Formal Concept Analysis for Qualitative Data Analysis over Triple Stores

Frithjof Dau and Barış Sertkaya

SAP Research Center Dresden, Germany
(frithjof.dau|baris.sertkaya)@sap.com

Abstract. Business Intelligence solutions provide different means like OLAP, data mining or case based reasoning to explore data. Standard BI means are usually based on mathematical statistics and provide a quantitative analysis of the data. In this paper, a qualitative approach based on a mathematical theory called "Formal Concept Analysis" (FCA) is used instead. FCA allows clustering a given set of objects along attributes acting on the objects, hierarchically ordering those clusters, and finally visualizing the cluster hierarchy in so-called Hasse-diagrams. The approach in this paper is exemplified on a dataset of documents crawled from the SAP community network, which are persisted in a semantic triple store and evaluated with an existing FCA tool called "ToscanaJ" which has been modified in order to retrieve its data from a triple store.

1 introduction

Business Intelligence (BI) solutions provide different means like OLAP, data mining or case based reasoning to explore data. Standard BI means are usually designed to work with numerical data, thus they provide a quantitative analysis of the data (aka "number crunching") based on mathematical statistics. In fact, classical BI examples show "accounting, finance, or some other calculation-heavy subject" [10]. To some extent, though arguably oversimplified, one can understand BI as acting on lists or tables filled with numbers.

Compared to number crunching, Formal Concept Analysis (FCA) [3] provides a complementing approach. The starting point of FCA are crosstables (called "formal contexts"), where the rows stand for some objects, the columns for some attributes, and the cells (intersections of rows and columns) carry the binary information whether an attribute applies to an object (usually indicated by a cross) or not. Based on this crosstable, the objects are clustered to meaningful sets. These clusters form a hierarchy, which can be visually displayed, e.g. by a so-called Hasse-diagram. A short introduction into FCA, as needed for this paper, is provided in the next section.

A general overview over the benefits of FCA in information science is provided by Priss in [8]. Relevant for this paper are the relationships between FCA and both Business Intelligence (BI) and Semantic Technologies (ST).

With respect to BI, FCA can be for example considered as a data mining technology, particularly for mining association rules [7]. More relevant to this

paper is the approach to explore data in relational databases with FCA. As described in the next section, a method called "conceptual scaling" allows transforming columns in a database, filled with arbitrary values, into formal contexts. Such scales can be compared to dimensions in BI applications. The exploration of data in databases with FCA is for example described in [4, 6, 13, 12]. A number of tools for FCA have been developed. Most important for this paper is Toscana [14, 9] , developed in C, and its Java-based successor ToscanaJ [1] [1]. Moreover, it should be mentioned that FCA has been used for exploring data warehouses as well [5].

FCA targets a formalization of the human understanding of concepts with their extensions and intensions, thus FCA indeed is a semantic technology. Though it does not belong to the core of Semantic Web technologies, FCA provides decent means to define and analyze concept hierarchies, so it comes as no surprise that FCA has been used in the realm of querying, browsing, completing and visualizing ontologies (e.g. OWLFCAViewTab and OntoComP[2] [11] plugins for the Protege ontology editor, and OntoViz), ontology alignment (e.g. FCA-Merge and OntEx), ontology engineering (e.g. relational exploration or role exploration) and ontology learning (e.g., Text2Onto). In this paper, we exemplify the benefits of FCA for (semantically enabled) BI by analyzing data in a triple store with FCA methods. In order to do so, the existing ToscanaJ tool has been modified such that it can retrieve data from triple stores instead of relational databases. A short introduction into ToscanaJ and its modifications are provided in Sec. 3. An often named benefit of ST compared to relational databases are the ST capabilities to better deal with unstructured data like text-documents. FCA has already been employed to create concept hierarchies out of the content of text documents. In this paper, we apply FCA on a dataset of documents crawled from the SAP community network[3] (SCN), but do not target to investigate the contents of the documents, but utilize meta-data of the documents (which have been created in the crawling process) for FCA-purposes. This use case is described in Sec. 4.

