathcal{T}|$. The second part of the experiment was based on this observation.

We used numerical data from Bilkent University in the second part of the experiments[37]. It was converted to formal contexts by the standard interodinal scaling. The scaled attributes are closely connected, i.e., there is a lot of pairs of attributes (m_1, m_2) such that the set of objects described by m_1 is a subset of objects described by m_2, i.e., $(m_1)' \subseteq (m_2)'$. Thus, we can say that $m_1 \leqslant m_2$. Using this property we built attribute trees from the scaled contexts. These trees had many more vertices than leaves, thus, the approach introduced in this study should be efficient. We compare our approach with the scaling approach. Moreover, recently, it was shown that interval pattern structures (IPS) can be efficiently used to process such kind of data Kaytoue et al. (2011b). Accordingly we also compared our approach with IPS.

The results are shown in Table 5.1b. Compared to Table 5.1a it has several additional columns. First of all, since for numerical data we typically got large lattices, in most of the cases we considered only part of the objects. The actual number of used objects is given in the column $|G|$, while the total size of the dataset is given in the column '#objects', e.g., `BK` dataset contained 96 objects, while we have used only 35. In addition for every dataset we also provide the number of the numerical attributes, e.g., `BK` has 5 numerical attributes. We should notice that when we built the lattice from some datasets by standard FCA, the lattice was so large that the memory was swapping and we stopped the computation. It was not the case for our approach since antichains requires less memory to store than the corresponding filters. The fact of swapping is shown by '*' next to computational time in column $t_{\mathbb{K}}$. In addition we also show the time for IPS to process the same dataset. For example, the processing of `BK` dataset took 37 seconds by our approach, took more than 42 seconds by standard FCA and memory had started swapping, and took 19 seconds by IPS.

This experiment shows that our approach takes not only less time to compute concept lattice, but also requires less memory, since there is no memory swapping. We can also see that the computation time for IPS is smaller than for our approach. However, IPS is only applicable for numerical data, while our approach can be applied for all cases when attributes of a context are structured. For example, we can deal with graph data scaled to the set of frequent subgraphs where many such attributes are subgraphs of other attributes.

[37] `http://funapp.cs.bilkent.edu.tr/DataSets/`

5.4 Discussion

In this chapter we recalled two approaches for dealing with structured attributes and explained how we can compute intersection of antichains in tree-shaped posets of attributes, an essential operation for working with structured attributes. Our experiments showed the computational efficiency of the proposed approach. In the next chapter, we discuss the application of the similarity measure proposed in this chapter. We introduce a framework that allows user to interactively explore RDF graph through a visualization tool. We directly involve the user in selecting the user/task specific datasets or subsets of data sets. Then, we apply our algorithm for obtaining classes of RDF triples. These classes are organized in a partially ordered manner using a pattern concept lattice, which then allows the user to further specify which parts of the pattern concept lattice is interesting for the user and which are not. User has the ability to hide non-interesting part of the concept lattice which allows the user to reduce her navigation space.

Chapter 6

Exploratory Data Analysis of RDF Resources using Formal Concept Analysis

Contents

With an increased interest in machine processable data, many datasets are now published in RDF (Resource Description Framework) format in Linked Data Cloud. These data are distributed over independent resources which need to be centralized and explored for domain specific applications. This chapter proposes a new approach based on interactive data exploration paradigm using Pattern Structures, an extension of Formal Concept Analysis, to provide exploration and navigation over Linked Data through concept lattices. It takes RDF triples and RDF Schema based on user requirements and provides one navigation space resulting from several RDF resources. This navigation space allows user to navigate and search only the part of data that is interesting for her. This chapter is based on Alam and Napoli (2015b,c).

6.1 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 data in the form of entity-relationship are published as Linked Open Data (LOD) cloud Bizer et al. (2009a). 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 LOD is that it follows a *decentralized* publication model Oren et al. (2008), 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 can be linked to these data to make sense based on real world conception. Some of the resources in LOD only contain the schema without the instances such as SWRC ontology Sure et al. (2005) and some semantic web documents may only contain RDF triples without the RDF Schema such as DBLP[38]. The problem of how to provide applications which allow guided navigation and exploration over these data sources still persists.

This chapter introduces a framework based on interactive data exploration van Leeuwen (2014) paradigm using Pattern Structures Ganter and Kuznetsov (2001a) which is an extension of Formal Concept Analysis (FCA) Ganter and Wille (1999). The goal of exploratory data mining is to provide an expert with an insight into the data. One of the basic principals is to take into account the user-requirements and task-specific information. This way the patterns obtained by pattern mining algorithms are more relevant and interesting. For doing so, the pattern mining algorithms need to be combined with visualization tools for providing human-computer-interaction. Moreover, in order to make these pattern mining algorithms useful, there is a need to apply these algorithms to smaller datasets or only interesting and relevant subsets of the datasets. This enables user to interactively explore the data and identify patterns of interest.

In the current study, we use small datasets/subsets of datasets present in the form of RDF graphs over Linked Open Data and use FCA for giving the user an insight into the RDF datasets with the help of a visualization tool. During this process we directly involve the user in defining the datasets or part of datasets she is interested in. Then FCA is applied to these data and then patterns are obtained. Finally, the patterns computed are explored with the help of visualization tool. A complete iterative process of interactive data exploration on Linked Open Data is shown in Figure 6.1. This framework takes into account the prior requirements of the user and selects the data sources which are relevant to the user application distributed over several resources over Linked Open Data Cloud in the form of RDF triples and RDF Schema. This new approach called RDF-Pattern Structures, takes RDF triples and the RDF Schema present on distributed locations as an input and integrates them into one *navigation space*. This navigation space provides centralization over distributed RDF resources and keeps a partially ordered organization of RDF triples with respect to RDF Schema. It serves as a navigational as well as an interactive exploratory space for the user where she can identify the samples of the data which are not relevant to her while navigation. These identified samples are then hidden from the user and are recorded as irrelevant. In the second iteration a new navigation space is built on user demand which keeps only the information that is relevant to the user. This navigation space is explored

[38] `http://dblp.l3s.de/d2r/`

and navigated for the purpose of data analysis and information retrieval over several data sources. To date this is the first attempt to deal with RDF data and exploration techniques with Pattern Structures and FCA.

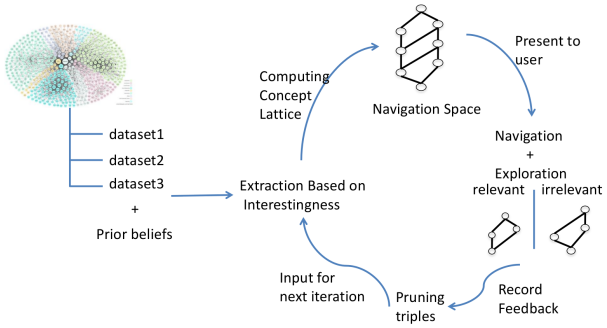


Figure 6.1: Interactive Data Exploration over RDF Data through Concept Lattices

6.1.1 Motivating Example

Consider the scenario where the user wants to search for the papers published in conferences or journals related to her field of research. The problems faced by her for retrieving such papers are as follows:

- She looks-up DBLP page of the authors working in her field. In that case she has to go through all the publications of each author and then browse through the DBLP pages of the co-authors of this author.
- Moreover, if she is searching for the papers which are targeting more than one problem areas such as information retrieval and World Wide Web then it is not possible to retrieve such papers directly.
- Finally, she will also not be able to detect the communities of the author who often work together to retrieve the relevant papers or establish the collaboration with these authors.

Accordingly we try to guide such kind of research based on a concept lattice which is built from an initial query and then is explored by the user according to her preferences.

6.2 Towards RDF-Pattern Structures

For creating navigation spaces over RDF as well as RDF Schema to provide navigation and exploration, the first operation is to select the RDF data set with no RDF Schema information and a suitable RDF Schema associated to these RDF triples from distributed data sources. This RDF schema will be used for organizing the RDF triples. An RDF Schema which will be used as a reference for comparing objects in the RDF triples is referred to as *reference schema*. This RDF Schema could also be a semantic resource such as Word Net or YAGO Suchanek et al. (2007) or DBpedia ontology. In the running scenario as we are targeting the organization of RDF triples in DBLP, we use ACCS as our reference schema.

A subset of RDF triples is extracted according to user requirements and these RDF triples are defined in terms of patterns structures i.e., specifying the entities, their descriptions and the mapping from entities to description. After these two operations, we define a similarity measure ⊓

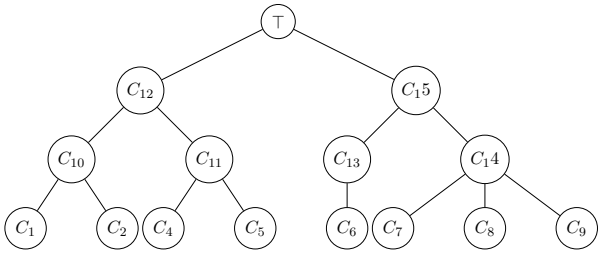


Figure 6.2: A small part from ACM Computing Classification System.

over two descriptions which we generalize to the set of descriptions. After defining the similarity measure, we explain how an RDF-pattern concept lattice is built using this similarity measure. More generally, an organization of RDF triples is built, based on concept lattice, w.r.t background knowledge. Finally, this lattice is used for navigation and interactive exploration purposes.

6.2.1 From RDF Triples to RDF-Pattern Structures

Firstly, we represent RDF triples extracted by the SPARQL query in Listing 6.1 as entities and their descriptions. The pattern structures are given as $(G, (D, \sqcap), \delta)$. A subject in an RDF triple is mapped to an entity g in the set of entities G and predicate object pair (p:o) is mapped to a description $d \in D$. As the set of entities G represent the subjects in triples, we represent it as S. The descriptions D are termed as descriptions and are denoted as D_s.

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:<http://purl.org/dc/terms/>

