Prospects for Document Retrieval using Formal Concept Analysis

Peter Becker, *Peter Eklund*

Knowledge, Visualization and Ordering Laboratory (KVO)
Distributed Systems Technology Centre (DSTC)
GRIFFITH UNIVERSITY
PMB 50 Gold Coast 9726, Australia

pbecker@dstc.edu.au, *peklund@dstc.edu.au*

Abstract

Formal Concept Analysis is a technique based on lattice theory that is well suited for document retrieval. The structure of the concept lattice created maps to a query in a natural way and refinements can be found easily. These refinements are ensured to be smaller but never empty. From the lattice structure a ranking on the documents can be found to give results that are related but do not lexically match.

Keywords Document Management, Information Retrieval, Formal Concept Analysis

1 Introduction

Formal Concept Analysis (FCA) is a technique derived from lattice theory that has been successfully used for various analysis purposes. More recently many researchers have started applying this technique for retrieval systems with very promising results.

The lattice created by Formal Concept Analysis is similar to the tree created by hierarchical clustering approaches but, due to the properties of a lattice, it does not suffer from one of the main problems of hierarchical approaches: documents indexed in a similar fashion will always be close in the concept lattice while they can be distant in a tree due to the limitations of the structure.

The distances in the lattice can also be used to offer a similarity ranking to the user of a retrieval system. First results indicate that this approach is very successful. Documents rated as similar by such systems are rated as similar by the users and vice versa.

Formal Concept Analysis will be introduced shortly in Section 2. In Section 3 we show how a concept lattice can be used for document retrieval purposes – an idea that is extended in Section 4 to include a notion of document vicinity useable for ranking. In Section 5, a reference implementation is introduced which a short discussion of the indexing approaches considered in this project. Section 6 gives a concluding discussion of this paper.

Proceedings of the Sixth Australasian Document Computing Symposium, Coffs Harbour, Australia, December 7, 2001.

2 Formal Concept Analysis

FCA[7] has a long history as a technique of data analysis ([10], [9]). Following this methodology, data is organized as a table (see Fig. 1) and is modeled mathematically as multi-valued context, (G, M, W, I) where G is a set of objects, M is a set of attributes, W is a set of attribute values and I is a relation between G, M, and W such that if (g, m, w_1) and (g, m, w_2) then $w_1 = w_2$. In the table there is one row for each object, one column for each attribute, and each cell is either empty or contains an attribute value.

Organization over the data is achieved via conceptual scales. A conceptual scale maps attribute values to new attributes and is represented by a mathematical entity called a formal context. A formal context is a triple (G, M, I) where G is a set of objects, M is a set of attributes, and I is a binary relation between the objects and the attributes, i.e. $I \subseteq G \times M$. A conceptual scale is defined for a particular attribute of the multi-valued context: if $\mathbb{S}_m = (G_m, M_m, I_m)$ is a conceptual scale of $m \in M$ then we require that $W_m \subseteq G_m$. The conceptual scale can be used to produce a summary of data in the multi-valued context as a derived context. The context derived by $\mathbb{S}_m = (G_m, M_m, I_m)$ wrt to plain scaling from data stored in the multi-valued context (G, M, W, I) is the context (G, M_m, J_m) where for $g \in G$ and $n \in M_m$

$$g J_m n \;:\Leftrightarrow\; \exists w \in W : (g, m, w) \in I \text{ and } (w, n) \in I_m$$

Scales for two or more attributes can be combined together in a derived context. Consider a set of scales, S_m, where each $m \in M$ gives rise to a different scale. The new attributes supplied by each scale can be combined together using a special type of union:

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

Then the formal context derived from combining all these scales together is:

$$gJ(m,n) \;\Leftrightarrow:\quad \begin{array}{l} \exists w \in W : (g,m,w) \in I \\ \text{and } (w,n) \in I_m \end{array}$$

