FCA_Retrieval: A Multi-operator Algorithm for Information Retrieval from Binary Concept Lattice

Fethi Fkih
MARS Research Lab LR17ES05
University of Sousse
Tunisia
fethi.fkih@gmail.com

Mohamed Nazih Omri
MARS Research Lab LR17ES05
University of Sousse
Tunisia
mohamednazih.omri@fsm.rnu.tn

Abstract

The use of a concept lattice for the visual representation of knowledge has become very common in the data mining field. Indeed, the theory of Formal Concept Analysis (FCA) gives the possibility of organizing and making reasoning on the knowledge stored in the lattice. In this context, we propose FCA_Retrieval a fast and efficient algorithm for information retrieval from a binary concept lattice. This algorithm can respond to queries containing several types of logical operators (conjunction, disjunction and negation). The experimental study proves the good performance of our algorithm.

1 Introduction

With the evolution of documents quantity in databases, it has become very difficult to extract the information corresponding to the exact needs of the user in a reasonable time without the design of an effective strategy. Indeed, this strategy must be based, first, on the right representation of the information found in the documents or in the query and, secondly, on the development of an efficient and fast algorithms for information retrieval (Fkih and Omri, 2012).

In this context, concept lattices are considered as very powerful tools for the visual presentation of knowledge (Poelmans et al., 2013). Also, Formal Concept Analysis (FCA) is considered a flexible and practical theoretical framework for binary relations management. The relation between a document and the information that presents it is a binary relation (the information presents the document or not). Therefore, a concept lattice can easily model the document/term relation and can be considered as an semantic index (Fkih and Omri, 2016).

In this paper, we propose an efficient algorithm (FCA_Retrieval) for the retrieval of information in a concept lattice. This algorithm uses the theory of formal concept analysis (FCA) to respond to a user query and retrieve the information in a reasonable time. The main advantage of this algorithm is that it can handle multiple logical operators queries (i.e. conjunctive, disjunctive and negation operators).

The remainder of this paper is structured as follows. In section 2,we provide an overview on the works that use the concept lattice for documents indexing. Section 3 presents the basic notions of the Formal Concept Analysis (FCA) theory. In section 4, we introduce our algorithm, FCA_Retrieval, for information retrieval in a concept lattice. Section 5 is reserved for an experimental comparative study between the proposed model and other models in the literature.

2 Related Works

The use of concept lattices for information retrieval began with the introduction of the Formal Concept Analysis (FCA) theory by (Wille, 1982) in the 1980s. Information retrieval is considered a direct application of the concept lattice due to the high correspondence between the theoretical foundations of this approach and those of information retrieval (Godin et al., 1995). Indeed, if we replace objects by documents and attributes by terms, the formal context becomes a simple inverted file (index) used for

documents indexing.

There are several approaches for information retrieval based on concept lattice, we cite: information retrieval systems for medical documentation (Cole and Eklund, 1996; Cole and Eklund, 1999), software documentation (Linding, 1995; Priss, 2000) and Bioinformatic Databases (Smal-Tabbone et al., 2005). We can classify FCA-based models into three main families: CLR (Concept Lattice-based Ranking) methods, refinement and organization methods and classification-based reasoning methods.

2.1 Concept Lattice-based Ranking methods

Approaches belonging to the CLR family are based essentially on the idea of considering the query as a new entry for the formal context. In this case, the search for documents satisfying the query turns into a problem of reconstruction of the lattice. These retrieval algorithms use incremental construction techniques to insure the insertion of the new entry in context. These incremental construction techniques are more efficient, from the point of view of complexity and efficiency, than those which reconstruct, from scratch, the lattice. In the literature there are several models belonging to the CLR family, such as: BR-Explorer (Messai et al., 2006), BMR (current best-match ranking) and HCR (model and hierarchical clustering-based ranking) (Carpineto and Romano, 2000a).

2.2 Refinement and organization methods

These approaches use a search engine to respond to a user query and exploit the links of documents as well as summaries and titles provided by the engine. Their primary role is to provide the user the ability to refine his query by browsing the concept lattice and exploring neighbouring concepts. For instance, we cite systems CREDO (Carpineto and Romano, 2004), FooCA (Koester, 2006) and SearchSleuth (Dau et al., 2008; Ducrou, 2007).