SELECT distinct ?title ?keywords ?author
where {
?paper dc:creator ?a .
?a rdfs:label ?author .
?paper dc:subject ?keywords .
?paper dc:title ?title .
FILTER(
   regex(STR(?keywords), "pattern based classification", "i")
|| regex(STR(?keywords), "unsupervised classification", "i"))
```

Listing 6.1: SPARQL for extracting triples. Prefixes are also defined in Table 6.1.

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 $d_{ij} = \{p_j : \{o_1, o_2, \ldots, o_m\}\}$ where $j \in \{1, \ldots, q\}$. In the running scenario, we have $p_1 = dc : subject$ and $p_2 = dc : creator$ as shown in Table 6.1. For p_1 the elements in the range can be compared with the help of reference schema and for p_2 the elements in the range are names of the author which can not be compared.

After this organization a suitable RDF Schema is selected based on what user needs and the fact that it contains the objects in the triples. RDF Schema contains many constructs such as property, sub-property etc. along with `rdfs: subClassOf` information but in this work we use the RDF Schema predicates such as `rdfs:subClassOf`, `skos:broader` which lead to a tree structure in an RDF graph. An RDF Schema from ACM Computing Classification System is shown in Figure 6.2 for the set of objects connected to predicate p_1. While the objects connected

Abbreviation	Term	Abbreviation	Term
p_1 (dc:subject)	http://purl.org/dc/elements/1.1/subject	p_2 dc:creator	http://purl.org/dc/elements/1.1/creator
p_3 (dc:title)	http://purl.org/dc/elements/1.1/title	p_4	rdfs:subClassOf
C_1	Web Crawling	C_8	Recommender Systems
C_2	Web Indexing	C_9	Clustering and Classification
C_3	Page and site Ranking	C_{10}	Web Search Engines
C_4	RDF	C_{11}	Semantic Web Description
C_5	OWL	C_{12}	World Wide Web
C_6	Similarity Measure	C_{13}	Retrieval Models and Ranking
C_7	Question Answering	C_{14}	Retrieval Tasks and Goals

Table 6.1: Prefixes and Abbreviations of the terms used in the rest of the chapter.

tid	Subject	Predicate	Object	Dataset
t1	s_1	p_1	o_{11}	DBLP
t2	s_1	p_2	o_{12}	DBLP
t3	s_2	p_1	o_{16}	DBLP
t4	s_2	p_2	o_{22}	DBLP
t5	s_1	rdf:type	Publication	DBLP
t6	o_{12}	rdf:type	Author	DBLP
t7	o_{11}	p_4	C_1	ACCS
t8	o_{12}	p_4	C_1	ACCS
t9	C_1	p_4	C_3	ACCS
⋮	⋮	⋮	⋮	⋮

Table 6.2: RDF triples about papers with their authors and keywords from DBLP.

to the second predicate p_2 do not have any associated schema. The circles represent classes and the lines between these circles represent predicate rdfs:subClassOf/skos:broader. Each object is replaced with their classes i.e., $C_1(o_{11})$ meaning that o_{11} is an instance of class C_1. Then, the description $\{(p_1 : \{o_{11}, o_{12}, \dots\})\}$ is given as $\{(p_1 : \{C_1, C_2, \dots\})$ This replacement is only performed for the description for which there is some RDF Schema present. The triples $t1, t2, t3$ in Table 6.2 are represented as entities and descriptions where the entity s_1 has the description $\delta(s_1) = \{d_{11}, d_{12}\}$ where $d_{11} = p_1 : \{C_1, C_2, C_4, C_7\}$ and $d_{12} = p_2 : \{o_{21}, o_{22}\}$. Table 6.3 shows a final representation of the RDF triples after replacing the objects with their corresponding classes (if reference schema is available) as entity descriptions.

Entities S	d_{i1}	d_{i2}
s_1	$(p_1 : \{C_1, C_2, C_7\})$	$(p_2 : \{o_{21}\})$
s_2	$(p_1 : \{C_6, C_8, C_9\})$	$(p_2 : \{o_{22}, o_{23}\})$
s_3	$(p_1 : \{C_4, C_5\})$	$(p_2 : \{o_{22}, o_{24}, o_{25}\})$
s_4	$(p_1 : \{C_4, C_7, C_8\})$	$(p_2 : \{o_{23}\})$
s_5	$(p_1 : \{C_8, C_9\})$	$(p_2 : \{o_{22}, o_{23}, o_{25}\})$

Table 6.3: RDF Triples as entities S and semantic descriptions D_s.

6.2.2 Similarity Operation Over Classes

The similarity operation between two different classes is computed w.r.t. the Least Common Subsumer (LCS) of two classes. Definition 19 is the definition of Least Common Subsumer in a partially ordered set. We denote $\sqsubseteq$ as $\preceq$ to avoid confusion between the concept lattice subsumption order.

Definition 19 (Least Common Subsumer). *Given a partially ordered set $(S, \preceq)$, 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 $C \preceq E$ and $D \preceq E$ and E is least i.e., if there is a class E' such that $C \preceq E'$ and $D \preceq E'$ then $E \preceq E'$.*

Given a reference schema which in our case is a tree, two elements whose LCS is $\top$ are considered as non-similar. Now we are in the reference schema, we consider classes (these are classes of the objects in triples). We want to compute the similarity of descriptions $p_i : X$ and $p_j : Y$. First, we compute the similarity with the following constraints:

- *Case 1:* $p_i = p_j$.
- *Case 2:* we consider X and Y be the set of objects such that $X = \{o_i\}$, $Y = \{o_j\}$ where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. Here we consider the case when there is no reference schema for the elements in X and Y. If $o_i = o_j$ then $X \cap Y = \{o_i\}$ otherwise $\varnothing$.

  For instance, in Table 6.3 we have d_{i2} for which there is no reference schema available. Lets choose two sets of objects then accordingly we have, $p_2 : \{\{o_{22}, o_{23}\} \cap \{o_{22}, o_{23}, o_{25}\}\} = p_2 : \{o_{22}, o_{23}\}$.
- *Case 3:* In this case, let $X = \{C\}$ and $Y = \{D\}$ and the elements of X and Y are in the reference schema, we consider the classes of the elements and then the LCS of these elements. More formally, it can be defined as follows: Let us take two descriptions $c = p_1 : C$ and $d = p_1 : D$ then:

$$\begin{aligned} c \sqsubseteq d &= p_1 : C \sqsubseteq p_1 : D \\ \Leftrightarrow p_1 : C \sqcap p_1 : D &= p_1 : C \\ \Leftrightarrow p_1 : lcs(C, D) &= p_1 : C \\ \Leftrightarrow D \preceq C & \quad \textit{in the reference schema} \end{aligned}$$

  This is the fundamental that should be verified by the similarity operator. The definition $\sqcap$ implies that specific class subsume the general class which is the super class. For example, for descriptions $D_s = \{C_2, C_4, C_5\}$ and the reference schema shown in Figure 6.2 the meet semi-lattice is given in Figure 6.3. Here, $C_{11} = C_4 \sqcap C_5$ and is subsumed by the class $C_{11} \sqsubseteq C_4$ and $C_{11} \sqsubseteq C_5$.
- *Case 4:* In this case we compute the similarity between sets of classes. The similarity between the sets of classes is defined in the same way as Case 3. LCS is computed for every pair of classes in two different sets of description and then only the most specific classes are retained. Let C and D be two sets of classes then we have:

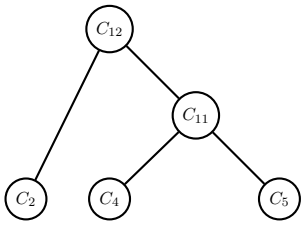


Figure 6.3: $(D_s, \sqcap)$

$$\begin{aligned} C \sqsubseteq D \quad &= \quad p_1 : C \sqsubseteq p_1 : D \\ \Leftrightarrow p_1 : C \sqcap p_1 : D \quad &= \quad p_1 : C \\ \Leftrightarrow p_1 : lcs(c_i, d_j) \quad &= \quad p_1 : c_i \\ where\ c_i \in C \quad &and \quad d_j \in D \end{aligned}$$

It means that $\forall c_1 \in C, \exists d_1 \in D, d_1 \preceq c_1$. The process is explained with the help of only one reference schema, multiple schemas can also be considered for several sets of objects.

For instance, lets choose two sets of classes $C = \{C_1, C_2, C_7\}$ and $D = \{C_4, C_7, C_8\}$ then the LCS between each pair will generate the set $\{C_{12}, C_7, C_{14}\}$. As only the specific elements are retained, the final set obtained is $E = \{C_{12}, C_7\}$. Accordingly, $E \sqsubseteq C$ and $E \sqsubseteq D$.

6.2.3 Building the RDF Pattern Concept Lattice

A representation of RDF triples as the set of descriptions given in Table 6.3 can be formalized as a pattern structure $(S, (D_s, \sqcap), \delta)$. When $A \subseteq S$ is a set of entities and $d \in (D_s, \sqcap)$ is a semantic description containing classes and objects then $A^{\square}$ returns a set of common objects (if reference schema is absent) present in the descriptions of each subject in A and $A^{\diamond}$ returns LCS of the classes (when reference schema is available) in the description of A. On the other hand, $d^{\square\diamond}$ returns the set of subjects which are described by the objects/classes of objects included in d.

Firstly, we explain how to build the lattice for the descriptions having the reference schema. So we build the lattice only with the descriptions d_{i1} in Table 6.3. The similarity between two subjects can be given as:

$$\begin{aligned} \{s_1, s_3\}^{\diamond} &= \prod_{s \in \{s_1, s_3\}} \delta(s) \\ &= \delta(s_1) \sqcap \delta(s_3) \\ &= \langle (p_1 : \{C_1, C_2, C_7\}) \sqcap (p_1 : \{C_4, C_5\}) \rangle \\ &= \langle (p_1 : \{lcs(C_1, C_4), lcs(C_1, C_5), \dots\}) \rangle \\ &= \langle (p_1 : \{C_{12}\}) \rangle \\ \langle (p_1 : \{C_{12}\}) \rangle^{\diamond} &= \{s \in S | \langle (p_1 : \{C_{12}\}) \rangle \sqsubseteq \delta(s)\} \\ &= \{s_1, s_3, s_4\} \end{aligned}$$

The pair $(A, d) = (\{s_1, s_3, s_4\}, \langle (p_1 : \{C_{12}\}) \rangle)$ is a pattern concept ($K\#2$ in Table 6.4) meaning that $A^{\diamond} = d$ and $d^{\diamond} = A$. A set of all pattern concept creates a pattern concept lattice shown in Figure 6.4. The subsumption order $\sqsubseteq$ between two patterns in pattern concepts (A_1, d_1) and (A_2, d_2) is given as follows: $(A_2, d_2) \sqsubseteq (A_1, d_1) \iff \forall c_2 \in d_2, \exists c_1 \in d_1, c_1 \preceq c_2$ *(in*

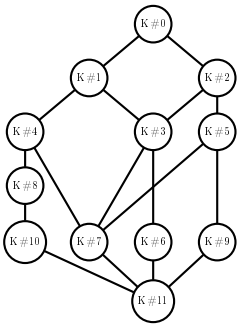


Figure 6.4: Pattern Concept lattice for DBLP and ACCS.

$K\#ID$	**Extent**	**Intent**
$K\#1$	s_1, s_2, s_4, s_5	$(p_1 : \{C_{14}\})$
$K\#2$	s_1, s_3, s_4	$(p_1 : \{C_{12}\})$
$K\#3$	s_1, s_4	$(p_1 : \{C_7, C_{12}\})$
$K\#4$	s_2, s_4, s_5	$(p_1 : \{C_8\})$
$K\#5$	s_3, s_4	$(p_1 : \{C_4\})$
$K\#6$	s_1	$(p_1 : \{C_1, C_2, C_7\})$
$K\#8$	s_2, s_5	$(p_1 : \{C_8, C_9\})$
$K\#7$	s_4	$(p_1 : \{C_4, C_7, C_8\})$
$K\#9$	s_3	$(p_1 : \{C_4, C_5\})$
$K\#10$	s_2	$(p_1 : \{C_6, C_8, C_9\})$

Table 6.4: Details of *Pattern Concept lattice* in Figure 6.4.

the reference schema). Similarly, it can be seen that $(A_2, d_2) \sqsubseteq (A_1, d_1) \Longleftrightarrow (A_1, d_1) \sqcap (A_2, d_2) = (A_2, d_2)$.

Now we consider an example with all the cases described in the previous section. From Table 6.3 we have $\delta(s_1) = (p_1 : \{C_1, C_2, C_7\}), (p_2 : \{o_{21}\})$ and $\delta(s_3) = (p_1 : \{C_4, C_5\}), (p_2 : \{o_{22}, o_{24}, o_{25}\})$, where p stands for a predicate, C stands for a class and o stands for an object having no class.

In this case, the first description $(p_1 : \{C_1, C_2, C_7\})$ has an associated reference schema, while the reference schema for second description $(p_2 : \{o_{21}\})$ is absent. Then the similarity between these two subjects is given as follows:

$$
\begin{aligned}
\{s_1, s_3\}^{\square\diamond} &= \prod_{s \in \{s_1, s_3\}} \delta(s) \\
&= \delta(s_1) \sqcap \delta(s_3) \\
&= \langle (p_1 : \{C_1, C_2, C_7\})(p_2 : \{o_{21}\}) \\
&\quad \sqcap (p_1 : \{C_4, C_5\})(p_2 : \{o_{22}, o_{24}, o_{25}\}) \rangle \\
&= \langle (p_1 : \{C_1, C_2, C_7\}) \sqcap (p_1 : \{C_4, C_5\}), \\
&\quad (p_2 : \{o_{21}\}) \sqcap (p_2 : \{o_{22}, o_{24}, o_{25}\}) \rangle \\
&= \langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle \\
\langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle^{\square\diamond} &= \{s \in S | \langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle \sqsubseteq \delta(s)\} \\
&= \{s_1, s_3, s_4\}
\end{aligned}
$$

The pair $(A, d) = (\{s_1, s_3, s_4\}, \langle (p_1 : \{C_{12}\})(p_2 : \{\}) \rangle)$ is a pattern concept ($K\#2$ in Table 6.5) meaning that $A^{\square\diamond} = d$ and $d^{\square\diamond} = A$. A set of all pattern concept creates a pattern concept lattice shown in Figure 6.5 and is termed as *navigation space.* Further details about the algorithm

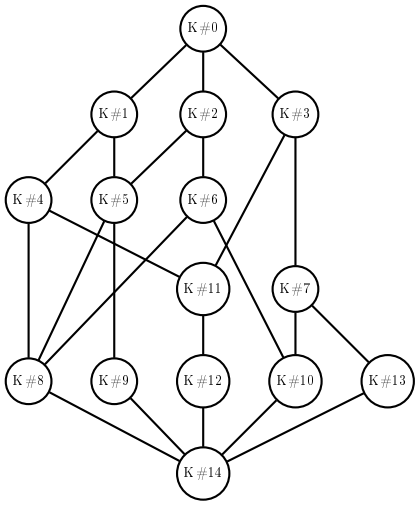


Figure 6.5: Pattern Concept lattice for DBLP and ACCS with and without reference schema.

$K\#ID$	Extent	Intent
$K\#1$	s_1, s_2, s_4, s_5	$(p_1 : \{C_{14}\})$, $(p_2 : \{\})$
$K\#2$	s_1, s_3, s_4	$(p_1 : \{C_{12}\})$, $(p_2 : \{\})$
$K\#3$	s_2, s_3, s_5	$(p_1 : \{\})$, $(p_2 : \{o_{22}\})$
$K\#4$	s_2, s_4, s_5	$(p_1 : \{C_8\})$, $(p_2 : \{o_{23}\})$
$K\#5$	s_1, s_4	$(p_1 : \{C_{12}, C_7\})$, $(p_2 : \{\})$
$K\#6$	s_3, s_4	$(p_1 : \{C_4\})$, $(p_2 : \{\})$
$K\#8$	s_4	$(p_1 : \{C_4, C_7, C_8\})$, $(p_2 : \{o_{23}\})$
$K\#7$	s_3, s_5	$(p_1 : \{\})$, $(p_2 : \{o_{22}, o_{25}\})$
$K\#9$	s_1	$(p_1 : \{C_1, C_2, C_7\})$, $(p_2 : \{o_{21}\})$
$K\#10$	s_3	$(p_1 : \{C_4, C_5\})$, $(p_2 : \{o_{22}, o_{24}, o_{25}\})$
$K\#11$	s_2, s_5	$(p_1 : \{C_8, C_9\})$, $(p_2 : \{o_{22}, o_{23}\})$
$K\#12$	s_2	$(p_1 : \{C_6, C_8, C_9\})$, $(p_2 : \{o_{22}, o_{23}\})$
$K\#13$	s_5	$(p_1 : \{C_8, C_9\})$, $(p_2 : \{o_{22}, o_{23}, o_{25}\})$

Table 6.5: Details of *Pattern Concept lattice* in Figure 6.5.

used for computing Least Common Subsumer are discussed in Alam et al. (2015c). It discusses how pattern structures is adapted for dealing with structured attribute sets.

6.3 Navigation and Interactive Exploration over Pattern Concept Lattice

6.3.1 Navigation Operations

Several navigation operations can be applied over the *navigation space* for obtaining precise information by navigation Codocedo et al. (2014). Here, each concept contains a group of subjects connected to the classes of the objects. The most general concepts near the top of the pattern concept lattice contain more number of subjects (entities) and lesser number of classes (description) meaning that the descriptions are very general. As the lattice is navigated downward more specific descriptions exist with lesser number of subjects.

Let us consider the scenario discussed in section 6.1.1. We consider the navigation space shown in Figure 6.5. If user wants to retrieve the scientific papers on some specific topic such as World Wide Web, she would easily locate the concept containing only the papers about

this topic i.e., $K\#2$. The retrieved papers will be s_1, s_3, s_4. For narrowing down her papers which are related to World Wide Web and Question Answering, the lattice can be navigated downwards to obtain $K\#5$ which contain the two papers s_1, s_4. Now the user has the choice for further narrowing down w.r.t. more specified classes such as papers on Question Answering and recommender systems over RDF i.e., $K\#8$.

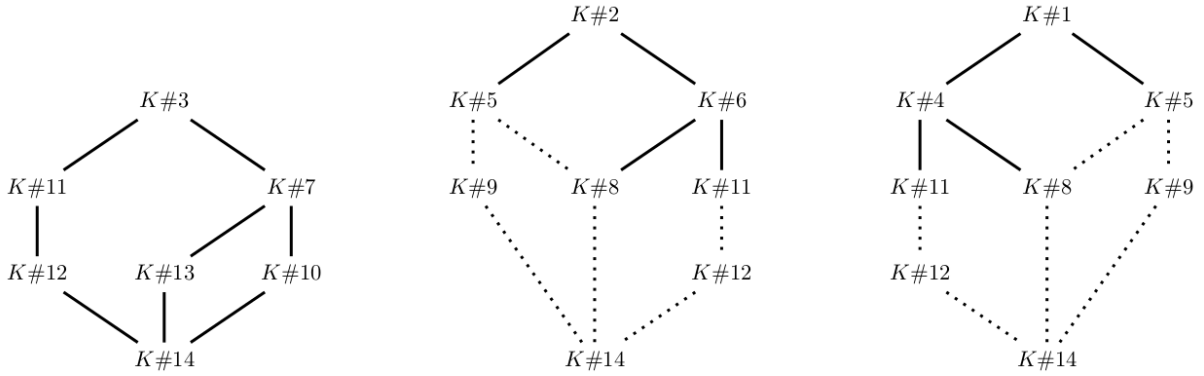


Figure 6.6: Sublattices from Figure 6.5

The obtained concept lattice keeps sub-spaces (sub-lattices) which are interpreted as the space related to some topic or author. Figure 6.6 shows three example subspaces (Note that these subspaces are not extracted, these are just drawn separately to give a clear look into what the navigation space contains).

Figure 6.6 shows the sub-space related to an author o_{22} which represents the community of the authors who work with this author. It contains 5 concepts $K\#3, K\#7, K\#10, K\#11, K\#12, K\#13$. $K\#3$ contains all the papers published by the author o_{22}, then this sub lattice can be navigated downwards to obtain specific concepts such as $K\#7$ and $K\#11$. These two concepts show co-authors of o_22 i.e., o_{23} and o_{25}. Based on the support of the concept i.e., It can also be seen that these two concepts represent the community of authors that often work together. The author o_{22} has stronger communication with the authors in this concept. However, when we move downwards in the lattice the communication becomes weaker as the number of papers published together decrease. Based on the concepts $K\#7$ and $K\#10$, recommendation can be given to author o_{25} to work with author o_{23}. This way this navigation space can be used for recommending the social collaborations of the authors working in the similar field. Navigating from $K\#3$ to $K\#11$ the papers of the author o_{22} are filtered with respect to the topic of the paper.

Now let us consider two more sub-spaces w.r.t. the topic of the paper. Figure 6.6 provide the subspaces w.r.t. the topics World Wide Web and Retrieval tasks and goals respectively. Both the subspaces can navigated from top to bottom to obtain papers on general topics and can be navigated down to get a smaller list of papers based on sub-topics or the list of authors. The important point to notice is that the dotted part in both the subspaces represent the common space of the two topics C_{12} and C_{14} meaning that this common sub-sub-space keep the papers which are common to both the topics. It also keeps the combination of different classes which benefits the user while finding the subjects which related to more than one object or class of object simultaneously. The cases indicated in the navigation and interpretation scenario above is very generic and can be used in any domain.