2 Formal Concept Analysis

Formal Concept Analysis (FCA) is a field of applied mathematics that is based on a lattice-theoretic formalization of the notions of concepts and conceptual hierarchies. FCA provides efficient algorithms for analyzing data and discovering hidden dependencies in the data. It also allows the user to visualize the data in an easily understandable way. In FCA, data is represented in the form of a *formal context*, which in its simplest form is a way of specifying which attributes are satisfied by which objects.

[1] `http://toscanaj.sourceforge.net`
[2] `http://ontocomp.googlecode.com`
[3] `http://www.sdn.sap.com/irj/scn/index`

	male	female	modelCars	books	PCgames	makeUp
mary		×			×	
john	×		×		×	
erika		×		×		×
max	×		×	×	×	

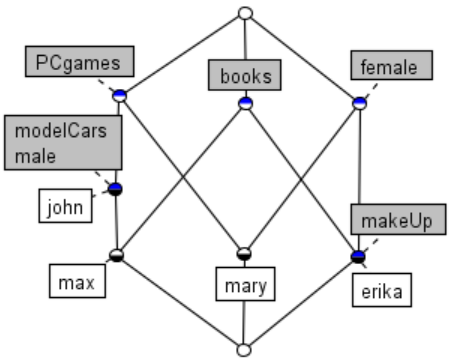


Fig. 1. Formal context of customers (left) and its concept lattice (right)

Example 1. Consider the formal context in Fig. 1. It shows information about the gender of four customers and the product groups they are interested in. For instance the last row states that Max is a male, he is interested in model cars, books and PC games but he is not interested in make-up items.

Given such a formal context, the first step for analyzing this context is usually computing the *formal concepts* of this context, which are "natural clusterings" of the data in the context. A formal concept is a pair consisting of an object set A and an attribute set B such that the objects in A share the attributes in B, and B consists of exactly those attributes that the objects in A have in common. The object set A is called the *extent*, and the attribute set B is called the *intent* of the formal concept (A, B).

Example 2. Consider the formal context given in Ex. 1. It has nine formal concepts. One of them is $(\{max\}, \{male, modelCars, PCgames, books\})$ with the extent $\{max\}$ and the intent $\{male, modelCars, PCgames, books\}$. Note that max is male and has exactly the interests in modelCars, PCgames, books. On the other hand, max is the only male person with these interests. Another (less trivial) formal concept is $(\{john, max\}, \{male, modelCars, PCgames\})$. Both john and max are male and have modelCars and PCgames as (common) interests, and they are indeed the only male persons with (at least) these interests.

Once all formal concepts of a context are obtained, one orders them w.r.t. the inclusion of their extents (equivalently, inverse inclusion of their intents). For example, the two formal concepts of the above given example are ordered that way. This ordering gives a complete lattice (e.g. a hierarchy where any two elements have -like in trees- a least upper bound and -unlike in trees- a greatest lower bound), called the *concept lattice* of the context. A concept lattice contains all information represented in a formal context, i.e., we can easily read off the attributes, objects and the incidence relation of the underlying context. Moreover, concept lattice can be visualized, which makes it easier to see formal

concepts of a context and interrelations among them. Thus it helps to understand the structure of the data in the formal context, and to query the knowledge represented in the formal context.

The nodes of a concept lattice represent the formal concepts of the underlying context. In order to improve readability of the lattice, we avoid writing down the extent and intent of every single node. Instead, we label the nodes with attribute and object names in such a way that every name appears only once in the lattice. In this labelling, the intent of the formal concept corresponding to a node can be determined by the attribute names that can be reached by the ascending lines, and its extent can be determined by the object names that can be reached by the descending lines. For instance consider the concept lattice in Figure 1 that results from the formal context in Example 1. The attribute names are written in boxes with gray background and object names are written in boxes with white background. The intent of the formal concept marked with the attribute name *books* is $\{books\}$ since there is no other attribute name that can be reached by an ascending line, and its extent is $\{max, erika\}$ since these are the only two object names that can be reached by a descending line from it. Similarly, the concept marked with the attribute names $modelCars$ and $male$, and the object name $john$ has the intent $\{modelCars, male, PCgames\}$ and the extent $\{john, max\}$.