2.3 Classification-based reasoning methods

This approach uses the concept lattice as an index of a documentary database and applies navigation algorithms to the lattice to respond to a query. The main aim of this type of method is to maximize the lattice navigation. In this case, the lattice is seen as a semantic index for information retrieval since it is the result of a clustering operation. Among systems belonging to this family, we cite CLAIR (Codocedo et al., 2014). This system is based on the insertion of a query in a classification-based reasoning.

3 Basic notions of Formal Concept Analysis

Definition 1 *(Formal context) A formal context is a triplet $F = (D, T, I)$ where:*

- *D : set of documents.*
- *T : terms set.*
- *I : binary relation between D and T verifying that $I \subseteq D \times T$*

I is called the incidence relation of F. Let $d \in D$ and $t \in T$, the relation $(d, t) \in I$ means that the document d contains the term t (we also say that the term t is satisfied by the document d).

Definition 2 *(Derivation operator) Also called sufficiency operator as in (Dubois et al., 2007), derivation operators are considered as the basis of the formal analysis. Given $X \subseteq D$ and $Y \subseteq T$, we denote by X^{Δ} (equation 1) the set of terms satisfied by the set of documents X and Y^{Δ} (equation 2) the set of documents satisfying the set of terms Y.*

$$X^{\Delta} = \{y \in T \mid X \subseteq I(y)\} \tag{1}$$

$$Y^{\Delta} = \{x \in D \mid Y \subseteq I(x)\} \tag{2}$$

Intuitively, X^{Δ} is the set of attributes common to all the objects of X and Y^{Δ} is the set of the objects that share all the attributes of Y.

Definition 3 *(Formal concept) A formal concept is a pair $\langle X, Y \rangle$ such that $X^{\Delta} = Y$ and $Y^{\Delta} = X$. X and Y are called respectively the extent and the intent of the concept $\langle X, Y \rangle$. An order relation, denoted $\preceq$, is defined on the set of all concepts. We have the following equivalence (equation 3):*

$$\begin{aligned} \langle X_1, Y_1 \rangle \preceq \langle X_2, Y_2 \rangle &\Leftrightarrow X_1 \subseteq X_2 \\ &\Leftrightarrow Y_2 \subseteq Y_1 \end{aligned} \tag{3}$$

Definition 4 *(Lattice Concept) The set of concepts ordered by $\preceq$ forms a complete lattice denoted $B(D,T,I)$. We say that $B(D,T,I)$ is the lattice associated with the formal context $F = (D,T,I)$. We denote:*

- *τ: the most general concept belonging to $B(D,T,I)$, also called the top element.*
- *β: the most specific concept belonging to $B(D,T,I)$, also called the bottom element.*

4 Proposed algorithms

Our approach is based mainly on the use of the specification/generalization relation that exists between the lattice concepts (see figure 1). Indeed, a concept lattice is the result of a clustering operation which groups the documents sharing the same terms in common concepts. The lattice top (τ) is considered as the most general concept in the lattice; it groups in its extent all the documents of the collection. The intent of τ contains common terms between all documents if they exist, the empty set otherwise. If we move to the lower hierarchy level, we find the maximum subsets of documents that share the same terms. In fact, the more we descend in the lattice more we add terms to the intent of each concept, i.e., we specify the documents. The most specific concept is the bottom of the lattice (β), it contains all the terms in its intent and contains the documents that share all the terms in its extent (the empty set, if there are no documents).

For a given query Q, containing a variety of logical operators, our algorithm treats each operator type alone. That is, it separates the conjunction operators from those of disjunction or negation. As it presents the algorithm 1, it begins with the disjunctive part of the query and then processes the disjunctive part. For the negation operator, the algorithm 2 calls algorithm 1 to treat the conjunction and disjunction operators then it applies the negation to the returned result.

4.1 Conjunctive operators of the query

For conjunction operators, the idea is to find (in the lattice) the most general concept (C_g^{conj}) containing in its intent all the conjunctive terms of the query, by starting the search with the most specific concept

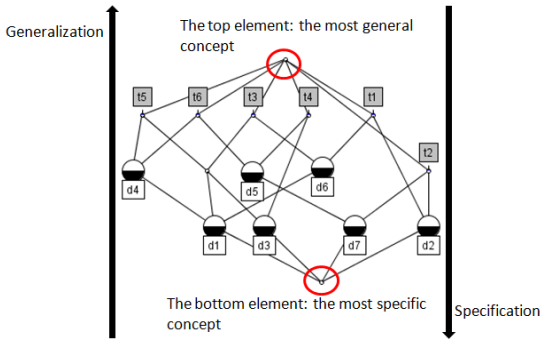

Figure 1: Specification/Generalization relation in a concept lattice.