The derived context is then displayed to the user as a lattice of concepts. A concept of a formal context (G, M, I) is a pair (A, B) where $A \subseteq G$, $B \subseteq M$, $A = \{g \in G \mid \forall m \in B : (g,m) \in I\}$ and $B = \{m \in M \mid \forall g \in A : (g,m) \in I\}$. For a concept (A, B), A is called the extent and is the set of all objects that have all of the attributes in B. Similarly, B is called the intent and is the set of all attributes possessed in common by all the objects in A. As the number of attributes in B increases, the concept becomes more specific, i.e. a specialization ordering is defined over the concepts of a formal context by:

$$(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow B_2 \subseteq B_1$$

In this representation more specific concepts have larger intents and are considered "less than" ($<$) concepts with smaller intents. The same partial ordering is achieved by considering extents, in which case more specific concepts have smaller extents. The partial ordering over concepts is always a lattice. Partial orders are commonly drawn using a Hasse diagram. A property of concept lattices can be exploited to achieve an efficient labelling. Each attribute has a single maximal concept (wrt the specialization ordering) possessing that attribute. If attribute labels are only attached to their maximal concepts then the intent of a concept can be determined by collecting labels from all greater concepts. A similar situation is achieved for objects. Each object has a minimal concept to which its label is attached and the extent of a concept can be determined by collecting labels from all lesser concepts. Attribute and object labels are disambiguated by attaching object labels from below and attribute labels from above.

Consider Fig. 1. The set of objects in the shown multi-valued context is a set of documents each denoted by a number: $G = \{1, 2, 3, 4, 5, 6\}$. For simplicity we consider just one attribute, giving $M = \{"Keywords"\}$, where the attribute itself has different sets of keywords as values, the set of values is $W = \{\{FCA\}, \{Retrieval\}, \{FCA, Retrieval\}, \{FCA, Scales\}\}$ and the incidence relation I is given in the table cells.

The right table shows one of the most direct ways of scaling for this kind of attribute where each possible member is mapped onto a single-valued attribute. In this way we can split an attribute with values given as a set into a set of boolean values showing the signature of the sets. Since we assume no dependencies on the different values we allow each combination thereby getting the cube as the lattice for the scale, as shown in the lower right corner of Fig. 1.

Applying this scale to the original context we get a lattice as depicted in the lower left corner. In this diagram we can easily see some relations between the attributes and the documents, e.g. the position of the attribute "Scales" below "FCA" indicates the implication that each document indexed with "Scales" is indexed with "FCA", too. The position of the document "5" below "Retrieval" and "FCA" indicates that this document is indexed with these two keywords, in this way we can reproduce the original data from the diagram – no information has been lost in this scaling process.