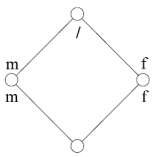
FCA, as it has been described so far, can only deal with *binary* attributes. For real data, the situation is usually different: Attributes assign specific values (which might be strings, numbers, etc) to data. For example, RDF-triples (s, p, o) are exactly of this form: The attribute p - from now on we will use the RDF-term "property" instead- assigns the value o to the entity s. In FCA, a process called "conceptual scaling" is used to deal with this issue.

Let a specific property be given with a set of possible values. A *conceptual scale* is a specific context with the values of the property as formal objects. The choice of the formal attributes of the scale is a question of the design of the scale: The formal attributes are meaningful attributes to describe the values; they might be different entities or they might even be the values of the property again. To exemplify conceptual scaling, we reuse a toy example from [15], which is the following table provided on the right with two many-valued properties "sex" and "age". Note that empty cells are possible as well.

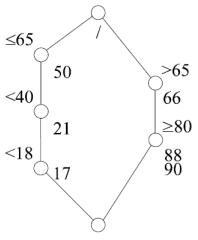
	sex	age
Adam	m	21
Betty	f	50
Chris		66
Dora	f	88
Eva	f	17
Fred	m	
George	m	90
Harry	m	50

Next, two conceptual scales for the properties "sex" and "age" and their line diagrams are provided.

S_1	m	f
m	×	
f		×



S_2	< 18	< 40	≤ 65	> 65	≥ 80
17	×	×	×		
21		×	×		
50			×		
66				×	
88				×	×
90				×	×



With conceptual scales, the initial many-valued context can be transformed into a standard context, so that the corresponding concept lattice can be displayed.

For conceptual scales, the following two points should be noted:

1. There is no standard or even neccessary interpretation of an attribute: It has to be decided by the field expert which scale is appropriate. As discussed in [2], this is indeed not a drawback, but an advantage of FCA.
2. Conceptual scales do not depend on the real data, but only on the properties (and their values, of course) used in the data set. As one can see in the example, a realized context is derived from the scales and the real data in a later step after the scales have been created.

Both points are important for ToscanaJ, which is discussed in the next section.

3 ToscanaJ

There is a variety of software for FCA available. Most of them support the creation of contexts from scratch and the subsequent computation and display of the corresponding concept lattices. Contrasting this approach, Elba and ToscanaJ are a suite of mature FCA-tools which allow to query and navigate through data *in databases*. They are intended to be a *Conceptual Information System (CIS)*. CISs are "systems that store, process, and present information using concept-oriented representations supporting tasks like data analysis, information retrieval, or theory building in a human centered way." Here, a CIS is an FCA-based system used to analyze data stored in one table of an RDBMS.

Similar to other BI-systems, in CIS we have to distinguish between a design phase and a run-time-phase (aka usage phase), with appropiate roles attached to the phases. In the design phase, a CIS engineer (being an expert for the CIS) together with a domain expert who has limited knowledge of a CIS) develops the CIS schema, i.e. those structures which will be later on used to access the system. This schema consists of manually created conceptual scales. Developping the scales is done with a CIS editor (Elba) and usually a highly iterative process. In the run-time phase, a CIS browser (ToscanaJ) allows a user to explore and analyze the real data in the database with the CIS schema.

The original Elba/ToscanaJ-suite has been developped to analyze data in a *relational table*, i.e. a table in a RDBMS or an excel-file. We have extended the suite in order to be able to access data in a *triple store*. This extended version of the suite uses the Sesame framework[4] for accessing a triple store and querying the RDF data therein. It provides two ways of connecting to a triple store over Sesame. One of them is over HTTP via Apache Tomcat[5], the other one is over the SAIL API[6]. Tomcat is an open source software implementation of the Java

[4] see http://www.openrdf.org/doc/sesame2/system
[5] see http://tomcat.apache.org/
[6] see http://www.openrdf.org/doc/sesame2/system/ch05.html

Servlet and JavaServer Pages technologies by the Apache Software Foundation. The SAIL API (Storage And Inference Layer) is a low level system API for RDF stores and inferencers. It is used for abstracting from the storage details, allowing various types of storage and inference to be used.