(β). The proposition 1 defines the characteristics of the concept C_g^{conj}.

Proposition 1 *Let T_{conj}^{Q} The set of conjunctive terms of the query Q, and let $C_g^{conj} = \langle Intent_{C_g^{conj}}, Extent_{C_g^{conj}} \rangle$ the most general concept in the lattice such that: $T_{conj}^{Q} \subseteq Intent_{C_g^{conj}}$ then, we say that: the documents set $Extent_{C_g^{conj}}$ satisfy the conjunctive part of the Query Q.*

The β concept must include, in its intent, all the conjunctive terms of the query; otherwise it's said that the lattice can't satisfy the query. If the beta intent contains all the conjunctive terms we pass to the direct parents of the concept (the highest hierarchical level). If one of the concepts don't contain all the conjunctive terms in its intent then it is useless to continue the search in its parents (this will simplify the search space and decrease the complexity of the algorithm). This simplification is justified by the fact that the intent of the parent is included in the intent of his son then if the terms don't exist in the intent of the son, they don't also exist in the intent of the father. Otherwise, if a concept contains all the conjunctive terms and all of its direct parents don't contain them then it's considered as C_g^{conj} and the result will be the documents of its extent. The algorithm stops when it finds C_g^{conj} or if it finishes the entire lattice.

4.2 Disjunctive operators of the query

After completing the conjunctive part of the query Q, the algorithm proceeds to the processing of the disjunctive part. Our idea is to look for the most general concept (C_g^{disj}) containing at least one term

Algorithm 1: FCA_Retrieval(): Generalized algorithm for multiple logical operators query

Data: $F = (D, T, I)$, $B(D, T, I)$
Q: multiple logical operators query
$T_Q = \{t_1, t_2, ..., t_{n-1}, t_n\}$
ψ_{conj}: set of concept to explore by conjunctive operators
ψ_{disj}: set of concept to explore by disjunctive operators
τ: Lattice Top (most general concept in the lattice)
β: Lattice Bottom (most specific concept in the lattice)
Result: R_d: set of documents satisfying the query Q)

begin
$R_d \longleftarrow \{\emptyset\}$
$\psi_{conj} \longleftarrow \{\beta\}$
$\psi_{disj} \longleftarrow \{\tau\}$
for $C \in \psi_{conj}$ **do**
  if $T_Q \subseteq intent(C)$ **then**
    $Clean(R_d)$
    $R_d \longleftarrow extent(C)$
    $\psi_{conj} \longleftarrow \psi_{conj} \cup Fathers(C)$
    $\psi_{conj} \longleftarrow \psi_{conj} \setminus \{C\}$
  else
    $\psi_{conj} \longleftarrow \psi_{conj} \setminus \{C\}$
    delete $Ancestors(C)$ from the lattice
  end
end
for $C \in \psi_{disj}$ **do**
  if $T_Q \cap intent(C) = \{\emptyset\}$ **then**
    $\psi_{disj} \longleftarrow \psi_{disj} \cup Sons(C)$
    $\psi_{disj} \longleftarrow \psi_{disj} \setminus \{C\}$
  else
    $R_d \longleftarrow R_d \cup extent(C)$
    delete $Descendants(C) \cup \{C\}$ from the lattice
    $\psi_{disj} \longleftarrow \psi_{dis} \setminus \{C\}$
  end
end
Return R_d
end

of the disjunctive query, by starting with the most general concept (τ). The following proposition 3 describes the characteristics of C_g^{disj}:

Proposition 2 *Let T_{disj}^Q The set of disjunctive terms of the query Q, and let $C_g^{disj} = \langle Intent_{C_g^{disj}}, Extent_{C_g^{disj}} \rangle$ the most general concept in the lattice such that: $T_{conj}^Q \bigcap Intent_{C_g} \neq \emptyset$ then, we say that: the documents set $Extent_{C_g^{disj}}$ satisfy the disjunctive part of the Query Q.*

The algorithm 1 explores the intent of τ, if it finds at least one term of the disjunctive part of the query Q in its intent then it's said that all documents in the collection satisfy the query, otherwise the algorithm passes to its direct sons. If we find a term, at least, in the intent of one of its sons then we take its extent and we ignore all its descendants. The research is continued until all the lattice concepts are checked.