6.3.2 Interactive Data Exploration over Navigation Space

The navigation space obtained using the RDF-Pattern Structures serves as an interactive exploration space for the user. The user can perform interactive exploration while navigating from any of the dimensions (in our case authors and topic). Each concept act as a sample for exploration. These concepts keep classes of RDF triples as described before. Now the user can mark these samples as irrelevant. If the dimension explored by the user does not have a reference schema then all the sub-concepts of the selected concept are marked irrelevant automatically (i.e., author dimension) because the descriptions in this concept are inherited by its sub-concepts. However, if the dimension is organized w.r.t. a reference schema then all the subclasses of the class in the marked concept are also marked irrelevant. So, all the sub-concepts of the marked concept containing the classes as well as its subclasses are marked irrelevant and are hidden from the user.

Suppose that the user is not interested in the papers on the topic of Semantic Web Description Languages i.e., C_{11}. The navigation space shown in Figure 6.5 works as the exploration space, while navigating the user sees $K\#2$ which contains papers on World Wide Web i.e., C_{12}, she will mark this concept as irrelevant then the sub-concepts $K\#5, K\#6, K\#8, K\#9, K\#10$ are also marked as irrelevant because these concepts either keep the class C_{12} or a sub-class of C_{12} i.e., C_1, C_2, C_4. Consider that the user is exploring the navigation space w.r.t author dimension and she marks $K\#3$ as irrelevant because of the author o_{22} then the sub-lattice obtained by following the links from $K\#3$ until the bottom will be marked as irrelevant i.e., the concepts $K\#7, K\#10, K\#11, K\#12, K\#13$.

6.4 Experimentation

In this section we discuss the experimental results for the RDF Pattern Structures. The proposed algorithm was coded in C++ and the experiments were performed using 3GB RAM on Ubuntu version 12.04.

6.4.1 Drug Search

The datasets containing information about drugs are DrugBank, SIDER, MedDRA and MeSH. *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[39][40].

The schema associated to the side effects of the drug is available on *BioPortal* Whetzel et al. (2011), which is a web portal that provides access to a repository of biomedical ontologies. *BioPortal* is developed by National Center for Biomedical Ontology (NCBO) Musen et al. (2012). 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 used the MeSH vocabulary already present in the

[39] `http://wifo5-04.informatik.uni-mannheim.de/drugbank/snorql/`
[40] `http://wifo5-04.informatik.uni-mannheim.de/sider/snorql/`

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 6.6: Statistics of two datasets and navigation space.

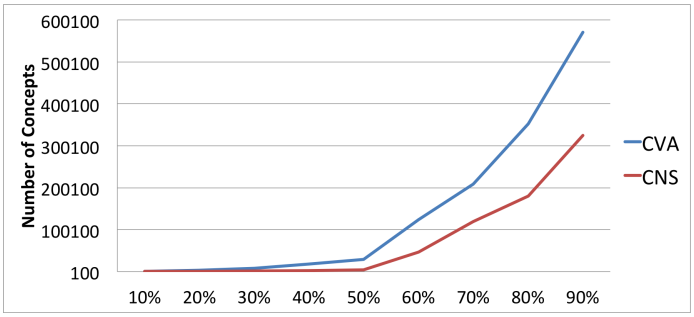


Figure 6.7: Size of the Navigation Space for each dataset.

form of RDF in Bio2RDF Belleau et al. (2008); Dumontier et al. (2014), 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). In the following, we study the scalability of *RDF-Pattern Structures* over large dataset. Table 6.6 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 6.7b 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.

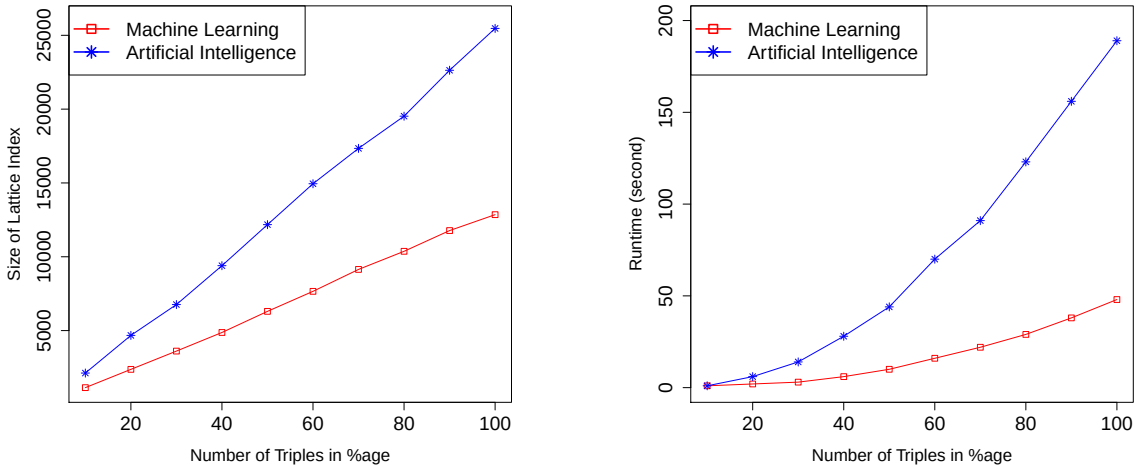
6.4.2 DBLP

The dataset used for experimentation was DBLP which keeps bibliographic information about millions of journals, conferences and authors. DBLP is converted to RDF and made available with the help of D2R server[41]. D2R server Bizer and Cyganiak (2006) provides a mapping from SQL database schema to RDF triples. However, in the current experiment the triple store used is the RDF data dump for DBLP is made available at RDF-HDT[42] Fernández et al. (2013). RDF-HDT (Header, Dictionary, Triples) is a compact data structure for RDF data which provides efficient storage by compressing big datasets. It also provides search and browse operations without prior decompression. For the experimentation two subsets of datasets were extracted from DBLP. First includes all the papers on Artificial Intelligence (AI) and the second dataset includes all the papers on Machine Learning (ML). The reference schema used for this purpose is ACM Computing Classification System (ACCS) which is available on-line in several formats.

[41] `http://dblp.l3s.de/d2r/`
[42] `http://www.rdfhdt.org/datasets/`

Datasets	No. of Triples	No. of Subjects	No. of Objects
AI	31045	9986	31140
ML	18141	5571	17633

Table 6.7: Statistics of two datasets.



(b) Variation in the size of Navigation Space. (a) Runtime for Creating the Navigation Space.

Figure 6.8: Experimental Results.

The RDF Format used for ACCS uses SKOS[43] vocabulary, an application of RDF, to define the background knowledge about the topics of the papers. For conducting the experiments, titles were considered as entities and keywords and authors were kept as descriptions. ACCS was used as a reference schema for keywords and authors did not have any reference schema.

After extracting the datasets navigation spaces are built on each of the data sets using RDF Pattern Structures. The statistics regarding both the data sets are shown in Table 6.7. The number of triples extracted for AI dataset are 31045 and for ML are 18141. Figure 6.8(a) depicts the size of the navigation spaces created for AI and ML data sets. It can be seen that the size of navigation space is suitable for exploration purposes, however the interactive data exploration will further reduce its size when the user will mark the concepts as irrelevant. For example, in AI if the user is not interested in the papers about robotics then she can choose the general class as irrelevant. This will further decrease the navigation space of the user. Figure 6.8(b) illustrates the runtime for creating the navigation space. In the current experiments we extracted the RDF data using SPARQL queries. These SPARQL queries specify the initial user and task- specific requirements and extract only small subsets of data interesting for the user. Following this line it is safe to assume that our approach is well adapted to exploratory data mining as we are using small subsets of data for exploration and visualization using a visualization tool discussed in next section. Finally, the main focus of our approach is the qualitative analysis of the data and allow user with interaction and exploration.

[43] http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/

6.4.3 Visualization

Another experiment was performed on the papers published by a Data Mining Team in a research lab. For this purpose all the papers from 2010-2014 published in international journals and conferences were selected. The papers chosen for this purpose were all in English Language. A pattern concept lattice (navigation space) was built using the paper titles, their keywords and authors. The results were visualized using the tool RV-Xplorer (Rdf View eXplorer) Alam et al. (2015d). It visualizes and allows interaction over the view defined over RDF graph through SPARQL queries by classifying SPARQL query answers in the form of a concept lattice. A dedicated web page to visualize and interact with the navigation space is available *http://webloria.loria.fr/~alammehw/rdfps/#/*.

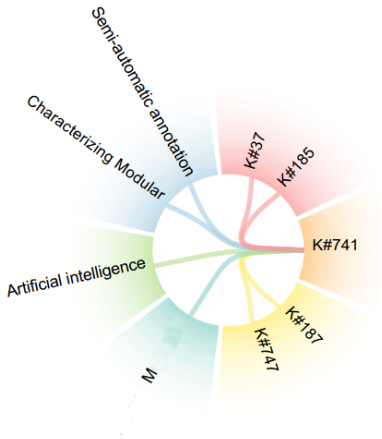


Figure 6.9: Concept

Figure 6.9 shows one concept from the graphical user interface for visualizing the resulting navigation space. The circle represents the selected concept which is the top of the concept lattice by default. It displays the contents of the selected concept i.e., the extent, intents, parent concepts and children concepts. The pink and yellow part in the selected concept ($K\#741$) show the parent ($K\#37, K\#185$) and child concepts ($K\#187, K\#747$) respectively. The green and cyan part show two different types of intent i.e., topics and authors. The blue part shows the extent of the concept i.e., the group of papers sharing some authors and topic. Now let us consider that the user chooses a concept keeping the papers about Database Management System. Figure 6.10 shows the selected concept (K#139). This visualization allows the user to navigate upwards and downwards in the navigation space to access specific as well as general information as discussed in section 6.3.1. If the user wants to navigate downwards in the lattice she should select one of the concepts in the yellow part of the selected concept which keep the sub-concepts (K#405, K#417, K#688). As the sub-concepts can be large in number we display what kind of papers the sub-concept of the lattice contains on mouse hover (see Figure 6.11). This phenomena guides the user in deciding which path to take while navigating. Now the user wants to narrow down papers about Relational database query languages, she will click on K#688 in the yellow part of the selected concept and then K#688 becomes the selected concept. This navigation is referred to as *level-wise navigation*.

The navigation space on the left hand side of the visualization shows the complete lattice to track the position of the user. The selected concept is highlighted in red color. Now if the user is on the current concept and she is interested in the papers in this concept and wants to find other

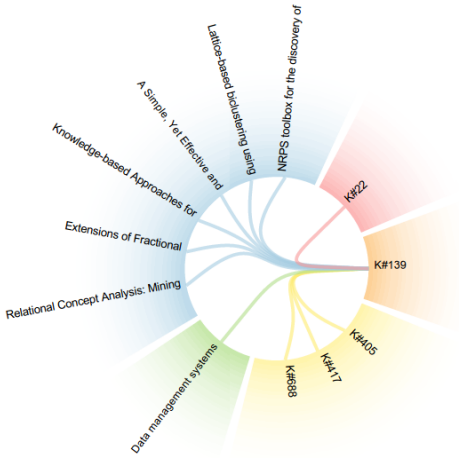


Figure 6.10: Papers on Database Management System

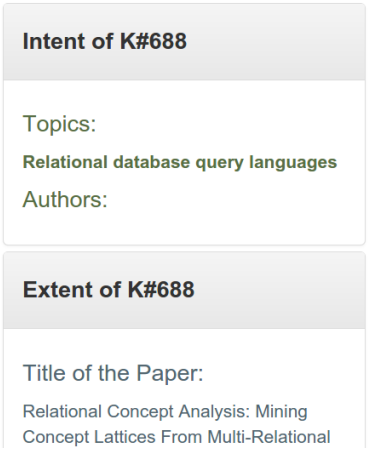


Figure 6.11: Details of Subconcept K#688

papers similarly to this paper then on mouse hover on this paper title. This will highlight all the concepts where this paper is present and the user can then directly navigate to the concept of interest. Toolkit also allows direct navigation from one concept to another without going from one hop to another. Such kind of navigation is referred to as *direct navigation*.

Finally, it helps in decreasing the navigation space of the user by enabling her to focus on only the interesting parts in the navigation space and hide the rest of the lattice (see section **??**). If the user right-clicks on a concept in the navigation space, it is marked as irrelevant to the user and is hidden from the user. Once marked irrelevant the hidden part can not be accessed unless marked relevant.

6.5 Related Work

There have been several studies which apply FCA to RDF data but to-date this is the first attempt to deal with RDF graph and pattern structures. In Ferré (2010), are based on "local views" of concept lattices, with on-the-need computation of neighbour concepts. However, in our case several RDF resources can be navigated from one platform based on user requirements.

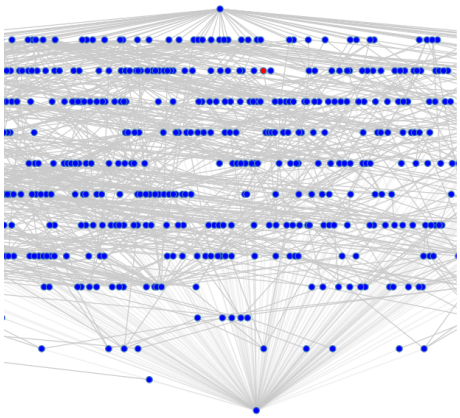


Figure 6.12: Selected Sub-concept

Hiding the non-interesting part of the concept lattice is the feature very unique to our approach. Moreover, Coulet et al. (2013) 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 to Coulet et al. (2013), RDF-Pattern Structures provide navigation space over RDF graphs as well as schema level information from several resources allowing user to access information from one platform.

In d'Amato et al. (2010), the authors focus on clustering the SPARQL query answers based on some background knowledge and provide access to these clusters using a tree structure. As a contrast our approach can directly deal with RDF graphs. Moreover, several reference schemas related to different types of objects can be used through RDF Pattern Structures. Also, our approach can handle the objects which do not have any existing reference schema.

6.6 Discussion

This chapter proposes a new approach for navigating semantic web data and targets the capabilities of Pattern Structures to deal with RDF data. It provides navigation space over RDF data by organizing RDF triples with respect to reference schema with the help of RDF Pattern Structures. The pattern concepts in the concept lattice are considered as clusters of RDF triples used for information retrieval purposes over RDF data. The proposed framework is very general and can be applied to any RDF data set having heterogeneity i.e., some of the objects containing the reference schema and some of the objects containing no schema.

Chapter 7

Conclusion and Perspectives

Contents

This thesis targets several open problems related to Semantic Web. Linked Open Data is increasing day by day and there is a need to enable the users to access this information and use it in their domain specific applications effectively. It is important for the common user to be able to effectively interact with RDF data for interpretation and information retrieval purposes. Moreover, it is necessary to consider the "subjective interestingness" of the domain expert by enabling him to provide feedback which in turn guides the system to focus only on interesting patterns. During this thesis we target the issue how KDD process can be applied to Linked Data to build user specific applications and how it can be used to extract some knowledge units. We focus on directly involving the user in the KDD process where she specifies the task and requirements according to which datasets are extracted. Afterwards, several variants of Formal Concept Analysis are applied as a data mining algorithm for generating clusters. Finally, the user was allowed to provide feedback to the system by specifying interesting information. During each study we were able to help the domain expert in obtaining the required information and limiting the navigation space to only interesting patterns. The contribution is threefold: the first part allows the user to create views over RDF graphs by clustering SPARQL query answers. With the help of a visualization tool we were able to provide user with the ability to navigate these answers. This navigation may lead to the discovery of several incompletions in the RDF data. In the second part, we use data mining techniques such as association rules to complete DBpedia data and show that our approach can be applied to RDF data from any domain and to any dataset which uses the same kind of vocabulary for building their RDF resources. During this study we move from SPARQL query answers to RDF triples because the SPARQL query answers are always limited to the resource that user wants to query and the answers may not necessarily be complete. In the third part, we defined an approach for clustering RDF triples with respect to a background knowledge in the form of taxonomy. This allows the user to navigate RDF

triples as well as the classes in the reference taxonomy scattered over several and heterogeneous resources to retrieve user specific answers.

7.1 Summary of Contributions

7.1.1 Lattice-Based View Access (LBVA)

During this study, we introduced a framework based on FCA for classifying the set of tuples obtained as a result of SPARQL queries over LOD. These classes are organized as a concept lattice referred to as a "view" built using a new clause `VIEW BY`. The non-interesting part of the answers are hidden without going back to the original querying and without recomputing the classes hence reducing the computation overhead. This view allows user interaction for information retrieval purposes and feedback for specifying only interesting concepts. Implications were also computed for identifying knowledge units. We developed a new navigational tool for concept lattices called as RV-Xplorer which provides exploration over SPARQL query answers. Interestingly, this tool is not designed for only specific purpose any kind of concept lattice can be visualized and data from any domain can be analyzed using this tool. Several experiments show that LBVA is rather tractable and can be applied to large data. During the experimentations over DBpedia, some missing information was detected. In the subsequent work, we used association rule mining for completing RDF triples in DBpedia based on incompletions observed while navigating the RDF-views.