In a triple store we do not directly have the notions of tables and columns like in databases. As table information we use the type information in the triple store: we treat the objects of triples with the predicate `rdf:type` as tables. As column information, we use the predicates relating the subjects of the selected type to any object. More precisely, in order to detect the columns we get those subjects of the selected type and retrieve all distinct predicates that relate these subjects to an object.

The Elba/ToscanaJ-suite provides different kinds of conceptual scales. We have extended three of them –namely nominal scales, attribute scales and context tables– in order to act on triple stores.

Nominal scales are the simplest type of scales one can automatically create in Elba. They are used for properties with mutually exclusive values. For a given property, the formal attributes of the nominal scale are selected values of that property. As each object is assigned at most one of these values, the attribute concepts form an anti-chain, and by definition, the scale cannot reveal any insight into attribute dependencies. In the example provided in the next section, we consider a predicate `threadStatus` for messages, which has the values `Answered` and `Unanswered`. This predicate is modelled as conceptual scale.

Attributes scales offer an attribute centered view which is very close to "classical" formal contexts and which allows to create complex scales in an intuitive manner. In an attribute list scale, each attribute is a property-value pair, which is manually selected from the triple store. Moreover, the CIS engineer can choose between a) ?use only combinations existing in the database? and b) ?use all possible combination?. If option a) is selected, then the diagram will only consist of concepts that could be derived from the data in the triple store, thus the diagram will reveal insights into dependencies between property-value pairs. If b) is chosen, a diagram of a Boolean lattice of all listed property-value pairs will be created independent of whether there exists objects in the triple store for each property-value combination or not.

Context table scales offer the most freedom and power to the CIS engineer. In context tables, arbitrary labels act as formal attributes. As now, in contrast to the last two types of scales, no property-value pairs are chosen as attributes, it has now explicitely to be specified which objects of the data set fulfill the formal attributes. This is done by entering SPARQL expressions, which act as formal objects, and by entering the incidence relation as well, i.e. the relation which here relates the formal objects (the SPARQL expressions) to the attributes (the labels).

4 Use case

In order to evaluate our approach, we have used a dataset crawled from the SAP Community Network (SCN). SCN contains a number of forums for SAP users and experts to share knowledge, or get help on SAP topics and products. The dataset we have used is taken from the forum *Service-Oriented Architecture (SOA)*, which contains 2600 threads and 10076 messages. The dataset is annotated by the crawler using ontologies from the NEPOMUK project. The used ontologies and their meanings are provided below along with short descriptions taken from the project website [7].

- NEPOMUK Information Element Ontology (NIE): The NIE Framework is an attempt to provide unified vocabulary for describing native resources available on the desktop.
- NEPOMUK file ontology (NFO): The NFO intends to provide vocabulary to express information extracted from various sources. They include files, pieces of software and remote hosts.
- NEPOMUK Message Ontology (NMO): The NMO extends the NIE framework into the domain of messages. Kinds of messages covered by NMO include Emails and instant messages.
- NEPOMUK Contact Ontology (NCO): The NCO describes contact information, common in many places on the desktop.

From these ontologies, our dataset uses the following classes as types:

- nie#DataObject: A unit of data that is created, annotated and processed on the user desktop. It represents a native structure the user works with. This may be a file, a set of files or a part of a file.
- nfo#RemoteDataObject: A file data object stored at a remote location.
- nie#InformationElement: A unit of content the user works with. This is a superclass for all interpretations of a DataObject.
- nco#Contact: A Contact. A piece of data that can provide means to identify or communicate with an entity.
- nmo#Message: A message. Could be an email, instant messanging message, SMS message etc.