4.3 Negation operators of the query

The idea of negation is very simple: first, ignored the negation operator, in this case we can apply the algorithm 1. Then take the complement of the set of returned documents. Algorithm 2 is based on the following proposition:

Proposition 3 *Let $Q = \neg Q'$, and let $D_{Q'}$ the documents set that satisfying to Q', then, we say that: the documents set $\overline{D_{Q'}}$ satisfy the Query Q.*

Algorithm 2: FCA_Retrieval_Negative(): an algorithm for negative query

Data: $F = (D, T, I)$, $B(D, T, I)$
Q': negative query
$T_{Q'} = \{t_1, t_2, ..., t_{n-1}, t_n\}$
$Q' = \neg Q$
Result: R_d^{Neg}: set of documents satisfying the query Q'

begin
$R_d^{Neg} \longleftarrow FCA\_Retrieval(\mathrm{Q})$
Return R_d^{Neg}
end

4.4 Demonstrative example

Table 1 presents a formal context I that models a binary relation between the set of documents $D =$

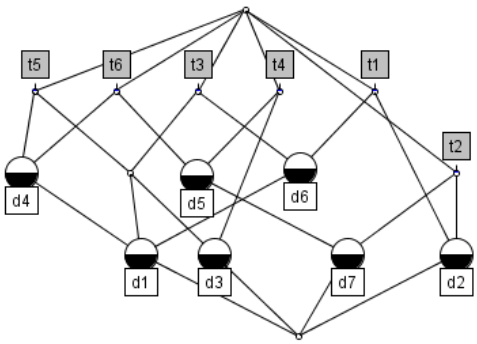

Figure 2: Concept lattice $B(D, T, I)$.

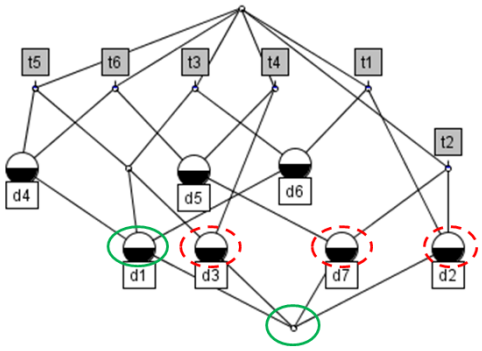

Figure 3: There are a one concept contains in its intent the query terms (surrounded by a green circle)

$\{d1, d2, d3, d4, d5, d6, d7\}$ and the set of terms $T = \{t1, t2, t3, t4, t5, t6\}$. In this case, the formal context is considered as a binary document-term incidence matrix. Practically, if a document contains a term then we put 1, 0 otherwise.

Table 1: An example of a formal context $F = (D, T, I)$ modelling the relation document-term

Document-Term	t1	t2	t3	t4	t5	t6
d1	1	0	1	0	1	1
d2	1	1	0	0	0	0
d3	0	0	1	1	1	0
d4	0	0	0	0	1	1
d5	0	0	0	1	0	1
d6	1	0	1	0	0	0
d7	0	1	0	1	0	1

Figure 2 shows the Hasse diagram of the concept lattice $B(D, T, I)$ corresponding to the formal context $F = (D, T, I)$. This diagram is drawn by ConExp[1], a free software for building and visualizing lattices.

For this example, the corresponding concepts to the top (τ) and the bottom (β) of the lattice $B(D, T, I)$ are:

$$\tau = \langle \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}, \{\emptyset\} \rangle \quad (4)$$

$$\beta = \langle \{\emptyset\}, \{t_1, t_2, t_3, t_4, t_5, t_6\} \rangle \quad (5)$$

In this example, we try to apply our algorithm 1 to the lattice described in the figure 2. This set of documents (noted R_d) To do this, we look for documents that satisfy to the query $Q = t_5 \wedge t_6 \vee t_2$.

First, we start with the processing of the conjunctive part of the query, i.e, $t_5 \wedge t_6$. As the algorithm specifies, we begin the exploration of the lattice by the processing of the concept β (the most specific concept in the lattice). In this level, we find that β contains all query terms, so $R_d = \{\emptyset\}$.