7.1.2 Mining definitions from RDF annotations using Formal Concept Analysis

Recently, there has been a significant interest in this topic. Accordingly, several other studies are performed. In Paulheim and Bizer (2013) authors use an inference mechanism which considers the links between instances to obtain their class types, assuming that some relations occur only with certain classes. Moreover, there are some studies which focus on the correction of numerical data present in DBpedia Wienand and Paulheim (2014b) using *outlier detection method*, which identify those facts which deviate from other members of the sample. By contrast, our approach focuses on completing RDF data with the help of association rule mining. In Zaveri et al. (2013), authors propose a manual and semi-automatic methodology for evaluating the quality of LOD resources w.r.t. a taxonomy of "quality problems". Quality assessment is based on user inputs (crowd-sourcing) and measures the correctness of schema axioms in DBpedia. In Yu and Heflin (2011a,b), authors try to detect triples which are regarded as erroneous w.r.t. similar triples. The detection is based on probabilistic rule learning and on the discovery of generalized functional dependencies that are used to characterize the abnormality of the considered triples. To conclude, during this study we introduce a mechanism based on association rule mining for the completion of RDF datasets. Moreover, we use heterogeneous pattern structures to deal with heterogeneity in LOD. Several experiments have been conducted over four datasets and an evaluation was conducted for each of the experiments. This study shows the capabilities of FCA for completing complex RDF data.

7.1.3 Pattern Structures for Structured Attribute Sets

In Pattern Structures for Structured Attribute Sets, we recalled two approaches for dealing with structured attributes and proposed how we can compute intersection of antichains in tree-

shaped posets of attributes, an essential operation for working with structured attributes. Our experiments show the computational efficiency of the proposed approach. Accordingly, we are interested in applying our approach to other kinds of data such as graph data. The generalization of our approach to other kinds of posets is also of high interest. Then, we present how this approach can be applied to RDF data and also to biomedical data to answer very specific questions in biomedical domain.

This study proposes a new approach for interactively navigating RDF data instead of only SPARQL query answers and targets the capabilities of FCA to deal with RDF data. It allows navigation over RDF data by organizing RDF triples with respect to reference schema with the help of RDF Pattern Structures. To deal with such an organization, this study uses the pattern strucutres for structured attributes. The pattern concepts in the concept lattice are considered as clusters of RDF triples which enhance help the user for information retrieval purposes over RDF data. The proposed framework is very general and can be applied to any RDF data set having heterogeneity i.e., some of the objects containing the reference schema and some of the objects containing no schema. One such application is the RDF triples contained in Drugbank where the objects are considered as drugs and the attributes are side effects, their categories and target proteins. There are reference schemas regarding the side effects and categories of drugs such as MeSH and MedDRA but there is no reference schema defined for protein targeted by these drugs.

7.2 Perspectives

This thesis paved way for many perspectives. This section highlights some of the perspectives for each of the methods proposed in this thesis.

In chapter 3, we defined a framework called Lattice-Based View Access (LBVA). Currently it considers at least two variables in the SELECT clause of the SPARQL query. One of the perspectives is to consider more than one variable in the `View By` clause, which will yield more than one set of objects. This will lead to the creation of a family of contexts called as "Relational Context Family". Finally, this RCF creates a space of concept lattice which can be navigated. The problem arises how a family of concept lattices can be managed by a common user who has only a slight know-how of concept lattices. To solve this problem, a method needs to be defined where the user navigates a space of several concept lattices by only visualizing one concept lattice. Meaning that, only the target concept lattice is shown to the user but in the background several concept lattices are at work.

We also introduced a tool for navigating SPARQL query answers called `RV-Xplorer`. As discussed before it allows the user navigate SPARQL query answers from the point-of-view of attribute variable as well as object variable. As a future perspective, we want to add the functionality to the tool where it incrementally builds the concept lattice while navigation instead of navigating a pre-computed concept lattice. This can be done by using the already proposed algorithms for creating the concept lattice incrementally such as AddIntent van der Merwe et al. (2004). Another such algorithm is discussed in Guérin et al. (2013) for generating immediate successors. We also want to provide a more sophisticated approach for evaluating the user interface of RV-Xplorer. It involves asking several questions to the user which can only be answered by applying the proposed lattice navigation operations. Then, these answers need to be scored based on correctness and the time taken by the user to provide a correct answer.

In chapter 4, we discussed about the incompletion found in the RDF datasets while exploring the concept lattice obtained from the previous part of the thesis. Currently we compute

implications from the formal context and compute their confidence in the opposite direction. We use this measure to rank the implications for evaluation purposes. Different interesting perspectives are opened following this work. As we have discussed, categories represent some pre-loaded information needs in Wikipedia, i.e. a pre-answered questions whose answer is relevant for a group of people. Thus, an interesting application would be to translate these information needs into description logics definitions, instead of attributes. It is possible to think that, instead of annotating each French film with a "FrenchFilm" tag, we could define the category as $\texttt{FrenchFilm} \equiv \texttt{Film} \sqcap \texttt{hasCountry}.\{\texttt{FRANCE}\}$, or $\texttt{LamborghiniCars} \equiv \texttt{Automobile} \sqcap \texttt{manufacturedBy}.\{\texttt{LAMBORGHINI}\}$. Given that these definitions are more restrictive than typing (*rdf:type*), our work should be adapted to deal with "near-definitions" in which both directions ($X \implies Y$ and $Y \implies X$) are association rules with high confidence. Albeit, we have presented our approach applied to DBpedia, its applicability is more general than this specific knowledge resource. Another future perspectives is to perform attribute exploration over these ranked rules for knowledge base completion process Sertkaya (2010). Until now there has been no study which performs attribute exploration over pattern structures. In this case the "attribute exploration" can be renamed as "pattern exploration" and the method may vary based on the types of patterns D used in the pattern structures. Finally, we also want to provide visualization of pattern concept lattice obtained during this process. This concept lattice would be used for navigation purposes and while navigating the user should be able to perform "pattern exploration".

In chapter 5 we introduced a similarity measure between two classes and sets of classes w.r.t. some already existing taxonomy. With the help of this similarity measure the RDF triples were enriched with background knowledge. The final navigation space obtained is a concept lattice which allows simultaneous navigation and exploration over RDF triples and background knowledge. As a future perspective we want to take into account the complete schematic information such as the property and sub-property relation, relations between classes etc. In order to do so a new similarity measure needs to be introduced since least common subsumer can not handle such kind of constructs. One of the solutions would be to consider the interval between two classes as the similarity between two classes. Here, the interval refers to the path between those two classes between which the similarity is to be computed. This path consists of a chain nodes and arcs where nodes represent entities i.e., subjects and objects in RDF triple and the arc represents predicates or a properties. Moreover, in order to restrict the length of the interval considered a threshold on maximum number of hops to crawled by the algorithm can be set.

Currently, we can only consider the background knowledge for subjects and objects, however predicates are not dealt with very effectively. One of the solutions would be to use Relational Concept Analysis to deal with these predicates. But the problem encountered while dealing with relational concept analysis is that it produces two lattices for one relation. In case if we have $n-$number of predicates then it will create $n-1$ number of concept lattices. There is a need to introduce a kind of pattern structures to deal with such kind of relations i.e., instead of taking into account only two dimensions i.e., subject-predicate, it should be able to deal with the the third dimension also i.e., the predicate. This way the pattern concept lattice obtained should be search-able with respect to all three dimensions. Such kind of pattern structures can be named as "RCA-Lite" or "Relational Pattern Structures".

7.3 List of Publications

Conferences

- Mehwish Alam, Amedeo Napoli and Matthieu Osmuk: RV-Xplorer: A Way to Navigate

Lattice-Based Views over RDF Graphs. International Conference on Concept Lattice and their Applications (CLA'2015), Clermont-Ferrand, France.

- Mehwish Alam, Aleksey Buzmakov, Amedeo Napoli and Alibek Alibek Sailanbayev: Revisiting Pattern Structures for Structured Attribute Sets. International Conference on Concept Lattice and their Applications (CLA'2015), Clermont-Ferrand, France.

- Mehwish Alam and Amedeo Napoli: Exploring Distributed RDF Resources using Formal Concept Analysis. IEEE International Conference on Data Science and Advanced Analytics (IEEE DSAA'2015), Paris, France.

- Mehwish Alam, Aleksey Buzmakov, Victor Codocedo and Amedeo Napoli: An Approach for Improving RDF Data with Formal Concept Analysis. International Joint Conference Artificial Intelligence (IJCAI'2015), Buenos Aires, Argentina.

- Mehwish Alam and Amedeo Napoli: Defining Views with Formal Concept Analysis for Understanding SPARQL Query Results. International Conference on Concept Lattice and their Applications (CLA'2014), Košice, Slovakia. CEUR Proceedings Volume:1525 (http://ceur-ws.org/Vol-1525).

- Melisachew Wudage Chekol, Mehwish Alam, Amedeo Napoli: A Study on the Correspondence between FCA and ELI Ontologies. International Conference on Concept Lattice and their Applications (CLA'2013), La Rochelle, France. CEUR Proceedings Volume:1062 (http://ceur-ws.org/Vol-1062).

- Mehwish Alam, Adrien Coulet, Amedeo Napoli, Malika Smail-Tabbone: Formal Concept Analysis Applied to Transcriptomic Data. International Conference on Concept Lattice and their Applications (CLA'2012), Fuengirola (Málaga), Spain. CEUR Proceedings Volume:972 (http://ceur-ws.org/Vol-972).

Workshops

- Mehwish Alam and Amedeo Napoli: Navigating and Exploring RDF Data using Formal Concept Analysis. LD4KD 2015, co-located with PKDD 2015, Porto, Portugal (2015).

- Mehwish Alam, Aleksey Buzmakov, Víctor Codocedo and Amedeo Napoli, An Approach Towards Semantic Indexing over RDF data based on Pattern Structures. Proceedings of the 4th International Workshop "What can FCA do for Artificial Intelligence?", FCA4AI 2015, co-located with the International Joint Conference on Artificial Intelligence (IJCAI 2015 CEUR Proceedings Volume:1430 (http://ceur-ws.org/Vol-1430).

- Mehwish Alam, Amedeo Napoli, An Approach Towards Semantic Indexing over RDF data based on Pattern Structures. Proceedings of the International Workshop on Formal Concept Analysis and Applications, co-located with 13th International Conference on Formal Concept Analysis (ICFCA 2015), Nerja, Spain. CEUR Proceedings Volume:1434 (http://ceur-ws.org/Vol-1434).

- Mehwish Alam and Amedeo Napoli: Lattice-Based Views over SPARQL Query Results. LD4KD 2014, co-located with PKDD 2014, Nancy, France (2014). CEUR Proceedings Volume:1232 (http://ceur-ws.org/Vol-1232).

- Mehwish Alam and Amedeo Napoli: Lattice-Based View Access: A way to Create Views over SPARQL Query for Knowledge Discovery. In FCA4AI workshop, co-located with ECAI 2014, Prague, Czech Republic (2014). CEUR Proceedings Volume:1257 (http://ceur-ws.org/Vol-1257)
- Mehwish Alam, Melisachew Wudage Chekol, Adrien Coulet, Amedeo Napoli, Malika Smail-Tabbone: Lattice Based Data Access (LBDA): An Approach for Organizing and Accessing Linked Open Data in Biology. Data Mining for Linked Data 2013. CEUR Proceedings Volume:1082 (http://ceur-ws.org/Vol-1082).
- Mehwish Alam, Adrien Coulet, Amedeo Napoli and Malika Smail-Tabonne: Formal Concept Analysis Applied to Transcriptomic Data. In FCA4AI workshop, co-located with ECAI 2012, Montpellier, France (2012) CEUR Proceedings Volume:939 (http://ceur-ws.org/Vol-939).

Appendix

Appendix A

A Study on the Correspondence between FCA and $\mathcal{ELI}$ Ontologies

Contents

This chapter is based on Chekol et al. (2013), which was done during the stay of Melisachew Wudage Chekol in LORIA for Post-Doc. The description logic $\mathcal{EL}$ has been used to support ontology design in various domains, and especially in biology and medicine. $\mathcal{EL}$ is known for its efficient reasoning and query answering capabilities. By contrast, ontology design and query answering can be supported and guided within an FCA framework. Accordingly, in this paper, we propose a formal transformation of $\mathcal{ELI}$ (an extension of $\mathcal{EL}$ with *inverse roles*) ontologies into an FCA framework, i.e. $K_{\mathcal{ELI}}$, and we provide a formal characterization of this transformation. Then we show that SPARQL query answering over $\mathcal{ELI}$ ontologies can be reduced to lattice query answering over $K_{\mathcal{ELI}}$ concept lattices. This simplifies the query answering task and shows that some basic semantic web tasks can be improved when considered from an FCA perspective.

A.1 Introduction

Relying on Semantic Web (SW) languages and principles, several ontologies have been created in various domains, especially, in biology and medicine. In addition to that, since the conception of linked data publishing principles, over 295 linked (open) datasets have been produced[44]. Querying these data is mainly done through the W3C recommended query language SPARQL[45].

[44] `http://linkeddata.org/`
[45] `http://www.w3.org/TR/sparql11-query/`

In parallel, knowledge discovery in data represented by means of objects and their properties can be done using formal concept analysis (FCA) Ganter and Wille (1999). Concept lattices can reveal hidden relations within data and can be used for organizing and classifying data. A survey of the benefits of FCA to SW and vice versa has been proposed in Sertkaya (2010). As mentioned in that paper, a few of these benefits ranges from knowledge discovery, ontology completion, to computing subsumption hierarchy of least common subsumers. Additionally, studies in d'Aquin and Motta (2011) and Kirchberg et al. (2012) are based on FCA for managing SW data while finite models of description logics (as $\mathcal{EL}$) are explored in Baader and Distel (2008, 2009). All these studies propose methods to use FCA in the analysis of SW data. Nevertheless, none of them offer a precise way of representing SW data within a formal context. We deem it necessary to provide mathematically founded methods to formalize the representation and the analysis of SW data.