For analyzing experience levels of the users of the SOA forum, we used the *Contact* type above and created a scale based on the number of posts, number of questions, number of resolved questions information provided in the data. We have named users that have less than 50 posts as *newbie*, users that have more than 300 posts as *frequent*, users that have more than 1000 posts as *profi*, users that have asked more than 310 questions as *curious* and people that have resolved more than 230 questions as *problem solver*. Note that this scale uses different measures (number of posts, number of questions, numbers of answers). The concept lattice in Figure 2 shows number of users with the mentioned experience levels. The diagram clearly displays the sub/super-concept-relationships

[7] `http://www.semanticdesktop.org/ontologies`

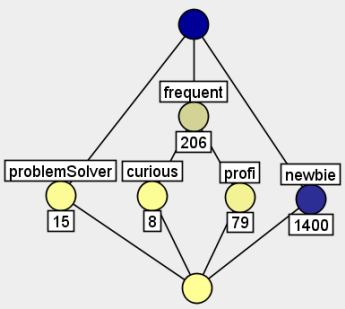


Fig. 2. Diagram of the scale based on number of posts, questions, resolved questions

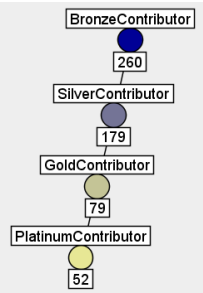


Fig. 3. Diagram of the scale based on number of points

between the experience levels, which is one of the main distinguishing features of visualizing data using concept lattices. E.g. we can read from the lattice that curious and professional users are both also frequent users, whereas problem solvers and newbies are not.

Next, for analyzing experience levels based on the *number of points* information in our dataset we created another scale. This time, as labels we took contributor types that are officially defined by SCN as *bronze*, *silver*, *gold* and *platinium* contributors, which have more than 250, 500, 1500 and 2500 points respectively. The concept lattice of this scale is shown in Figure 3. This scale is a so-called *ordinal* scale, which means that the formal concepts are ordered as a chain. This is also easily seen in the concept lattice of this scale. Obviously, a user that has more than 2500 points also has more than 1500 points, and so on.

The above displayed concept lattices are separately informative about the properties of forum users, i.e., the first one about experience levels based on number of posts, questions and resolved questions, and the second one about number of points. One of the most powerful techniques of FCA is to "combine" such lattices to give a combined view of several lattices together, which is called a *nested line diagram*. In its simplest form, a nested line diagram is a concept lattice whose concepts are themselves also concept lattices. Nested line diagrams allow the user to select a concept and zoom into it to see the lattice nested in that concept. Figure 4 shows the nested line diagram of the diagrams in the Figures 2 and 3. Note that the outer diagram is actually the one in Figure 3. The four bigger circles correspond to the four types of contributors in that figure. The inner diagrams are the diagram in Figure 2. Figure 5 shows an excerpt of the nested diagram that corresponds to the node *golden contributor*, and Figure 6 shows the inner diagram of this node. Note that the number of users corresponding to different levels of experience in this diagram differs from that of diagram in Figure 2. The reason is that, now we zoomed into the node gold contributor so the information in the inner diagram is restricted to the gold contributors only. For instance, as seen in this diagram there are no newbies that are gold contributors, which is quite natural. On the other hand 79 of the

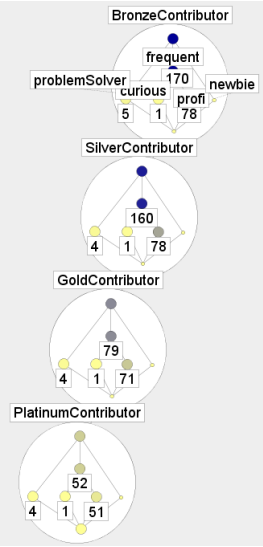


Fig. 4. Nesting the above two scales

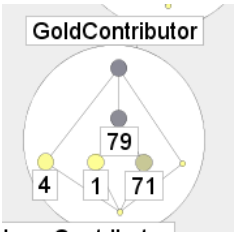


Fig. 5. Detail of Figure 4

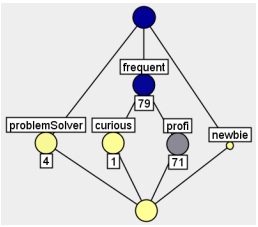


Fig. 6. Inner diagram of Fig. 5

gold contributors are profi users. In ToscanaJ, and thus in our extension of it to triple stores, one can nest an arbitrary number of diagrams and can browse nested diagrams easily by zooming in and out.