Thereafter,we move on to the exploration of higher level concepts (direct fathers). We find a single concept that its intent contains all the query terms (surrounded by a green circle in the figure 3), all the fathers of the other concepts (surrounded by a red circle in the figure 4) will be ignored by the algorithm to simplify the search space. The extent of this concept contains a single document $d1$. Then, we update the set of results $R_d = \{d1\}$.

Before stopping the processing of the conjunctive part of the query, there is one concept to explore (surrounded by a green circle in the figure 4). We find that the intent of this concept contains all the terms of the query, so we assign its extent to R_d. The set of results becomes $R_d = \{d1, d4\}$.

After having finished processing the conjunctive part of the query, the algorithm begins processing the disjunctive part, i.e., it looks (in our example) for the documents containing the term $t2$. To do this, our algorithm explores the concept τ (the most general concept in the lattice). In our example, the algorithm can't find the term $t2$ in the intent of τ, so it pursues the search in its direct descendants. The algorithm finds $t2$ in the intent of a direct descendant of τ (surrounded by a green circle in the figure 5), then it adds all the documents that are in its extent

[1] http://sourceforge.net/projects/conexp/

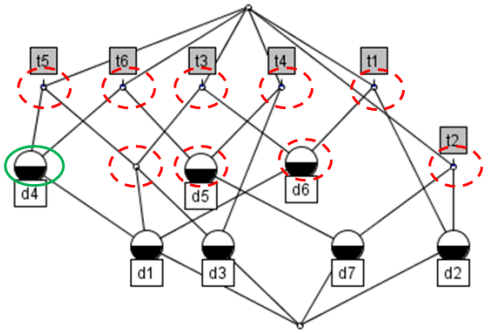

Figure 4: The processing of the conjunctive part of the query stops when the algorithm processes all the concepts of the lattice.

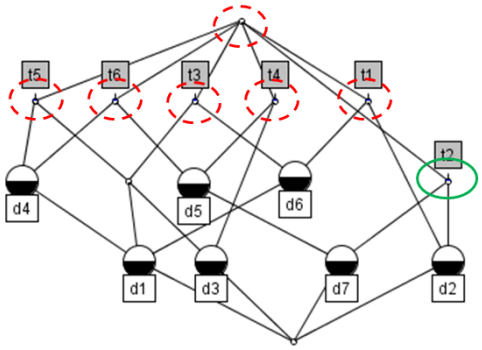

Figure 5: The algorithm starts the processing of the disjunctive part of the query.

(in this case, $d2$ and $d7$) to the set of results, which becomes: $R_d = \{d1, d4, d2, d7\}$. In this stage, the algorithm ignores all the descendants of the concept containing the term $t2$. In addition, it ignores the neighbours of this concept, because, it's useless to look for $t2$ in concepts of the same hierarchical level (all concepts surrounded by red circles in figure 6). Thus, the algorithm stops the processing of the query Q and returns the set of documents R_d.

To simulate the behaviour of our algorithm vis--vis the logical negation operator, we try to process the query $Q^{'} = \neg Q$. As indicated in algorithm 2, we first call algorithm 1 and then we look for the complement of the returned set, i.e., if R_d is the set of documents returned by algorithm 1, then the set of documents returned by algorithm 2 can be defined by: $R_d^{Neg} = \overline{R_d} = \overline{\{d1, d4, d2, d7\}} = \{d3, d5, d6\}$.

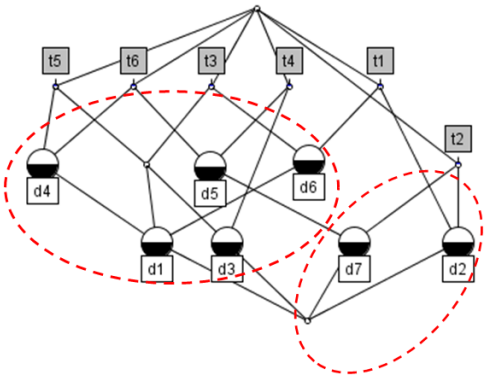

Figure 6: If the algorithm finds the term $t2$ in a concept, then it ignores its descendants and neighbors and stops processing.