In this work, we focus particularly on $\mathcal{ELI}$ (an extension of $\mathcal{EL}$ with inverse roles) ontologies. $\mathcal{EL}$ is one of OWL 2 profiles (OWL 2 $\mathcal{EL}$). In fact, OWL 2 $\mathcal{EL}$ is used mainly for designing large biomedical ontologies such as SNOMED-CT[46], and the NCI thesaurus[47]. A common feature of these ontologies is that they possess large concept hierarchies that can be queried with SPARQL. Answering SPARQL queries is done by binding variables of the query into terms of the queried ontology. However, including inferred data in the query answers requires either a reasoner to infer all implicit data or query rewriting using regular expression patterns (that enable navigation in a hierarchy) Glimm (2011). The latter obliges the user to know the nuts and bolts of SPARQL. To overcome these difficulties, we reduce SPARQL query answering in $\mathcal{ELI}$ ontologies into query answering in concept lattices along with the transformation of the queried ontology into a formal context. Querying a concept lattice appears to be a less complex task than using SPARQ. Further, the lattice organization, i.e., partial ordering, can help understanding the relations between data and visualization of SW data.

Overall, in this paper, we work towards (i) a formal characterization of the translation of ontologies into a formal context, (ii) minimizing the difficulty of SPARQL query answering over ontologies into LQL (Lattice Query Language) query answering over concept lattices, and finally (iii) providing organization of SPARQL query answers with the use of concept lattices.

A.2 Preliminaries

In this section, we provide a very brief and intuitive introduction of the description logic $\mathcal{ELI}$ and FCA. For a detailed discussion, we refer the readers to Baader et al. (2005); Ganter and Wille (1999); **?**.

In $\mathcal{ELI}$, classes are inductively defined from a set $\mathrm{N_C}$ of *class* names, a set $\mathrm{N_R}$ of *role* names, and a set $\mathrm{N_I}$ of *individual* names ($\mathrm{N_C}$, $\mathrm{N_R}$, and $\mathrm{N_I}$ are finite), using the constructors: $\top$, $C \sqcap D$, and $\exists R.C$ are classes. Where C and D refer to classes, R refers to a role name or its inverse R^-, and in the assertion $C(a)$, a refers to an individual. In this paper, we consider $\exists R.C$ classes with $C \in \mathrm{N_C}$, i.e, C is an atomic concept in the expression of $\exists R.C$. The TBox of a $\mathcal{EL}$ knowledge base contains a set of class inclusion axioms such as $C \sqsubseteq D$. The ABox contains class and role assertions: $C(a)$ and $R(a, b)$. The semantics of $\mathcal{ELI}$ is broadly discussed in Baader et al. (2005). The semantics of $\mathcal{ELI}$-classes is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the *the interpretation function* $.^{\mathcal{I}}$ maps each class name $A \in \mathrm{N_C}$ to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $R \in \mathrm{N_R}$ to a binary relation $R^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, and

[46] `http://www.ihtsdo.org/snomed-ct/`
[47] `http://ncit.nci.nih.gov/`

each individual name $a \in \mathcal{N}_\mathcal{I}$ to an individual $a^\mathcal{I} \in \Delta^\mathcal{I}$. The extension of $.^\mathcal{I}$ to arbitrary class descriptions is defined inductively Baader et al. (2005).

SPARQL is a W3C recommended query language based on simple graph patterns. It allows variables to be bound to components in the queried graph. In addition, operators akin to relational joins, unions, left outer joins, selections, and projections can be combined to build more expressive queries. Queries are formed from query patterns which in turn are defined inductively from *path patterns*, i.e., tuple $t \in \text{UBV} \times e \times \text{UBLV}$, with V a set of variables disjoint from UBL (URIs, Blank nodes and Literals – are used to identify values such as strings, integers and dates.), and e is regular path expression. Path patterns grouped together using operators `AND` (.) and `UNION` form *query patterns*.

Definition 20. *A query pattern q is inductively defined as:*

$$\begin{aligned} q ::= &\ \text{UBV} \times e \times \text{UBLV} \mid q_1\ .\ q_2 \mid \{q_1\}\texttt{UNION}\{q_2\} \\ e ::= &\ \epsilon \mid U \mid V \mid e_1/e_2 \mid e_1 \shortmid e_2 \mid e^+ \mid e^* \end{aligned}$$

A SPARQL *SELECT* query can be formed according to the following syntax: SELECT W FROM $\mathcal{O}$ WHERE $\{q\}$. The FROM clause identifies the queried ontology $\mathcal{O}$ on which the query will be evaluated, WHERE contains a query pattern q that the query answers should satisfy and SELECT singles out the answer variables $W \in V$ from the query pattern. For this work, we consider only `AND` and `UNION` SPARQL queries.

A formal context represents data using objects, attributes, and their relationships. Formally, it is a triple $K = (G, M, I)$ where G a set of objects, M a set of attributes, and $I \subseteq G \times M$ is a relation. A derivation operator ($'$) is used to compute *formal concepts* of a context. Given a set of objects, the operator derives common attributes of these objects and vice versa. A set of formal concepts ordered with the set inclusion relation form a *concept lattice* **?**.

In the next section, we show the transformation of $\mathcal{ELI}$ ontologies into formal contexts.

A.3 Transforming $\mathcal{ELI}$ Ontologies into Formal Contexts

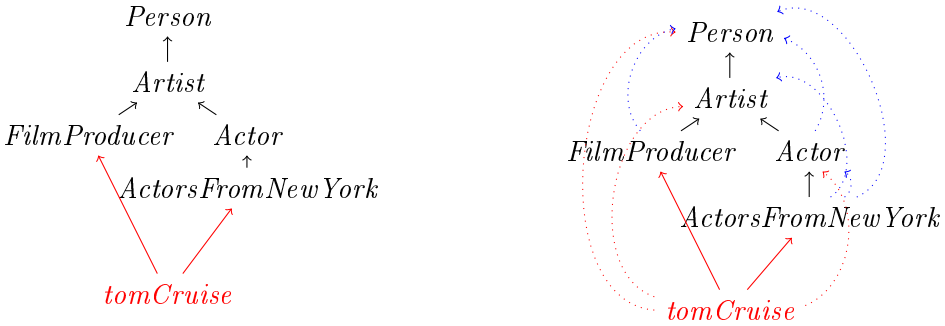
In the following, we introduce some terms and notions that we use. *Materialization* (closure) refers to computing the deductive closure of an ontology (alternatively, making all implicitly stored data explicit by using inference) ter Horst (2005). *Ontology completion Baader et al. (2007)*–refers to computing the closure of the ontology and adding additional instances by following class inclusions in the TBox (for instance, if Actor is a subclass of Artist and the instance Tom is an Actor, add another instance who is not an Actor but is an Artist. In this case, if an instance is not known, one can use anonymous resource to identify the unknown instance). *Loss of semantics*–the transformation of an ontology into a formal context results in loss of semantics if the context mixes TBox (schema axioms) and ABox (instance) data and if the concept lattice obtained from the formal context does not maintain the class hierarchy. Before presenting how a $\mathcal{ELI}$ ontology can be transformed into a formal context, we motivate our approach with an example.

A.3.1 Motivation

Example 12. *Consider the following $\mathcal{ELI}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:*

$$\mathcal{T} = \{\texttt{ActorsFromNewYork} \sqsubseteq \texttt{Actor},\ \texttt{FilmProducer} \sqsubseteq \texttt{Artist},$$
$$\texttt{Actor} \sqsubseteq \texttt{Artist},\ \texttt{Artist} \sqsubseteq \texttt{Person}\}$$
$$\mathcal{A} = \{\texttt{tomCruise}^{\mathcal{I}} \in \texttt{ActorsFromNewYork}^{\mathcal{I}}\}$$

In order to compare graphical representations of DL ontologies and their corresponding concept lattices, we represent $\mathcal{O}$ and its respective materialization $\mathcal{O}'$ as graphs as shown below:



In the graphs, dotted edges denote inferred instance and class subsumption relations.

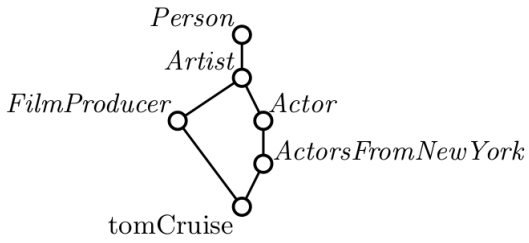
Starting with Example 12, one can ask whether it is possible to obtain a formal context from the ontology $\mathcal{O}$ while maintaining its semantics. The problem here is that DLs and FCA work on different assumptions, i.e, while DL languages are based on the *open world assumption (OWA)*, FCA relies on the *closed world assumption (CWA)*. The former permits to specify only known data whereas the later demands all data should be explicitly specified. To slightly close the gap between these two worlds:

- one can generate the formal context from the closure of the ontology. However, this approach fails when it is not possible to compute the closure of the ontology as this is the case for ontologies created from a DL language equipped with negation and disjunction constructs[48], and
- before transforming the ontology into a formal context, complete the ontology. A drawback of the second approach is that it adds unnecessary data, consequently, giving unwanted results when querying.

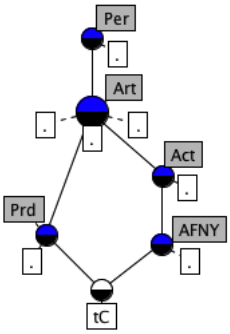
To this end, our main objective is to come up with an approach which transforms an ontology into a formal context while maintaining the semantics. Accordingly, a formal context corresponding to the ontology in Example 12 has an associated lattice that looks like the one in Figure A.1a. From this onwards, when we speak of this lattice, we refer to it as the *target* lattice. The target lattice *maintains the semantics* because: the class hierarchy of the ontology (TBox) is the same as that of the lattice, and the instance (ABox) and schema (TBox) part of the ontology are treated separately as discussed in Section A.3.2.

In the following, we provide various formal contexts associated with the ontology in Example 12. For the sake of readability, we shorten concept and individual names as: ActorsFromNewYork (AFNY), FilmProducer (Prd), Actor (Act), Artist (Art), Person (Per), and tomCruise (tC). Consider the following transformations:

[48] `http://www.w3.org/TR/owl2-primer/`



(a) Target lattice associated with the ontology in Example 12.



(b) Lattice associated with the context in Example 13.

Figure A.1: tomCruise (tC) and (.) are objects and the rest are attributes.

Naive approach: materialized ABox

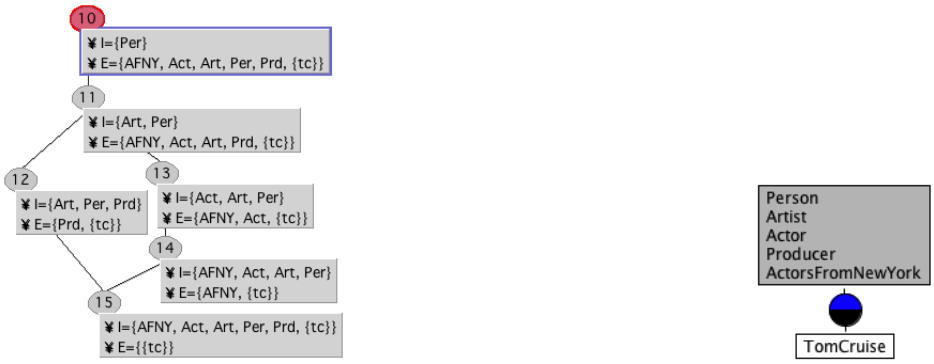
	AFNY	Prd	Act	Art	Per
tC	x	x	x	x	x

This formal context is obtained from the materialized ABox of the ontology. It does not include subclass relations as they can be acquired using *attribute exploration* Ganter and Wille (1999). But unfortunately, the resulting lattice considers all the attributes to be equivalent, implying loss of semantics, as it can be seen from the lattice in Figure A.2b.

Direct approach: materialized ABox and TBox

	{tC}	AFNY	Prd	Act	Art	Per
{tC}	x	x	x	x	x	x
AFNY		x		x	x	x
Prd			x		x	x
Act				x	x	x
Art					x	x
Per						x

This formal context is produced by taking all the subclass hierarchy of all atomic concepts C and all nominal concepts $\{a\}$ for all individuals a. Formally, a formal context is constructed using: (i) $a^{\mathcal{I}} \in C^{\mathcal{I}}$ into $\{a\}, C \in G, M$, and $(\{a\},\{a\}),(C,C),(\{a\},C) \in I$, and (ii) $C \sqsubseteq D$ into $C, D \in G, M$, and $(C,D),(C,C),(D,D) \in I$. The context is a transformation of the materialized ontology (both the closures of the ABox and TBox are computed as depicted in the right-hand graph of Example 12). The concept lattice of this formal context is shown in Figure A.2a. As it can be seen, it does not maintain the semantics because the concept hierarchy is different from that of our target lattice (in Figure A.1a). In other words, the concept hierarchy of the concept lattice is different from that of the ontology (in Example 12). Everything is mixed: attributes are also objects and vice versa. Obviously, it is possible to find several other ways of transforming an ontology into a formal context. To avoid any semantic loss, we propose an another approach, where we separately manage the transformation of ABox and TBox assertions. In FCA, attribute exploration is used to discover implicit knowledge. In that, given a concept lattice, a domain expert is asked a series of questions to produce implications that correspond to DL like inclusion axioms. Ontologies contain instance and schema data, where the latter is



(a) (b)

Figure A.2: Concept lattice associated with the ontology in Example 12.

similar to implications of concept lattices. Hence, when transforming, individuals in the ABox to become objects and concept names to become attributes, besides, assertions in the ABox are transformed into relations. Additionally, class inclusions of the TBox become background implications. The overall transformation procedure leads to a formal context with respect to existing knowledge (this is also known as *background implications* according to Ganter and Wille (1999)). This procedure is formally described in definition 21.

A.3.2 Proposal

To transform a $\mathcal{ELI}$ knowledge base KB $= \langle \mathcal{T}, \mathcal{A} \rangle$ into a formal context $K = (G, M, I)$, the schema axioms in the TBox become background implications $\mathcal{L}((G, M, I))$ and the ABox assertions become objects, attributes and relations. To elaborate, in K, individuals in the ABox constitute objects G, class names in the ABox and TBox yield attributes in M, and ABox assertions create relations between objects and attributes $I \subseteq G \times M$. Here, we consider acyclic TBoxes so as to avoid class names becoming objects in a context.

Definition 21 (Transforming $\mathcal{ELI}$ Ontologies into Formal Contexts)**.** *We define the transformation of* KB $= \langle \mathcal{T}, \mathcal{A} \rangle$ *into a formal context* (G, M, I) *thanks to a transformation function* σ *as follows:*

- *An axiom* $C \sqsubseteq D$ *in* $\mathcal{T}$ *corresponds to an implication in* $\mathcal{L}((G, M, I))$*, i.e., the set of implications based on* (G, M, I)*:* $C \sqsubseteq D \longmapsto C \rightarrow D \in \mathcal{L}((G, M, I))$.
- *Concept expressions* C *(class name),* $\exists R.C$*, and* $\exists R^{-}.C$*, correspond respectively to attributes* C*,* $\exists R.C$*, and* $\exists R^{-}.C$ *in* M.
- *An individual* a *in* $\mathcal{A}$ *corresponds to an object* a *in* G.
- *When* a *is an instance of* C *resp.* $\exists R.C$*,* $\exists R^{-}.C$*, then* $(a, C) \in I$ *resp.* $(a, \exists R.C) \in I$*,* $(a, \exists R^{-}.C) \in I$.
- *When* a *is related to* b *through* R*, then* $(a, \exists R.\top) \in I$ *and* $(b, \exists R^{-}.\top) \in I$.

Example 13. *The translation of the ontology in Example 12 into a formal context* K *and its background implications* $\mathcal{L}$ *are shown below:*

K	$AFNY$	Prd	Act	Art	Per
tC	x				

$\mathcal{L} = \{$ AFNY $\rightarrow$ Act, Prd $\rightarrow$ Art,
Act $\rightarrow$ Art, Art $\rightarrow$ Per $\}$

Construction of concept lattices: there are several algorithms that can compute concept lattices associated with a formal context. Some of these are discussed in the literature Ganter and Wille (1999); **?** and have also been implemented. They work on an empty implication base. Thus, most are not suitable for contexts with background implications. In Ganter and Wille (1999), the author provides an algorithm for attribute exploration with background implications. This technique can be employed for our purpose. As a result, the concept lattice associated with the formal context and background implications of Example 13 is depicted in Figure A.1b.

Next we show that concept lattices associated with $\mathcal{ELI}$ ontologies can be queried by LQL – lattice query language.

A.4 Querying Concept Lattice

SPARQL query answering over $\mathcal{ELI}$ ontologies can be considered as lattice query answering over $K_{\mathcal{ELI}}$ concept lattices. To do this, we need to introduce a query language for concept lattices. Each node in a lattice can be seen as a query formed by a conjunction of: a concept intent and a concept extent. Intuitively, querying concept lattices amounts to fetching the objects given a set of attributes as query constants, alternatively, fetching the attributes given a set of objects as query constants or terms. Query terms can be connected using the logical operators: AND and OR to form a complex term. A term is either a set of objects called *object term* (OT) or a set of attributes called *attribute term* (AT).

Definition 22 (Object and Attribute Terms)**.** *Given a formal context $K = (G, M, I)$, an object term (OT) and an attribute term (AT) are defined inductively as:*

$$\text{OT} = \{g\} \mid \text{OT}_1 \text{ AND } \text{OT}_2 \mid \text{OT}_1 \text{ OR } \text{OT}_2, \text{ where } g \in G$$
$$\text{AT} = \{m\} \mid \text{AT}_1 \text{ AND } \text{AT}_2 \mid \text{AT}_1 \text{ OR } \text{AT}_2, \text{ where } m \in M$$