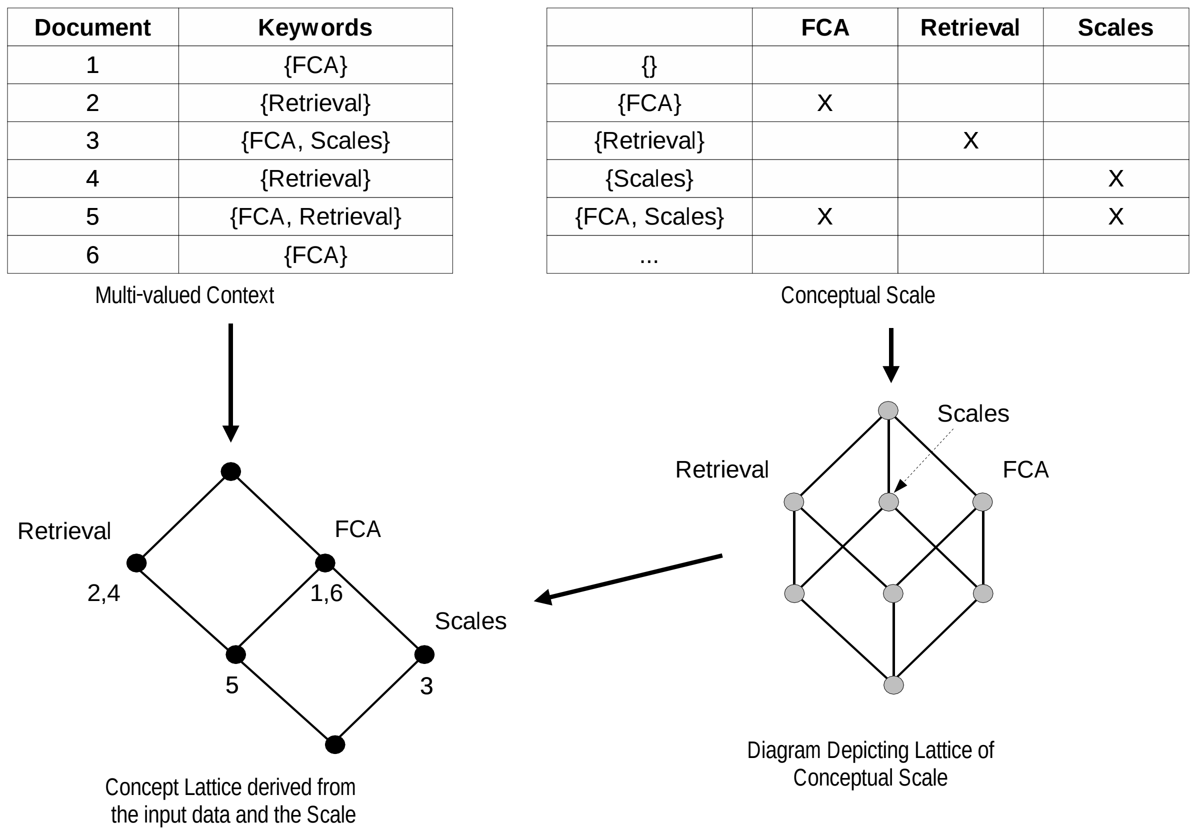
Another typical scaling technique is to map numerical or similar values onto intervals as shown in Table 1. Here we deliberately reduce the information in the original data to reduce the number of attributes created by the scale. Note that we create attribute implications using this scale, thereby creating inference chains in the lattice. For example every document written in December 2000 will be written in the year 2000, too. Therefore the node labeled "written Dec 2000" will be below the node labeled "written in 2000".

3 Concepts and Queries

Although Formal Concept Analysis has been mainly used for data analysis purposes it has a number of features that make it well suited for retrieval. When using Formal Concept Analysis as a retrieval tool one can see a concept as a representation of a query state, where the intent of the concept represents the query itself and the extent represents all documents that match the query.

The first interesting feature that can be found in the lattice structure is that adding additional attributes into a query will always move the query state to some concept below the current state, which maps mathematically to the definition of the subconcept relation. To find all query states that model such extensions of the current query we have to find all subconcepts of the current position.

By comparing the intents of the concepts along the edges we can map the attributes that can be added to the new position in the lattice. In the example given in Fig. 1 the possible refinements for a query "FCA" would be "Retrieval", pointing to the concept with document 5 in the extent and



Document	Keywords
1	{FCA}
2	{Retrieval}
3	{FCA, Scales}
4	{Retrieval}
5	{FCA, Retrieval}
6	{FCA}

	FCA	Retrieval	Scales
{}			
{FCA}	X		
{Retrieval}		X	
{Scales}			X
{FCA, Scales}	X		X
...			

Multi-valued Context
Conceptual Scale
Scales
Retrieval
FCA
Retrieval
FCA
2,4
1,6
Scales
5
3
Diagram Depicting Lattice of Conceptual Scale
Concept Lattice derived from the input data and the Scale

Figure 1: Example showing the process of generating a derived concept lattice from a multi-context and a conceptual scale for the attribute *Keywords*.

"Scales", pointing to the concept labeled with this attribute.

The bottom element of the lattice is another query state that might be considered as refinement for the current state, it will be reached by adding both "Retrieval" and "Scales" to the query. But since there is no document indexed with both of these keywords the bottom element has no extent and therefore we do not consider it as a valid refinement to avoid having an empty result set. Every query that contains some contradiction, i.e. queries for something that does not exist in the data set will point to a bottom element with an empty extent and we can avoid offering these refinements easily.

Another useful feature when finding query refinements this way is that once we found the states that are possible refinements we have the size of the new result set which is the size of the extent of the concept that will be reached. Using this feature of the lattice we can give the user the information on the size of the next result set before she chooses a refinement, a front-end can allow sorting or filtering the refinements by this size.