5 Experimental study

To show that our algorithm is faster and simpler than other algorithms in the literature, we conduct a comparative study between FCA_Retrieval, CLR (Carpineto and Romano, 2000b), BR-Explorer (Messai et al., 2006) and CLAIR (Codocedo et al., 2014). In order to achieve this, we implement all these algorithms in Java programming language on a Windows 7 platform. To evaluate the performance in execution time, the experimental tests were conducted on a Intel Core i5-6200U machine having a clock frequency of 2.8Ghz and 8GB of main memory.

5.1 Test Data description

The different models were evaluated using Connect-4[2], a standard dataset. This test data is seen as a multi-valued formal context (each attribute can take one of three values of the set $\{x, o, b\}$). To make it more compatible with our application, we have transformed this multi-valued formal context to a binary formal context. Indeed, each attribute of the formal context takes 0 if its value is b, 1 otherwise.

In this stage, our goal is to study the behaviour of the different models (taking the execution time as an indicator) if we vary the size of the data set (number of objects). To achieve this, we have prepared 15 formal contexts (extracted from Connect-4) each one contains 42 attributes and a number of instances ranging from 1 to 10000 (see table 5.1).

[2]http://archive.ics.uci.edu/ml/datasets/Connect-4

Table 2: Test Data description

Connect-4		
Data sets	Objects number	Concepts number
S_1	1000	575
S_2	2000	685
S_3	3000	1314
S_4	4000	1387
S_5	5000	1564
S_6	6000	1620
S_7	7000	1620
S_8	8000	1620
S_9	9000	2203
S_{10}	10000	2564

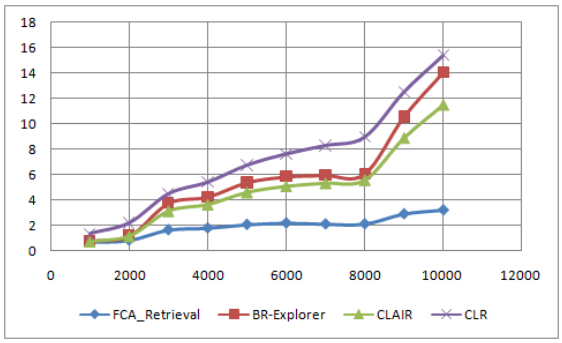

Figure 7: Comparative study between our algorithm and other algorithms of the literature.

5.2 Results and discussion

Each model is tested on the 10 concept lattices using 10 different queries. In practice, we calculate for each model the response time to a given query. Subsequently, we calculate the average time for all queries. Table 5.2 summarizes the results obtained for all models. Figure 7 shows the variation of the execution speed of each algorithm according to the number of objects in the lattice.

The experimental study shows that our algorithm is faster than the other algorithms. In fact,the gap between the performance of our algorithm and other algorithms increases with the number of objects. By way of example, the difference between the execution times of FCA_Retreival and CLR can reach 12.22 for a trellis of 10000 objects.

The experimental result can be justified by a theoretical study. Therefore, if we compare the complexity of our algorithm with the complexity of other algorithms, we find that FCA_Retrieval is the least complex. The table shows the temporal complexity of each model. The complexity of FCA_Retrieval is $O(n)$ (worst case, where n is the number of concepts in the lattice) while the complexity of the others is $O(n^2)$ which justifies the superiority of our algorithm in execution time.

6 Conclusion

In this paper, we have presented a novel algorithm for information retrieval from a binary concept lattice. This algorithm is based on the theory of formal concept analysis and can handle a query containing a variety of logical operators (conjunction, disjunction and negation). The experimental study shows that our model is efficient and provides the most optimal execution time.

As future work, we try to integrate semantic resources into our model to deal with the semantic ambiguity that exists in user queries.

References

Claudio Carpineto and Giovanni Romano. 2000a. Order-theoretical ranking. *J. Am. Soc. Inf. Sci.*, 51(7):587–601, May.

Claudio Carpineto and Giovanni Romano. 2000b. Order-theoretical ranking. *J. Am. Soc. Inf. Sci.*, 51(7):587–601, May.

Claudio Carpineto and Giovanni Romano. 2004. *Concept Data Analysis: Theory and Applications*. John Wiley & Sons, Chichester, England.

Victor Codocedo, Ioanna Lykourentzou, and Amedeo Napoli. 2014. A semantic approach to concept lattice-based information retrieval. *Annals of Mathematics and Artificial Intelligence*, 72(1-2):169–195.