The expression OT_1 AND OT_2 denotes the greatest lower bound (GLB) in the concept lattice $\underline{\mathcal{B}}(G, M, I)$. The expression OT_1 OR OT_2 denotes the least upper bound (LUB) in $\underline{\mathcal{B}}(G, M, I)$. Dually, the expression AT_1 AND AT_2 denotes the GLB in the concept lattice $\underline{\mathcal{B}}(G, M, I)$ (keeping the orientation of $\underline{\mathcal{B}}(G, M, I)$ based on the extents). Finally, the expression AT_1 OR AT_2 denotes the LUB in $\underline{\mathcal{B}}(G, M, I)$.

Based on the definitions of object and attribute terms, we introduce LQL queries. In this paper, we do not address the problem of negation in the query (and thus set difference).

Definition 23 (LQL - Lattice Query Language)**.** *Given an object term* OT*, an attribute term* AT*, and variables $x, y \in V$ where V is a finite set of variables, an LQL query can take the following forms:*

$$q(y) = (\text{OT}, y);\ q(x) = (x, \text{AT});\ q() = (OT, AT)$$

$q(y), q(x)$, and $q()$ do not necessarily correspond to formal concepts, only when OT and AT are closed sets. If OT is a closed set in $q(y) = (\text{OT}, y)$, then y corresponds to the intent associated with OT. The same thing happens with x when AT is a closed set in $q(x) = (x, \text{AT})$. For evaluating x and y in every possible case we do the following:

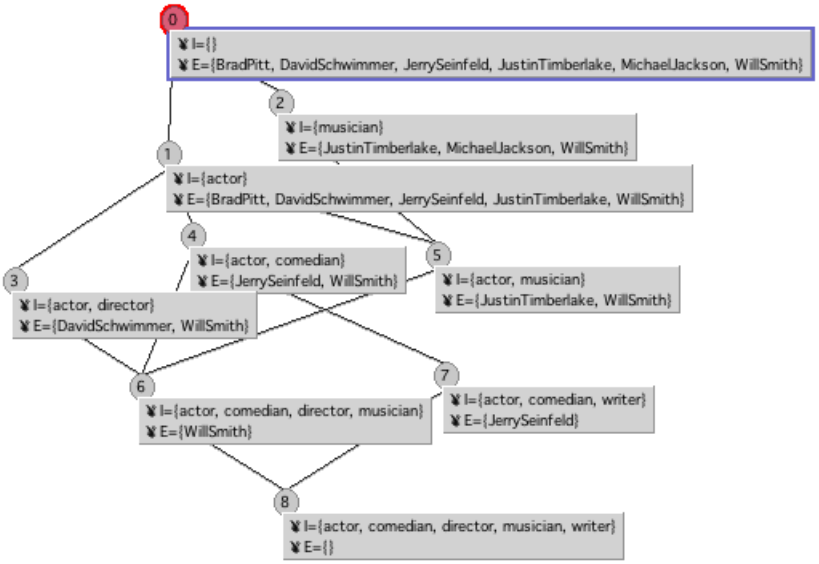


Figure A.3: A concept lattice representing artists professions.

- if OT $= \{g\}$, then $y = \{g\}'$, i.e., all attributes that are associated with the object g.
- if OT $= \{g_1\}$ AND $\{g_2\}$, then $y = \{g_1, g_2\}'$
- if OT $= \{g_1\}$ OR $\{g_2\}$, then $y = \{g_1\}' \cup \{g_2\}'$

Similarly, the evaluation of $q(x) = (x, \text{AT})$ is given as follows:

- if AT $= \{m\}$, then $x = \{m\}'$, i.e., all objects that are associated with the attribute m.
- if AT $= \{m_1\}$ AND $\{m_2\}$, then $x = \{m_1, m_2\}'$
- if AT $= \{m_1\}$ OR $\{m_2\}$, then $x = \{m_1\}' \cup \{m_2\}'$

Finally, the evaluation of $q() = (\text{OT}, \text{AT})$ is:

- *true* if OT $=$ AT$'$ or AT $=$ OT$'$ and *false* otherwise.

Example 14. *Let us consider querying the concept lattice shown in Figure A.3.*

- *For* $q_1(x) = (x, \{actor\}$ AND $\{comedian\}$ AND $\{writer\})$, *we have* $x = \{JerrySeinfeld\}$.
- $q_2(x) = (x, \{writer\}$ OR $\{director\})$, *we have* $x = \{DavidSchwimmer, WillSmith, JerrySeinfeld\}$.
- $q_3(y) = (\{JustinTimberlake\}$ AND $\{WillSmith\}, y)$, *we have* $y = \{actor, musician\}$.
- $q_4(y) = (\{DavidSchwimmer\}$ OR $\{JustinTimberlake\}, y)$, *we have* $y = \{actor, director, musician\}$.

The complexity of answering LQL queries is *polynomial* in the size of the formal context, i.e., $\mathcal{O}|(G, M, I)|$. The advantage of LQL over SPARQL is that, it allows to compute the least upper bound and greatest lower bound of query answers. We now present one important part of this work which is reducing SPARQL query answering over $\mathcal{ELI}$ ontologies into LQL query answering over $K_{\mathcal{ELI}}$ concept lattices.

A.5 SPARQL query answering over ontologies vs LQL query answering over concept lattices

Recently, SPARQL has been extended with different entailment regimes and regular path expressions[49]. The semantics of SPARQL relies on the definition of basic graph pattern matching that is built on top of simple entailment Glimm (2011). However, it may be desirable to use SPARQL to query triples entailed from subclass, subproperty, range, domain, and other relations which can be represented using DL schema languages such as $\mathcal{ELI}$. The SPARQL specification defines the results of queries based on simple entailment. The specification also presents a general parametrized definition of graph pattern matching that can be expanded to other entailments beyond simple entailment. Query answering under an entailment regime can be achieved via: (1) materialization (computing the deductive closure of the queried graph), (2) rewriting the queries using the schema, and (3) hybrid (combining materialization and query rewriting) Glimm (2011).

Example 15. *Let us consider the evaluation of the SPARQL query Q on the ontology $\mathcal{O}$ and $\mathcal{O}'$ of Example 12. Q = select all those who are artists.*

SELECT ?x WHERE {?x a Artist.}

Under simple entailment evaluation of a SPARQL query, the answers of Q over $\mathcal{O}$ is empty, i.e., $Q(\mathcal{O}) = \emptyset$. For the reason that, simple entailment is based on graph matching which requires the variable ?x in the query to be bound with a term in the graph. Since there is no term where it can be bound to, the result is empty. However, under higher entailment regimes (such as the RDFS entailment Glimm (2011)) the result of Q is non-empty because inferred instances obtained through reasoning are taken into account for computing the answers. To get a non-empty answers for the above query, one can use one of the following approaches:

1. Materialization: *involves all implicit data to be computed before the evaluation of the query. This can be done by using a DL reasoner. Consequently, in Example 12, the materialization of $\mathcal{O}$ is $\mathcal{O}'$. Thus, the evaluation of Q over $\mathcal{O}$ is $Q'(\mathcal{O})$* = {tomCruise}.

2. Query rewriting: *is the task of converting a SPARQL query into one that involves schema axioms. It can be done using SPARQL property paths (a.k.a. regular path expressions). For instance, the above query can be rewritten as:*

   SELECT ?x WHERE {?x a/$\sqsubseteq^*$ Artist.}

   This query Q' selects all instances of Artist and that of its subclasses by navigating through the subclass relation ($\sqsubseteq^*$). The rewriting can be evaluated over $\mathcal{O}$ to obtain $Q'(\mathcal{O})$ = {tomCruise}.

In summary, materialization requires a reasoner to expand the knowledge base, the complexity of this task depends on the type of the schema language. On the other hand, query rewriting requires modifying query patterns using SPARQL property paths. This also results in a further jump in the complexity of query answering.

As described above, unlike SPARQL query answering over ontologies, query answering over a concept lattice is relatively easier. Due to the fact that once the concept lattice is obtained from the ontology, LQL can be used to query the lattice. Consequently, alleviating those expensive tasks. The above SPARQL query can be converted into an LQL query as: $q(x) = (x, Artist)$.

[49] `http://www.w3.org/TR/sparql11-query/`

The evaluation of this query over a concept lattice obtained from $\mathcal{O}$ is as expected $Q'(\mathcal{O}) = \{\text{tomCruise}\}$.

The complexity of SPARQL query answering over $\mathcal{ELI}$ ontologies is larger than that of LQL query answering over $K_{\mathcal{ELI}}$ concept lattices. Since, the expressive power of SPARQL is superior than that of LQL. For $\mathcal{ELI}$ ontologies a query language like LQL is sufficient to retrieve individuals (or objects) and classes (or attributes).

A.6 Related work

To date, several studies have been carried out to assess the relevance and benefits of FCA for DL Baader et al. (2007); Baader and Distel (2008, 2009); Sertkaya (2010); d'Aquin and Motta (2011); Kirchberg et al. (2012). Notably, the work in Sertkaya (2010) presents a survey on the advantageous of FCA for DL ontologies. Accordingly, some of the benefits that FCA can bring to the DL world include: knowledge discovery, extended subsumption hierarchy (of conjunctions of concepts) Baader et al. (2003), subsumption hierarchy of least common subsumers, exploring finite models Baader and Distel (2008, 2009), role assertion analysis, supporting bottom-up construction and completion of ontologies. Since the survey, other studies, d'Aquin and Motta (2011) and Kirchberg et al. (2012), have carried out experiments to characterize and analyse SW data using FCA tools. The former provides an entry point to a linked data using questions in a way that can be navigated. It gives a translation of an RDF graph into a formal context where the subject of an RDF triple becomes the object, a composition of the predicate and object of the triple becomes an attribute. The latter obliges the user to specify objects and attributes of a context. With that, it creates SPARQL queries to extract content from linked data in order to populate the formal context. Despite the fact that all these works have employed FCA techniques, to the best of our knowledge, none of them provide a formal and precise translation of ontologies into a formal context as we did here.

A.7 Conclusion

In this work, firstly, we have proposed a formal transformation of $\mathcal{ELI}$ ontologies into formal contexts. This enables to benefit from some advantages that FCA may offer to the DL world. Then we have shown that SPARQL query answering over $\mathcal{ELI}$ ontologies can be considered as lattice query answering over $K_{\mathcal{ELI}}$ concept lattices. This alleviates some reasoning and query rewriting tasks that are required for SPARQL query answering.

Moreover, even if there already exist substantial work relating DL, semantic web and FCA, there remains a lot of research work to be carried out. Such a research work is concerned with the correspondence between concept lattices from FCA and DL-based class hierarchies, query answering and information retrieval, and scalability as well. In addition, as FCA could benefit from DL-based reasoning capabilities, semantic web and DL-driven applications can take advantage of FCA-based ontology design, data analysis and knowledge discovery capabilities of FCA. In the future, we plan to extend and experiment with the proposed approach. We will investigate how well it scales, given the size of ontologies.

Appendix B

Enriching Transcriptomic Data with Linked Open Data

Contents

This chapter discusses how Linked Open Data can be used to enrich Transcriptomic Data Alam et al. (2012a,b). Linked Open Data can be used as a source of Background Knowledge such as hierarchical information. In case of the first study, transcriptomic data was enriched with the hierarchical information from KEGG. However, LOD can also be used to enrich a list of genes with all its relevant properties and relations.

B.1 Introduction

A significant amount of Linked Open Data (LOD) is already available on the web Bizer et al. (2009a). The data sets which are published on LOD keep semantically linked structures. Then, knowledge discovery methodologies dealing with such data must be able to take into account these existing relations. Thus, there is a need to adapt knowledge discovery methods for analyzing LOD data. Besides that, life science experiments generate amounts of relational data that raise new challenges for data modeling and analysis, and make it harder for the user to select interesting features and links. These experimental datasets can be enriched with data collected from LOD, for improving data access, information retrieval and knowledge discovery. Moreover, information required by biologists is present in LOD at least for a large part, and a user must be able to search for the point of interest without following a too large set of web links, especially when a

user has no technical knowledge of semantic web Dadzie et al. (2011). Again, there is a need for designing methodologies helping a user to search and analyze web of data for solving a particular problem.

This chapter proposes an approach for enriching the biological data obtained by medical experimentations with background knowledge. During this study, the cancer genes are enriched with the properties from MSigDB which in turn is enriched by the hierarchical information about pathways i.e., KEGG hierarchy present in Linked Data as a part of Bio2RDF project.

B.2 Enriching Transcriptomic Data with Hierarchical Information from Linked Data

Over past few years, large volumes of transcriptomic data were produced but their analysis remains a challenging task because of the complexity of the biological background. In the field of transcriptomics, biologists analyze routinely the transcription or expression of genes in various situations (e.g., in tumor samples versus non-tumor samples).

Some earlier studies aimed at retrieving sets of genes sharing the same transcriptional behaviour with the help of Formal Concept Analysis (see, e.g., Kaytoue-Uberall et al. (2009), Rioult et al. (2003a), Rioult et al. (2003b)). Further studies analyze gene expression data by using gene annotations to determine whether a set of differentially expressed genes is enriched with biological attributes [Berriz et al. (2003), Doniger et al. (2003), S et al. (2004)]. Many useful resources are available online and several efforts have been made for integrating heterogeneous data Galperin and Fernández-Suarez (2012); Khatri and Draghici (2005). A recent example is of the Broad Institute where biological data were gathered from multiple resources to get thousands of predefined gene sets stored in the Molecular Signature DataBase, MSigDB [Liberzon et al. (2011)]. A predefined gene set is a set of genes known to have a specific property such as their position on the genome, their involvement in a biological process (or a molecular pathway) etc. Subsequently, given an experimental gene list as input the GSEA (Gene Set Enrichment Analysis) program is used to asses whether each predefined gene set (in the MSigDB database) is significantly present in the input list by computing an enrichment score [Subramanian et al. (2005)].

In this study, we are interested in applying knowledge discovery techniques for analyzing a differentially expressed gene set and identifying functions or pathways shared by these genes assumed to be responsible for cancer. Knowledge discovery aims at extracting relevant and useful knowledge patterns from a large amount of data. It is an interactive and iterative process involving a human (analyst or domain expert) and data sources. We show how various gene annotations and domain knowledge are integrated in a database which is then queried for building in a flexible way formal contexts. We present here a preliminary experiments using these data. It was performed on a core context with the addition of domain knowledge (by context apposition). The considered domain knowledge are the hierarchical relationships between molecular pathways. Pruning the obtained lattices allows us to retrieve interesting concepts which we discuss. The results obtained from both experiments are also compared.

B.3 Complex Biological Data Integration

In this section, we introduce and describe the biological data on which we are working.

B.3.1 Molecular Signature Database (MSigDB)

Molecular Signature Database (MSigDB) is an up-to-date database which contains data from several resources such as KEGG, BIOCARTA, REACTOME, and Amigo [Liberzon et al. (2011)]. It is a collection of 6769 predefined gene sets. A predefined gene set is a set of genes having a specific property such as their position on the genome (e.g., the genes at position chr5q12, i.e., band 12 on arm q of chromosome 5), their involvment in a biological process or a molecular pathway (e.g., the genes which are involved in the KEGG APOPTOSIS pathway)... A pathway is a series of actions among molecules in a cell that leads to a certain change in a cell. KEGG is a database storing hundreds of known pathways[50]. Besides, the MSigDB gene sets are grouped into five categories (Table B.1). For instance, all the gene sets which are defined on the basis of gene position belong to the category C1. The category C5 groups the gene sets defined on Gene Ontology (GO) terms annotating the genes (with respect to their molecular function or their housing cellular component).

For our study, we used MSigDB Version 3.0. One entry, shown below in XML format, describes the gene set corresponding to the GO term 'RNA Polymerase II Transcription Factor Activity Enhancer Binding' (all the attribute names are underlined). The *Members* attribute contains the list of gene symbols belonging to the gene set. MSigDB was chosen as the main source for describing genes because it gathers up-to-date informations about many aspects of human genes.