The last interesting feature of a concept lattice used in document retrieval is that the extent of a subconcept will always be a true subset of the current concept. In terms of retrieval this means that the result set of a refinement will always be a true subset of the original result set. If an attribute is not in the current query, but is true for all documents in the current result set, it will implicitely be in the current query state since it will be positioned at the current concept or a superconcept.

These features ensure that refinements found using Formal Concept Analysis will always be true refinements in the sense that the new result set will be smaller, but the result set can be ensured to be never empty. In addition to avoiding the empty result set one can tell the new size of the result sets for all refinements. This gives the user more information to make her decision.

4 Concept Lattice Ranking

Another property of a concept lattice can be used to create a ranking on documents and to find documents that do not precisely match a query ([4]). Whenever two documents have similar attributes they will be located in concepts close together in the lattice, e.g. the documents 3 and 1 in Fig. 1 are close together since they differ only in the additional attribute "Scales" for document 3, while

Date of Creation	**written in 2000**	**written Dec 2000**	**written week 51/2000**
11 Nov 2000	×		
24 Nov 2000	×		
13 Dec 2000	×	×	×
22 Dec 2000	×	×	
07 Jan 2001			

Table 1: Example scale for a date. The attributes create a chain of refinements.

the distance between the documents 3 and 2 is far bigger – they have no attribute in common.

This property can be used to define a distance on the documents. There are different ways to define the distance, depending on the semantic used for a query. Generally, all documents in the extent of a concept are matches to all attributes in the query, which means all documents labeled at any concept below the current query state are matches for the query. Following an edge upwards in the lattice diagram to some concept, not below the current query state, means loosing at least one match. Documents labeled at one of these nodes remain close to the query so they can be ranked as possible matches with a lower ranking than the exact matches.

In addition to this approach one might consider documents labeled directly at the current query state as better hits than documents attached to subconcepts of the current query state, since the latter match more attributes and are therefore less specific. These documents are probably a better match than documents found by going upwards, although they might be considered less exact hits.

Another approach to refine this distance measure is to use the sizes of the intent and the extent into the ranking calculation. If only one document is found in a neighbour this might be considered as more relevant to the current query as other documents in a concept as close with a larger difference in the extent. A dual argument can be given for the attributes: if the intent differs significantly between two concepts, we might consider them as more different than two concepts with only slight changes to the intent. Since the lattice ensures the subset relations for upper and lower neighbours comparing the sizes is sufficient for comparing the sets.

Once these distances are calculated the documents can be presented ordered by this ranking. Experiments show that this kind of ranking might be superior to classic approaches like best-match ranking and hierarchical clustering-based ranking, the latter suffering from the limitations of the underlying tree structure.

5 The Score Project

A reference implementation for a Web-based system using the techniques described has been implemented and is accessible on the web (http://meganesia.int.gu.edu.au/projects/score). At the time of writing it is still in a evolving state but shows the basic features described in Section 3.

This system is implemented in different subsystems using a relational database as central storage for the indexing information. The architecture of the system allows the combination of different indexers and frontends while keeping the Formal Concept Analysis implementation, currently written in Java. The frontend uses Java Servlets to create HTML pages that are displayed in any Web browser. The indexer is a simple Perl script communicating with the WebKB ontology server (http://www.webkb.org) using the UNIX command line to find keywords suitable for indexing.

The data set used is a mailing list archive for network communications, keywords are English nouns taken from an ontology of communication terms. The keywords are stemmed using a simple stemming technique and the information on synonyms in the ontology is used but no other text indexing techniques are used yet.

It is planned to implement different indexing techniques and apply them on different data sets, from a large collection of short news articles to an online book which can be addressed on the paragraph level. The indexing techniques shall involve ontologies and classic text retrieval techniques like using inverse document frequency and stemming. Kim and Compton use ripple-down rules ([8]) to achieve well-structured interactive indexing.