5 Conclusion and Further Research

We have discussed how FCA can be used as a methodology for analyzing data in triple stores by extending the Toscanaj suite. As scales in the ToscanaJ workflow are manually crafted in the design phase of a CIS, this workflow is feasible for stable schemata. For ST, this is usually not the case: here the paradigm of agile schema development is prevalent. As future work we plan to implement automatic or at least semi-automatic generation of scales based both on the schema information and the actual data in the triple store. existing BI approaches. Another future research direction is the development of hybrid solutions, combining "classical" BI with FCA. This covers combinations of scales and their diagrams with BI diagrams for numerical data, like pie charts or sun-burst diagrams, and compared to nesting of scales, different approaches for using simultaneously several scales. In our work we have considered RDF models only as simple object-attribute-value models, ignoring the subclass relationships. As future work we are also going to work on integrating such knowledge into FCA as background knowledge for concept lattice computation.

Disclaimer: Parts of this work have been carried out in the CUBIST project, which is funded by the European Commission under the 7th Framework Programme of ICT, topic 4.3: Intelligent Information Management.

References

1. P. Becker, J. Hereth, and G. Stumme. ToscanaJ: An open source tool for qualitative data analysis. *Advances in Formal Concept Analysis for Knowledge Discovery in Databases, (FCAKDD 2002)*, 2002.
2. F. Dau and J. Klinger. From formal concept analysis to contextual logic. *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *LNCS*, pages 81–100. Springer-Verlag, 2005.
3. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, Germany, 1999.
4. J. Hereth. Formale begriffsanalyse und data warehousing. Masters thesis, TU Darmstadt, Germany, 2000.
5. J. Hereth. Relational scaling and databases. *Conceptual Structures: Integration and Interfaces, Proceedings of the 10th International Conference on Conceptual Structures, (ICCS 2002)*, volume 2393 of *LNCS*, pages 62–76. Springer-Verlag, 2002.
6. J. Hereth, G. Stumme, R. Wille, and U. Wille. Conceptual knowledge discovery - a human-centered approach. *Journal of Applied Artificial Intelligence (AAI)*, 17(3):281–301, 2003.
7. L. Lakhal and G. Stumme. Efficient mining of association rules based on formal concept analysis. *Formal Concept Analysis: Foundations and Applications*, volume 3626 of *LNAI*, pages 180–195. Springer-Verlag, 2005.
8. U. Priss. Formal concept analysis in information science. *Annual Review of Information Science and Technology*, 40, 2005.
9. M. Roth-Hintz, M. Mieth, T. Wetter, S. Strahringer, B. Groh, and R. Wille. Investigating snomed by formal concept analysis. In *Proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000.
10. S. Scheps. *Business Intelligence for Dummmies*. John Wiley and Sons Ltd., Chichester, UK, UK, 2008.
11. B. Sertkaya. Ontocomp: A protege plugin for completing owl ontologies. *Proceedings of the 6th European Semantic Web Conference, (ESWC 2009)*, volume 5554 of *LNCS*, pages 898–902. Springer-Verlag, 2009.
12. G. Stumme. Conceptual on-line analytical processing. *Information Organization and Databases*, chapter 14. Kluwer, Boston-Dordrecht-London, 2000.
13. G. Stumme, R. Wille, and U. Wille. Conceptual knowledge discovery in databases using formal concept analysis methods. *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*, volume 1510 of *LNAI*, pages 450–458. Springer-Verlag, 1998.
14. F. Vogt and R. Wille. Toscana — a graphical tool for analyzing and exploring data. *Graph Drawing*, pages 226–233. Springer-Verlag, 1995.
15. K. E. Wolff. A first course in formal concept analysis. *Proceedings of Advances in Statistical Software 4*, pages 429–438, 1993.