```
<GENESET StandardName ="RNAPolymeraseIITranscription..."
         SystematicName="M900" Organism="Homosapiens" Chip=
         "HumanGeneSymbol" CategoryCode="c5" SubCategoryCode="MF"
         Contributor="GeneOntology" Members="MYOD1,TFAP4,..."
         MembersEZID ="7023,2034,..." Status="public">
</GENESET>
```

Table B.1: Categories of MSigDB Gene Sets

Category	Description	Data Provenance
C1: Positional Gene Sets	Location of the gene on the chromosome.	Broad Institute
C2: Curated Gene Sets	Pathways	KEGG, REACTOME, BIOCARTA
C3: Motif Gene Sets	microRNAs, Transcription Factor Targets.	Broad Institute
C4: Computational Gene Sets	Cancer Modules	Broad Institute
C5: Gene Ontology (GO) Gene Sets	Biological Process, Cellular Components, Molecular Functions	AmiGO

B.3.2 Domain Knowledge

Besides the gene annotations included in MSigDB, many types of domain knowledge are interesting to use when analyzing genes. The first type of such domain knowledge are the hierarchical relationships between GO terms or between KEGG pathways. Indeed, the KEGG hierarchy for human groups the KEGG pathways into 40 categories and 6 upper level categories. Figure B.1 illustrates the KEGG hierarchy detailing on one upper-level category and one category.

[50] `http://www.genome.jp/kegg/pathway.html`

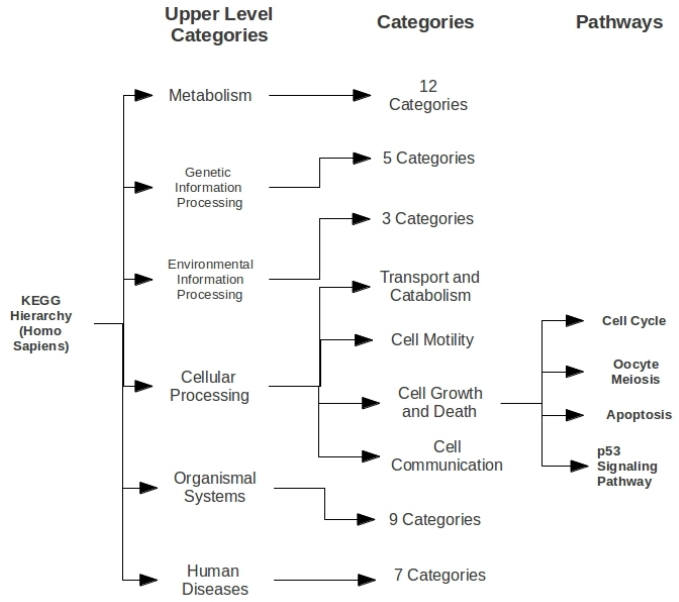


Figure B.1: Hierarchical Relationship in KEGG

In our study we have genes described by pathways involving them which may in turn be present in some category of pathways. For example, if a gene is involved in a pathway apoptosis it will also be in the category 'Cell Growth and Death'. In order to facilitate the knowledge discovery, it is important to identify the relevant data sources, organize, and integrate the data at one single database. In our case, the relevant primary data sources are MSigDB, KEGG PATHWAYS database, and AmiGO database.

B.4 From Data to Knowledge

Once the data are integrated in our database the next step is to build formal contexts for applying FCA. Our experiment focuses on applying FCA to a core context describing genes by MSigDB-based attributes and shows its extension based on the addition of domain knowledge.

B.4.1 Test Data Sets

The experiments described here are based on three published sets of genes corresponding to Cancer Modules defined in Segal et al. (2004). The authors compiled gene sets from various resources and a large collection of micro-array data related to cancers. These modules correspond to gene sets whose expression significantly change in a variety of cancer conditions (they are also defined as MSigDB gene sets in the C4 category). Our test data are composed of three lists of genes corresponding to the Cancer Modules 1 (Ovary Genes), 2 (Dorsal Root Ganglia Genes), and 5 (Lung Genes).

B.4.2 Using FCA for Analyzing Genes

We apply FCA for analyzing a context describing genes of each Cancer Module with MSigDB-based attributes. Table B.2 shows five genes (involved in Cancer Module 1) as a set of objects described by attributes which are the memberships to gene sets from MSigDB. For example,

CCT6A is in the set of genes (gene set) whose standard_name is *Reactome Serotonin Receptors*. Interestingly, by querying our integrated database the analyst is able to select the predefined gene sets to include in the formal context.

In order to extend the analysis of a list of genes, we need to take into account the domain knowledge. Hence, the same experiment was conducted with the addition of the KEGG hierarchy knowledge to the core contexts resulting in three extended contexts. All KEGG categories and upper-level categories were added as a set of attributes. If a gene is member of a KEGG pathway which in turn belongs to a category and an upper level category then a cross '×' is added in the corresponding cells in the extended context.

Table B.2 shows five genes (from Cancer Module 1) with the addition of one KEGG category (kc) and one KEGG upper level category (kuc). In the given example *CCT6A* is involved in pathway *KEGG PPAR Signaling Pathway* which belongs to the category *kc:Endocrine System* and upper level category *kuc:Organismal Systems*. The lattices were generated and the statistics for each Cancer Module are given in Table B.3. The concepts were filtered and ranked based on same criteria as in the first experiment.

Table B.2: A Toy Example of Formal Context with Domain Knowledge

Genes	TTTGCAC,MIR-19A,MIR-19B	Reactome Serotonin Receptors	KEGG PPAR Signaling Pathway	V$POU3F2_02	GO Cellular Component Assembly	chr5q12	kc:Endocrine System	kuc:Organismal Systems
BTB03	×				×	×		
PSPHL			×	×			×	×
CCT6A		×				×		
QNGPT1	×	×			×	×		
MYC	×			×				

Table B.3: Concept Lattice Statistics for the Cancer Modules with Domain Knowledge

Data Sets	No. of Genes	No. of Attributes	No. of Concepts	Levels
Module 1	361	3496	9,588	12
Module 2	378	3496	6,508	11
Module 5	419	3496	5,004	12

B.5 Results

In this study, biologists are interested in links between the input genes in terms of pathways in which they participate, relationship between genes and microRNAs etc. We obtained concepts with shared transcription factors, pathways, positions of genes and some GO terms. After the selection of concepts with higher support, we observed that there were some concepts with pathways from KEGG and REACTOME as their intent. These pathways are either related to cell

proliferation or apoptosis (cell death). The addition of domain knowledge effectively gives an opportunity to obtain the pathway categories shared by larger sets of of genes. Table B.4 shows the top-ranked concepts found in each module. For example, in module 5, we have confirmation that *Cytokine Cytokine Receptor Interaction* pathway comes under the category *Signaling Molecules and Interaction* and upper level category *Environmental Information Processing* (Concept ID 4938). The absolute support and stability of the concept containing only the category *Signaling Molecules and Interaction* and upper level category *Environmental Information Processing* as its intent are higher (Concept ID 4995, Table B.4) .

To sum up, we were able to discover interesting biological properties of subsets of genes in the three test data sets. As for example, the Focal Adhesion pathway was found to be associated to 17 genes in both modules 1 and 2; the KEGG category Immune System was found to be shared by 11 to 25 genes in the three cancer modules (Table B.4). Given the test data sets, these results are hopeful and constitute interesting positive control. This confirms that FCA-based analysis offers a powerful procedure to deeply explore sets of genes.

Table B.4: Top-ranked Concepts with Domain Knowledge

Dataset	**Concept ID**	**Intents**	**Absolute Support**	**Stability**
Module 1	**9585**	M2192:GGGAGGRR _V$MAZ_Q6	51	0.99
	9571	M2598:GO Membrane Part	27	0.99
	9566	kc:Immune System, kuc:Organismal Systems	25	0.99
	9402	chr19q13	10	0.99
	9078	M10792:KEGG MAPK Signaling Pathway, kc:Signal Transduction, kuc:Environmental Information Processing	12	0.87
Module 2	**6502**	M2192:GGGAGGRR _V$MAZ_Q6	44	0.99
	6496	kc:Immune System, kuc:Organismal Systems	15	0.99
	6388	chr6p21	10	0.97
	6335	M10792:KEGG MAPK Signaling Pathway, kc:Signal Transduction, kuc:Environmental Information Processing	11	0.89
Module 5	**5002**	kuc:Cellular Processes	48	0.99
	4995	kc:Signaling Molecules and Interaction, kuc:Environmental Information Processing	26	0.99
	4933	chr19q13	11	0.99
	4985	kc:Immune System, kuc:Organismal Systems	11	0.99
	4938	M9809:KEGG Cytokine Cytokine Receptor Interaction, kc:Signaling Molecules and Interaction, kuc:Environmental Information Processing	11	0.87

This study shows how Formal Concept Analysis can be applied to complex biological data. Data integration and FCA give the flexibility of using various types of attributes (pathways, GO terms, positions, microRNAs and Transcription Factor Targets) for analyzing a list of genes.

Our approach gives an insight into how domain knowledge can be introduced in the analysis with the help of FCA. As for future work, we plan to apply our approach to experimental gene lists and take into account gene-gene relationships (physical Protein Protein Interactions), term-term relationships (Gene Ontology relationships, namely *is-a*, *part-of*, and *regulates*) and relationships between gene positions. Moreover, in order to efficiently deal with the relationships present within the data we can use Relational Concept Analysis.

B.6 Conclusion

During this study we showed how Formal Concept Analysis can be applied to complex biological data. Data integration and FCA give the flexibility of using various types of attributes (pathways, GO terms, positions, microRNAs and Transcription Factor Targets) for analyzing a list of genes. Our approach gives an insight into how domain knowledge can be introduced in the analysis with the help of FCA. As for future work, we plan to apply our approach to experimental gene lists and take into account gene-gene relationships (physical Protein Protein Interactions), term-term relationships (Gene Ontology relationships, namely *is-a*, *part-of*, and *regulates*) and relationships between gene positions. Moreover, in order to efficiently deal with the relationships present within the data we can use Relational Concept Analysis.

Bibliography

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216.

Aït-Kaci, H., Boyer, R., Lincoln, P., and Nasr, R. (1989). Efficient Implementation of Lattice Operations. *ACM Trans. Program. Lang. Syst.*, 11(1):115–146.

Alam, M., Buzmakov, A., Codocedo, V., and Napoli, A. (2015a). Bridging dbpedia categories and dl-concept definitions using formal concept analysis. In Kuznetsov, S. O., Napoli, A., and Rudolph, S., editors, *Proceedings of the 4th International Workshop "What can FCA do for Artificial Intelligence?", FCA4AI 2015, co-located with the International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25, 2015.*, volume 1430 of *CEUR Workshop Proceedings*, pages 9–16. CEUR-WS.org.

Alam, M., Buzmakov, A., Codocedo, V., and Napoli, A. (2015b). Mining definitions from RDF annotations using formal concept analysis. In Yang, Q. and Wooldridge, M., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 823–829. AAAI Press.

Alam, M., Buzmakov, A., Napoli, A., and Sailanbayev, A. (2015c). Revisiting pattern structures for structured attribute sets. In *Proceedings of the 12th International Conference on Concept Lattices and Their Applications, Clermont-Ferrand, France, October 13-16*.

Alam, M., Coulet, A., Napoli, A., and Smaïl-Tabbone, M. (2012a). Formal concept analysis applied to transcriptomic data. In Szathmary and Priss (2012), pages 339–344.

Alam, M., Coulet, A., Napoli, A., and Smaïl-Tabbone, M. (2012b). Formal concept analysis applied to transcriptomic data. In *Proceedings International Workshop "What can FCA do for Artificial Intelligence"? co-located with the European Conference on Artificial Intelligence (ECAI 2012)) Montpellier, France, August 28, 2012.*, pages 15–22.

Alam, M. and Napoli, A. (2014a). Defining views with formal concept analysis for understanding SPARQL query results. In Bertet, K. and Rudolph, S., editors, *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Košice, Slovakia, October 7-10, 2014.*, volume 1252 of *CEUR Workshop Proceedings*, pages 255–266. CEUR-WS.org.

Alam, M. and Napoli, A. (2014b). Lattice-based view access: A way to create views over sparql query for knowledge discovery. In *Proceedings of the 3rd International Workshop "What can FCA do for Artificial Intelligence"? co-located with the European Conference on Artificial Intelligence (ECAI 2014)) Prague, Czech Republic, August 19, 2014*, pages 93–100.

Alam, M. and Napoli, A. (2015a). An approach towards semantic indexing over rdf data based on pattern structures. In *Formal Concept Analysis - 13th International Conference, ICFCA 2015, Nerja, Spain, June 23-26, 2015, Supplementary Proceedings*.

Alam, M. and Napoli, A. (2015b). Interactive exploration over RDF data using Formal Concept Analysis. In *International Conference on Data Science and Advanced Analytics, DSAA 2015, Paris, France, October 19 - October 21, 2015*. IEEE.

Alam, M. and Napoli, A. (2015c). Navigating and exploring rdf data using formal concept analysis. In *Proceedings International Workshop "Linked Data for Knowledge Discovery co-located with the ECML/PKDD 2015.*

Alam, M., Osmuk, M., and Napoli, A. (2015d). RV-Xplorer: A way to navigate lattice-based views over RDF graphs. In *Proceedings of the 12th International Conference on Concept Lattices and Their Applications, Clermont-Ferrand, France, October 13-16.*

Arenas, M., Gutierrez, C., and Pérez, J. (2009). Foundations of rdf databases. In Tessaris et al. (2009), pages 158–204.

Baader, F., Brandt, S., and Lutz, C. (2005). Pushing the EL envelope. In Kaelbling, L. P. and Saffiotti, A., editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 364–369. Professional Book Center.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baader, F. and Distel, F. (2008). A finite basis for the set of el-implications holding in a finite model. In Medina and Obiedkov (2008), pages 46–61.

Baader, F. and Distel, F. (2009). Exploring finite models in the description logic. In Ferré and Rudolph (2009), pages 146–161.

Baader, F., Ganter, B., Sertkaya, B., and Sattler, U. (2007). Completing description logic knowledge bases using formal concept analysis. In Veloso, M. M., editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 230–235.

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Becker, P., Hereth, J., and Stumme, G. (2002). ToscanaJ: An open source tool for qualitative data analysis. In Duquenne, V., Ganter, B., Liquiere, M., Nguifo, E. M., and Stumme, G., editors, *Advances in Formal Concept Analysis for Knowledge Discovery in Databases.*, pages 1–2, Lyon, France.

Belleau, F., Nolin, M., Tourigny, N., Rigault, P., and Morissette, J. (2008). Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):706–716.

Bender, M. A., Farach-Colton, M., Pemmasani, G., Skiena, S., and Sumazin, P. (2005). Lowest common ancestors in trees and DAGs. *J. Algorithms*, 57(2):75–94.

Benz, D., Hotho, A., and Stumme, G. (2010). Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge. In *Proceedings of the 2nd Web Science Conference*.

Berriz, G. F., King, O. D., Bryant, B., Sander, C., and Roth, F. P. (2003). Characterizing gene sets with FuncAssociate. *Bioinformatics*, 19(18):2502–2504.

Bie, T. D. (2011). An information theoretic framework for data mining. In Apté, C., Ghosh, J., and Smyth, P., editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 564–572. ACM.

Bizer, C. and Cyganiak, R. (2006). D2r server - publishing relational databases on the semantic web. Poster at the 5th International Semantic Web Conference.

Bizer, C., Heath, T., and Berners-Lee, T. (2009a). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009b). Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009c). DBpedia - A Crystallization Point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165.

Borchmann, D. (2012). A generalized next-closure algorithm - enumerating semilattice elements from a generating set. In Szathmary and Priss (2012), pages 9–20.

Boulicaut, J., Esposito, F., Giannotti, F., and Pedreschi, D., editors (2004). *Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, September 20-24, 2004, Proceedings*, volume 3202 of *Lecture Notes in Computer Science*. Springer.

Buzmakov, A., Egho, E., Jay, N., Kuznetsov, S. O., Napoli, A., and Raïssi, C. (2013). On projections of sequential pattern structures (with an application on care trajectories). In Ojeda-Aciego and Outrata (2013), pages 199–208.

Carpineto, C., Osiński, S., Romano, G., and Weiss, D. (2009). A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38.

Carpineto, C. and Romano, G. (1995). ULYSSES: A lattice-based multiple interaction strategy retrieval interface. In Blumenthal, B., Gornostaev, J., and Unger, C., editors, *Human-Computer Interaction, 5th International Conference, EWHCI '95, Moscow, Russia, July 3-7, 1995, Selected Papers*, volume 1015 of *Lecture Notes in Computer Science*, pages 91–104. Springer.

Carpineto, C. and Romano, G. (1996a). Information retrieval through hybrid navigation of lattice representations. *Int. J. Hum.-Comput. Stud.*, 45(5):553–578.

Carpineto, C. and Romano, G. (1996b). A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122.

Carpineto, C. and Romano, G. (2004a). *Concept Data Analysis: Theory and Applications*. John Wiley & Sons.

Carpineto, C. and Romano, G. (2004b). Exploiting the potential of concept lattices for information retrieval with CREDO. *J. UCS*, 10(8):985–1013.

Caspard, N., Leclerc, B., and Monjardet, B. (2012). *Finite Ordered Sets*. Cambridge University Press, Cambridge, UK. First published in French as "Ensembles ordonnés finis : concepts, résultats et usages", Springer 2009.

Chekol, M. W., Alam, M., and Napoli, A. (2013). A study on the correspondence between FCA and ELI ontologies. In Ojeda-Aciego and Outrata (2013), pages 237–248.

Cigarrán, J. M., Gonzalo, J., Peñas, A., and Verdejo, F. (2004). Browsing search results via formal concept analysis: Automatic selection of attributes. In Eklund (2004), pages 74–87.

Cigarrán, J. M., Peñas, A., Gonzalo, J., and Verdejo, F. (2005). Evaluating hierarchical clustering of search results. In Consens, M. P. and Navarro, G., editors, *String Processing and Information Retrieval, 12th International Conference, SPIRE 2005, Buenos Aires, Argentina, November 2-4, 2005, Proceedings*, volume 3772 of *Lecture Notes in Computer Science*, pages 49–54. Springer.

Codocedo, V., Lykourentzou, I., and Napoli, A. (2014). A semantic approach to concept lattice-based information retrieval. *Ann. Math. Artif. Intell.*, 72(1-2):169–195.

Codocedo, V. and Napoli, A. (2014). A Proposition for Combining Pattern Structures and Relational Concept Analysis. In *12th International Conference on Formal Concept Analysis*.

Cole, R. and Stumme, G. (2000). CEM - A conceptual email manager. In Ganter and Mineau (2000), pages 438–452.

Cole, R. J. and Eklund, P. W. (1999). Analyzing an email collection using formal concept analysis. In Zytkow, J. M. and Rauch, J., editors, *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Prague, Czech Republic, September 15-18, 1999, Proceedings*, volume 1704 of *Lecture Notes in Computer Science*, pages 309–315. Springer.

Coulet, A., Domenach, F., Kaytoue, M., and Napoli, A. (2013). Using pattern structures for analyzing ontology-based annotations of biomedical data. In *Formal Concept Analysis, 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings*, pages 76–91.

Cutting, D. R., Karger, D. R., and Pedersen, J. O. (1993). Constant interaction-time scatter/-gather browsing of very large document collections. In Korfhage, R., Rasmussen, E. M., and Willett, P., editors, *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993*, pages 126–134. ACM.

Cutting, D. R., Pedersen, J. O., Karger, D. R., and Tukey, J. W. (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In Belkin, N. J., Ingwersen, P., and Pejtersen, A. M., editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*, pages 318–329. ACM.

Dadzie, A.-S., Rowe, M., and Petrelli, D. (2011). *Hide the Stack:* toward usable linked data. In Antoniou, G., Grobelnik, M., Simperl, E. P. B., Parsia, B., Plexousakis, D., Leenheer, P. D., and Pan, J. Z., editors, *ESWC (1)*, volume 6643 of *Lecture Notes in Computer Science*, pages 93–107. Springer.

d'Amato, C., Fanizzi, N., and Lawrynowicz, A. (2010). Categorize by: Deductive aggregation of semantic web query results. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., and Tudorache, T., editors, *ESWC (1)*, volume 6088 of *Lecture Notes in Computer Science*, pages 91–105. Springer.

d'Aquin, M. and Motta, E. (2011). Extracting relevant questions to an rdf dataset using formal concept analysis. In *K-CAP*, pages 121–128. ACM.

Dau, F., Ducrou, J., and Eklund, P. W. (2008). Concept similarity and related categories in searchsleuth. In Eklund, P. W. and Haemmerlé, O., editors, *Conceptual Structures: Knowledge Visualization and Reasoning, 16th International Conference on Conceptual Structures, ICCS 2008, Toulouse, France, July 7-11, 2008, Proceedings*, volume 5113 of *Lecture Notes in Computer Science*, pages 255–268. Springer.

Delugach, H. S. and Stumme, G., editors (2001). *Conceptual Structures: Broadening the Base, 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, July 30-August 3, 2001, Proceedings*, volume 2120 of *Lecture Notes in Computer Science*. Springer.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., and Ghani, R., editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610. ACM.

Doniger, S., Salomonis, N., Dahlquist, K., Vranizan, K., Lawlor, S., and Conklin, B. (2003). MAPPFinder: using Gene Ontology and GenMAPP to Create a Global Gene-expression Profile from Microarray Data. *Genome Biology*, 4(1):R7.

Ducrou, J. (2007). Dvdsleuth: A case study in applied formal concept analysis for navigating web catalogs. In Priss, U., Polovina, S., and Hill, R., editors, *ICCS*, volume 4604 of *Lecture Notes in Computer Science*, pages 496–500. Springer.

Ducrou, J., Vormbrock, B., and Eklund, P. W. (2006). Fca-based browsing and searching of a collection of images. In Schärfe, H., Hitzler, P., and Øhrstrøm, P., editors, *Conceptual Structures: Inspiration and Application, 14th International Conference on Conceptual Structures, ICCS 2006, Aalborg, Denmark, July 16-21, 2006, Proceedings*, volume 4068 of *Lecture Notes in Computer Science*, pages 203–214. Springer.

Dumontier, M., Callahan, A., Cruz-Toledo, J., Ansell, P., Emonet, V., Belleau, F., and Droit, A. (2014). Bio2rdf release 3: A larger, more connected network of linked data for the life sciences. In Horridge et al. (2014).