The major question for the Formal Concept Analysis module is whether or not this approach is scalable to large object and attribute numbers. The current implementation is still completely memory-based, we are not aware of any research trying to handle large lattices. Storing the information in a relational database system or distributing the lattice using direct lattice products might be options to scale well.

For the frontend other questions arise. One of the main questions is how to find semantically useful refinements. One approach found in the literature is to offer only the lower neighbours of the current query state as refinements, a graphical interface for this approach is shown by Carpineto and Romano ([3]). A combination with a classic retrieval interface can be used to offer the user different ways of refinements to avoid problems with sets of refinements that are too large, see [8] as example.

Another option for the frontend can be the integration of complete line diagrams as e.g. presented by Cole, Eklund and Stumme ([5]). Using techniques developed for an interactive data analysis program called Cernato ([2]) it might be possible to offer further interaction with the line diagrams suitable even for inexperienced users. Ferré and Ridoux model the frontend as command-line interface ([6]) using the metaphor of a hierachical file system when navigating the lattice while another approach tries to integrate this retrieval technique into the UNIX desktop KDE ([1]) using a list-based frontend.

The Score system with its open architecture and major parts Open Sourced in the Tockit project (see http://tockit.sourceforge.net) should offer solid ground for comparing different approaches in the indexing and the frontend parts.

6 Summary

Formal Concept Analysis has many properties that are useful for document retrieval. In combination with different indexing techniques and different user interfaces promising research implementations have been created. The lattice structure created by Formal Concept Analysis does not suffer from the problem of having to drop edges as other tree-based techniques do and the graphical presentations used in other applications of Formal Concept Analysis might be useful as addition to a classical text-based frontend.

With the Tockit project presented here a common Open Source platform for working on Formal Concept Analysis should be created and Score should be a reference platform for different indexing and retrieval approaches using Formal Concept Analysis. More detailed results are expected to follow.

References

[1] P. Becker. Einsatz der Formalen Begriffsanalyse zur Dokumentennavigation. http://www.peterbecker.de/texts/becker99.pdf, 1999.

[2] P. Becker. Multi-dimensional representations of conceptual hierarchies. In *Conceptual Structures — Extracting and Representing Semantics, Contributions to ICCS 2001*, pages 145–158, 2001.

[3] C. Carpineto and G. Romano. Effective reformulation of boolean queries with concept lattices. In *Flexible Query Answering Systems FQAS'98*, pages 277–291, Berlin Heidelberg, 1998. Springer–Verlag.

[4] C. Carpineto and G. Romano. Order-theoretical ranking. *Journal of the American Society for Information Science (JASIS)*, Volume 51, Number 7, pages 587–601, 2000.

[5] R. Cole, P. Eklund and G. Stumme. CEM — a program for visualization and discovery in email. In *Proceedings of PKDD 2000*, number 1910 in LNAI, pages 367–374, Berlin, 2000. Springer-Verlag.

[6] S. Ferré and O. Ridoux. Searching for objects and properties with logical concept analysis. In *Conceptual Structures — Broadening the Base, ICCS 2001*, number 2120 in LNAI, pages 187–201, Berlin Heidelberg, 2001. Springer–Verlag.

[7] B. Ganter and R. Wille. *Formal Concept Analysis — Mathematical Foundations.* Springer–Verlag, Berlin Heidelberg, first edition, 1999.

[8] M. Kim and P. Compton. A web-based browsing mechanism based on conceptual structures. In *Conceptual Structures — Extracting and Representing Semantics, Contributions to ICCS 2001*, pages 47–60, 2001.

[9] W. Kollewe, M. Skorsky, F. Vogt and R. Wille. TOSCANA — ein Werkzeug zur begrifflichen Analyse und Erkundung von Daten. In R. Wille and M. Zickwolff (editors), *Begriffliche Wissensverarbeitung — Grundfragen und Aufgaben*, pages 267–288, Mannheim, 1994. B. I.–Wissenschaftsverlag.

[10] Frank Vogt and Rudolf Wille. Toscana — a graphical tool for analyzing and exploring data. In Roberto Tamassia and Ioannis G. Tollis (editors), *Graph Drawing*, pages 226–233, Heidelberg, 1995. Springer–Verlag.