Eklund, P. W., editor (2004). *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*, Lecture Notes in Computer Science. Springer.

Eklund, P. W., Ducrou, J., and Brawn, P. (2004). Concept lattices for information visualization: Can novices read line-diagrams? In Eklund (2004), pages 57–73.

Fayyad, U., Piatetsky-shapiro, G., and Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996b). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., and Arias, M. (2013). Binary RDF representation for publication and exchange (HDT). *J. Web Sem.*, 19:22–41.

Ferragina, P. and Gulli, A. (2004a). The anatomy of snaket: A hierarchical clustering engine for web-page snippets. In Boulicaut et al. (2004), pages 506–508.

Ferragina, P. and Gulli, A. (2004b). Experimenting snaket: A hierarchical clustering engine for web-page snippets. In Boulicaut et al. (2004), pages 543–545.

Ferré, S. (2009). Camelis: a logical information system to organise and browse a collection of documents. *Int. J. General Systems*, 38(4):379–403.

Ferré, S. (2010). 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.

Ferré, S. (2014a). Expressive and scalable query-based faceted search over SPARQL endpoints. In Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C. A., Vrandecic, D., Groth, P. T., Noy, N. F., Janowicz, K., and Goble, C. A., editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, volume 8797 of *Lecture Notes in Computer Science*, pages 438–453. Springer.

Ferré, S. (2014b). Reconciling expressivity and usability in information access - from file systems to the semantic web. In *Habilitation Thesis, University of Rennes 1, Rennes, France.*

Ferré, S. (2014c). SPARKLIS: a SPARQL endpoint explorer for expressive question answering. In *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014.*, pages 45–48.

Ferré, S. and Ridoux, O. (2000). A logical generalization of formal concept analysis. In Ganter and Mineau (2000), pages 371–384.

Ferré, S. and Rudolph, S., editors (2009). *Formal Concept Analysis, 7th International Conference, ICFCA 2009, Darmstadt, Germany, May 21-24, 2009, Proceedings*, volume 5548 of *Lecture Notes in Computer Science*. Springer.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172.

Fürber, C. and Hepp, M. (2011). Swiqa - a semantic web information quality assessment framework. In *19th European Conference on Information Systems.*

Gabow, H. N., Bentley, J. L., and Tarjan, R. E. (1984). Scaling and Related Techniques for Geometry Problems. In *Proc. Sixt. Annu. ACM Symp. Theory Comput.*, STOC '84, pages 135–143, New York, NY, USA. ACM.

Galperin, M. Y. and Fernández-Suarez, X. M. (2012). The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection. *Nucleic Acids Research*, 40(Database-Issue):1–8.

Ganter, B. and Kuznetsov, S. O. (2001a). Pattern structures and their projections. In Delugach and Stumme (2001), pages 129–142.

Ganter, B. and Kuznetsov, S. O. (2001b). Pattern structures and their projections. In Delugach and Stumme (2001), pages 129–142.

Ganter, B. and Mineau, G. W., editors (2000). *Conceptual Structures: Logical, Linguistic, and Computational Issues, 8th International Conference on Conceptual Structures, ICCS 2000, Darmstadt, Germany, August 14-18, 2000, Proceedings*, volume 1867 of *Lecture Notes in Computer Science*. Springer.

Ganter, B. and Wille, R. (1999). *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg.

Glimm, B. (2011). Using SPARQL with RDFS and OWL entailment. In Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., and Patel-Schneider, P. F., editors, *Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures*, volume 6848 of *Lecture Notes in Computer Science*, pages 137–201. Springer.

Guérin, C., Bertet, K., and Revel, A. (2013). An efficient java implementation of the immediate successors calculation. In Ojeda-Aciego and Outrata (2013), pages 81–92.

Guigues, J.-L. and Duquenne, V. (1986). Familles minimales d'implications informatives resultant d'un tablaeu de donnees binaires. *Mathématiques et Sciences Humaines*, 95:5–18.

Hacene, M. R., Huchard, M., Napoli, A., and Valtchev, P. (2007). A proposal for combining formal concept analysis and description logics for mining relational data. In Kuznetsov, S. O. and Schmidt, S., editors, *ICFCA*, volume 4390 of *Lecture Notes in Computer Science*, pages 51–65. Springer.

Hartig, O. and Langegger, A. (2010). A database perspective on consuming linked data on the web. *Datenbank-Spektrum*, 10(2):57–66.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers.

Horridge, M., Rospocher, M., and van Ossenbruggen, J., editors (2014). *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014*, volume 1272 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Hotho, A., Hotho, A., Staab, S., Staab, S., Stumme, G., and Stumme, G. (2003). Text clustering based on background knowledge.

Isele, R., Umbrich, J., Bizer, C., and Harth, A. (2010). Ldspider: An open-source crawling framework for the web of linked data. In Polleres, A. and Chen, H., editors, *Proceedings of the ISWC 2010 Posters & Demonstrations Track*, volume 658 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Kaytoue, M., Kuznetsov, S. O., and Napoli, A. (2011a). Revisiting numerical pattern mining with formal concept analysis. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence.*, pages 1342–1347.

Kaytoue, M., Kuznetsov, S. O., and Napoli, A. (2011b). Revisiting numerical pattern mining with formal concept analysis. *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, pages 1342–1347.

Kaytoue-Uberall, M., Duplessis, S., Kuznetsov, S. O., and Napoli, A. (2009). Two FCA-Based Methods for Mining Gene Expression Data. In Ferré and Rudolph (2009), pages 251–266.

Khatri, P. and Draghici, S. (2005). Ontological Analysis of Gene Expression Data: Current Tools, Limitations, and Open Problems. *Bioinformatics*, 21(18):3587–3595.

Kirchberg, M., Leonardi, E., Tan, Y. S., Link, S., Ko, R. K. L., and Lee, B.-S. (2012). Formal concept discovery in semantic web data. In Domenach, F., Ignatov, D. I., and Poelmans, J., editors, *ICFCA*, volume 7278 of *Lecture Notes in Computer Science*, pages 164–179. Springer.

Koester, B. (2005). Conceptual knowledge processing with google. In Bauer, M., Brandherm, B., Fürnkranz, J., Grieser, G., Hotho, A., Jedlitschka, A., and Kröner, A., editors, *Lernen, Wissensentdeckung und Adaptivität (LWA) 2005, GI Workshops, Saarbrücken, October 10th-12th, 2005*, pages 178–183. DFKI.

Koester, B. (2006). Conceptual knowledge retrieval with fooca: Improving web search engine results with contexts and concept hierarchies. In Perner, P., editor, *Advances in Data Mining, Applications in Medicine, Web Mining, Marketing, Image and Signal Mining, 6th Industrial Conference on Data Mining, ICDM 2006, Leipzig, Germany, July 14-15, 2006, Proceedings*, volume 4065 of *Lecture Notes in Computer Science*, pages 176–190. Springer.

Krajca, P., Outrata, J., and Vychodil, V. (2010). Advances in algorithms based on cbo. In Kryszkiewicz and Obiedkov (2010), pages 325–337.

Kryszkiewicz, M. and Obiedkov, S. A., editors (2010). *Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, October 19-21, 2010*, volume 672 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Kuznetsov, S. O. (2007). On stability of a Formal Concept. *Ann. Math. Artif. Intell.*, 49(1-4):101–115.

Lawrynowicz, A. (2009a). Grouping results of queries to ontological knowledge bases by conceptual clustering. In Nguyen, N. T., Kowalczyk, R., and Chen, S.-M., editors, *ICCCI*, volume 5796 of *Lecture Notes in Computer Science*, pages 504–515. Springer.

Lawrynowicz, A. (2009b). Query results clustering by extending sparql with cluster by. In Meersman, R., Herrero, P., and Dillon, T. S., editors, *OTM Workshops*, volume 5872 of *Lecture Notes in Computer Science*, pages 826–835. Springer.

Lawrynowicz, A., d'Amato, C., and Fanizzi, N. (2010). A refinement operator based method for semantic grouping of conjunctive query results. In Setchi, R., Jordanov, I., Howlett, R. J., and Jain, L. C., editors, *KES (3)*, volume 6278 of *Lecture Notes in Computer Science*, pages 359–368. Springer.

Lawrynowicz, A., Potoniec, J., Konieczny, L., Madziar, M., Nowak, A., and Pawlak, K. T. (2011). Asparagus - a system for automatic sparql query results aggregation using semantics. In Jedrzejowicz, P., Nguyen, N. T., and Hoang, K., editors, *ICCCI (1)*, volume 6922 of *Lecture Notes in Computer Science*, pages 304–313. Springer.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2014). DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Liberzon, A., Subramanian, A., Pinchback, R., Thorvaldsdóttir, H., Tamayo, P., and Mesirov, J. P. (2011). Molecular signatures database (msigdb) 3.0. *Bioinformatics*, 27(12):1739–1740.

Manning, C. D., Raghavan, P., and Schtze, H. (2008). *Introduction to Information Retrieval*.

Medina, R. and Obiedkov, S. A., editors (2008). *Formal Concept Analysis, 6th International Conference, ICFCA 2008, Montreal, Canada, February 25-28, 2008, Proceedings*, volume 4933 of *Lecture Notes in Computer Science*. Springer.

Messai, N., Devignes, M., Napoli, A., and Smaïl-Tabbone, M. (2010). Using domain knowledge to guide lattice-based complex data exploration. In Coelho, H., Studer, R., and Wooldridge, M., editors, *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 847–852. IOS Press.

Messai, N., Devignes, M.-D., Napoli, A., and Smaïl-Tabbone, M. (2006). BR-Explorer: A sound and complete FCA-based retrieval algorithm. In *Proceedings of the Fourth International Conference on Formal Concept Analysis, Dresden, Germany, February 13-17, 2006*.

Musen, M. A., Noy, N. F., Shah, N. H., Whetzel, P. L., Chute, C. G., Storey, M. D., and Smith, B. (2012). The national center for biomedical ontology. *JAMIA*, 19(2):190–195.

Nauer, E. and Toussaint, Y. (2007). Dynamical modification of context for an iterative and interactive information retrieval process on the web. In Eklund, P. W., Diatta, J., and Liquiere, M., editors, *Proceedings of the Fifth International Conference on Concept Lattices and Their Applications, CLA 2007, Montpellier, France, October 24-26, 2007*, volume 331 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Ojeda-Aciego, M. and Outrata, J., editors (2013). *Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013*, volume 1062 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., and Tummarello, G. (2008). Sindice.com: a document-oriented lookup index for open linked data. *IJMSO*, 3(1):37–52.

Osinski, S. and Weiss, D. (2005). Carrot^2: Design of a flexible and efficient web information retrieval framework. In Szczepaniak, P. S., Kacprzyk, J., and Niewiadomski, A., editors, *Advances in Web Intelligence Third International Atlantic Web IntelligenceConference, AWIC*

2005, Lodz, Poland, June 6-9, 2005, Proceedings, volume 3528 of *Lecture Notes in Computer Science*, pages 439–444. Springer.

Paulheim, H. and Bizer, C. (2013). Type inference on noisy rdf data. In *12th International Semantic Web Conference*.

Pichon, E., Lenca, P., Guillet, F., and Wang, J. W. (1994). Un algorithme de partition d'un produit direct d'ordres totaux en un nombre fini de chaînes. *Mathématiques, Informatique et Sciences Humaines*, 125:5–15.

Rioult, F., Boulicaut, J.-F., Crémilleux, B., and Besson, J. (2003a). Using Transposition for Pattern Discovery from Microarray Data. In *DMKD*, pages 73–79.

Rioult, F., Robardet, C., Blachon, S., Crémilleux, B., G, O., and Boulicaut, J.-F. (2003b). Mining Concepts from Large SAGE Gene Expression Matrices. In *In: Proceedings KDID'03 co-located with ECML-PKDD 2003, Catvat-Dubrovnik (Croatia*, pages 107–118.

Rouane-Hacene, M., Huchard, M., Napoli, A., and Valtchev, P. (2013). Relational Concept Analysis: Mining Concept Lattices From Multi-Relational Data. *Annals of Mathematics and Artificial Intelligence*, 67(1):81–108.

S, Z., F, S., O, L., MJ, K., C, W., and WH, W. (2004). GoSurfer: a graphical interactive tool for comparative analysis of large gene sets in Gene Ontology space. *Applied Bioinformatics*, 3(4):1–5.

Salamon, G. and Wiener, G. (2008). On finding spanning trees with few leaves. *Inf. Process. Lett.*, 105(5):164–169.

Segal, E., Friedman, N., Koller, D., and Regev, A. (2004). A Module Map Showing Conditional Activity of Expression Modules in Cancer. *Nat.Genet.*, 36:1090–8.

Sertkaya, B. (2010). A survey on how description logic ontologies benefit from FCA. In Kryszkiewicz and Obiedkov (2010), pages 2–21.

Staab, S. and Studer, R., editors (2009). *Handbook on Ontologies (Second Edition)*. Springer, Berlin.

Stefanowski, J. and Weiss, D. (2003). Carrot and language properties in web search results clustering. In Ruiz, E. M., Segovia, J., and Szczepaniak, P. S., editors, *Web Intelligence, First International Atlantic Web Intelligence Conference, AWIC 2003, Madrid, Spain, May 5-6, 2003, Proceedings*, volume 2663 of *Lecture Notes in Computer Science*, pages 240–249. Springer.

Stumme, G., Taouil, R., Bastide, Y., and Lakhal, L. (2001). Conceptual clustering with iceberg concept lattices. In Klinkenberg, R., Rüping, S., Fick, A., Henze, N., Herzog, C., Molitor, R., and Schröder, O., editors, *Proc. GI-Fachgruppentreffen Maschinelles Lernen (FGML'01)*, Universität Dortmund 763.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., and Mesirov, J. P. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A*, 102(43):15545–15550.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. ACM.

Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., and Oberle, D. (2005). The swrc ontology - semantic web for research communities. In *EPIA*, volume 3808 of *Lecture Notes in Computer Science*, pages 218–231. Springer.

Szathmary, L. and Priss, U., editors (2012). *Proceedings of The Ninth International Conference on Concept Lattices and Their Applications, Fuengirola (Málaga), Spain, October 11-14, 2012*, volume 972 of *CEUR Workshop Proceedings*. CEUR-WS.org.

ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *J. Web Sem.*, 3(2-3):79–115.

Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., and Schmidt, R. A., editors (2009). *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, volume 5689 of *Lecture Notes in Computer Science*. Springer.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.

van der Merwe, D., Obiedkov, S. A., and Kourie, D. G. (2004). Addintent: A new incremental algorithm for constructing concept lattices. In Eklund (2004), pages 372–385.

van Leeuwen, M. (2014). Interactive data exploration using pattern mining. In Holzinger, A. and Jurisica, I., editors, *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics - State-of-the-Art and Future Challenges*, volume 8401 of *Lecture Notes in Computer Science*, pages 169–182. Springer.

Weiner, P. (1973). Linear pattern matching algorithms. In *Proceedings of the 14th Annual Symposium on Switching and Automata Theory (Swat 1973)*, SWAT '73, pages 1–11, Washington, DC, USA. IEEE Computer Society.

Whetzel, P. L., Noy, N. F., Shah, N. H., Alexander, P. R., Nyulas, C., Tudorache, T., and Musen, M. A. (2011). 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.

Wienand, D. and Paulheim, H. (2014a). Detecting incorrect numerical data in dbpedia. In Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., and Tordai, A., editors, *ESWC*, volume 8465 of *Lecture Notes in Computer Science*, pages 504–518. Springer.

Wienand, D. and Paulheim, H. (2014b). Detecting incorrect numerical data in dbpedia. In *11th Extended Semantic Web Conference*.

Wray, T. and Eklund, P. (2014). Using formal concept analysis to create pathways through museum collections. In *Proceedings of the 3rd International Workshop "What can FCA do for Artificial Intelligence"? co-located with the European Conference on Artificial Intelligence (ECAI 2014)) Prague, Czech Republic, August 19, 2014*, pages 9–16.

Wray, T., Eklund, P. W., and Kautz, K. (2013). Pathways through information landscapes: Alternative design criteria for digital art collections. In Baskerville, R. and Chau, M., editors, *Proceedings of the International Conference on Information Systems, ICIS 2013, Milano, Italy, December 15-18, 2013*. Association for Information Systems.

Yu, Y. and Heflin, J. (2011a). Detecting abnormal data for ontology based information integration. In *2011 International Conference on Collaboration Technologies and Systems*.

Yu, Y. and Heflin, J. (2011b). Extending functional dependency to detect abnormal data in RDF graphs. In *10th International Semantic Web Conference*.

Zamir, O. and Etzioni, O. (1999). Grouper: A dynamic clustering interface to web search results. *Computer Networks*, 31(11-16):1361–1374.

Zaveri, A., Kontokostas, D., Sherif, M. A., Bühmann, L., Morsey, M., Auer, S., and Lehmann, J. (2013). User-driven quality evaluation of dbpedia. In *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems*.

Résumé

Récemment, le " Web des documents " est devenu le " Web des données ", *i.e*, les documents sont annotés sous forme de triplets RDF. Ceci permet de transformer des données traitables uniquement par les humains en données compréhensibles par les machines. Ces données peuvent désormais être explorées par l'utilisateur par le biais de requêtes SPARQL. Par analogie avec les moteurs de clustering web qui fournissent des classifications des résultats obtenus à partir de l'interrogation du web des documents, il est également nécessaire de refléchir à un cadre qui permette la classification des réponses aux requêtes SPARQL pour donner un sens aux données retrouvées. *La fouille exploratoire des données* se concentre sur l'établissement d'un aperçu de ces données. Elle permet également le filtrage des données non-intéressantes grâce à l'implication directe des experts du domaine dans le processus. La contribution de cette thèse consiste à guider l'utilisateur dans l'exploration du *Web des données* à l'aide de la *fouille exploratoire de web des données*. Nous étudions trois axes de recherche, *i.e* : 1) la création des vues sur les graphes RDF et la facilitation des interactions de l'utilisateur sur ces vues, 2) l'évaluation de la qualité des données RDF et la complétion de ces données 3) la navigation et l'exploration simultanée de multiples ressources hétérogènes présentes sur le *Web des données*. Premièrement, nous introduisons un *modificateur de solution* i.e., `View By` pour créer des vues sur les graphes RDF et classer les réponses aux requêtes SPARQL à l'aide de l'analyse formelle des concepts. Afin de naviguer dans le treillis de concepts obtenu et d'extraire les unités de connaissance, nous avons développé un nouvel outil appelé RV-Explorer (*RDF View Explorer*) qui met en oeuvre plusieurs modes de navigation. Toutefois, cette navigation/exploration révèle plusieurs incompletions dans les ensembles des données. Afin de compléter les données, nous utilisons l'extraction de règles d'association pour la complétion de données RDF. En outre, afin d'assurer la navigation et l'exploration directement sur les graphes RDF avec des connaissances de base, les triplets RDF sont groupés par rapport à cette connaissance de base et ces groupes peuvent alors être parcourus et explorés interactivement. Finallement, nous pouvons conclure que, au lieu de fournir l'exploration directe nous utilisons *ACF* comme un outil pour le regroupement de donnéees RDF. Cela permet de faciliter à l'utilisateur l'exploration des groupes de donnéees et de réduire ainsi son espace d'exploration par l'interaction.

Mots-clés: Données ouvertes, analyse de concepts formels, Pattern Structures, Exploration Interactive de données RDF.

Abstract

Recently, the "Web of Documents" has become the "Web of Data", i.e., the documents are annotated in the form of RDF making this human processable data directly processable by machines. This data can further be explored by the user using SPARQL queries. As web clustering engines provide classification of the results obtained by querying web of documents, a framework for providing classification over SPARQL query answers is also needed to make sense of what is contained in the data. Exploratory Data Mining focuses on providing an insight into the data. It also allows filtering of non-interesting parts of data by directly involving the domain expert in the process. This thesis contributes in aiding the user in exploring Linked Data with the help of exploratory data mining. We study three research directions, i.e., 1) Creating views over RDF graphs and allow user interaction over these views, 2) assessing the quality and completing RDF data and finally 3) simultaneous navigation/exploration over heterogeneous and multiple resources present on Linked Data. Firstly, we introduce a *solution modifier* i.e., `View By` to create views over RDF graphs by classifying SPARQL query answers with the help of Formal Concept Analysis. In order to navigate the obtained concept lattice and extract knowledge units, we develop a new tool called RV-Explorer (Rdf View eXplorer) which implements several navigational modes. However, this navigation/exploration reveal several incompletions in the data sets. In order to complete the data, we use association rule mining for completing RDF data. Furthermore, for providing navigation and exploration directly over RDF graphs along with background knowledge, RDF triples are clustered w.r.t. background knowledge and these clusters can then be navigated and interactively explored. Finally, it can be concluded that instead of providing direct exploration we use FCA as an aid for clustering RDF data and allow user to explore these clusters of data and enable the user to reduce his exploration space by interaction.

Keywords: Linked Open Data, Formal Concept Analysis, Pattern Structures, Interactive Exploration of RDF data.