R. J. Cole and P.W. Eklund. 1996. Text retrival for medical discharge summaries using snomed and formal concept analysis. In *The University of New South Wales*, pages 50–58.

Richard Cole and Peter W. Eklund. 1999. Scalability in formal concept analysis. *Computational Intelligence*, 15(1):11–27.

Frithjof Dau, Jon Ducrou, and Peter Eklund. 2008. Concept similarity and related categories in searchsleuth. In Peter Eklund and Ollivier Haemmerl, editors, *Conceptual Structures: Knowledge Visualization and Reasoning*, volume 5113 of *Lecture Notes in Computer Science*, pages 255–268. Springer Berlin Heidelberg.

Table 3: Evolution of the execution time according to the objects number

Data sets		**Average time (second)**			
Sets	Object count	FCA_Retrieval	BR-Explorer	CLAIR	CLR
S_1	1000	0.682	0.763	0.696	1.340
S_2	2000	0.827	1.143	1.126	2.185
S_3	3000	1.640	3.728	3.101	4.483
S_4	4000	1.785	4.223	3.612	5.404
S_5	5000	2.063	5.359	4.573	6.747
S_6	6000	2.183	5.818	5.058	7.642
S_7	7000	2.101	5.925	5.298	8.306
S_8	8000	2.108	6.032	5.537	8.970
S_9	9000	2.901	10.530	8.852	12.532
S_{10}	10000	3.208	14.027	11.466	15.433

Table 4: Complexity comparison

Model	Complexity
FCA_Retrieval	$O(n)$
CLR	$O(n^2)$
BR-Explorer	$O(n^2)$
CLAIR	$O(n^2)$

Didier Dubois, Florence Dupin de Saint-Cyr, and Henri Prade. 2007. A possibility-theoretic view of formal concept analysis. *Fundam. Inf.*, 75(1-4):195–213, January.

Jon Ducrou. 2007. Dvdsleuth: A case study in applied formal concept analysis for navigating web catalogs. In Uta Priss, Simon Polovina, and Richard Hill, editors, *Conceptual Structures: Knowledge Architectures for Smart Applications*, volume 4604 of *Lecture Notes in Computer Science*, pages 496–500. Springer Berlin Heidelberg.

Fethi Fkih and Mohamed Nazih Omri. 2012. Information retrieval from unstructured web text document based on automatic learning of the threshold. *IJIRR*, 2(4):12–30.

Fethi Fkih and Mohamed Nazih Omri. 2016. IRAFCA: an o(n) information retrieval algorithm based on formal concept analysis. *Knowl. Inf. Syst.*, 48(2):465–491.

R. Godin, R. Mineau, R. Missaoui, and H. Mili. 1995. Méthodes de classification conceptuelle basées sur les treillis de galois et applications. *Revue d'intelligence artificielle*, 9(2):105–137.

Bjoern Koester. 2006. Conceptual knowledge retrieval with fooca: Improving web search engine results with contexts and concept hierarchies. In Petra Perner, editor, *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, volume 4065 of *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin Heidelberg.

C. Linding. 1995. Concept-based component retrieval. In *IJCIA-95 Workshop on Formal Approaches to the Reuse of Plans, Proofs and Programs*, August.

Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smaïl-Tabbone. 2006. BR-Explorer: An FCA-based algorithm for Information Retrieval. In *Fourth International Conference On Concept Lattices and Their Applications - CLA 2006*, Hammamet/Tunisia, Oct.

Jonas Poelmans, Sergei O. Kuznetsov, Dmitry I. Ignatov, and Guido Dedene. 2013. Formal concept analysis in knowledge processing: A survey on models and techniques. *Expert Systems with Applications*, 40(16):6601 – 6623.

Uta Priss. 2000. Lattice-based information retrieval. *Knowledge Organization*, 27:132–142.

Malika Smal-Tabbone, Shazia Osman, Nizar Messai, Amedeo Napoli, and Marie-Dominique Devignes. 2005. Bioregistry: A structured metadata repository for bioinformatic databases. In Michael R. Berthold, Robert C. Glen, Kay Diederichs, Oliver Kohlbacher, and Ingrid Fischer, editors, *CompLife*, volume 3695 of *Lecture Notes in Computer Science*, pages 46–56. Springer.

Rudolf Wille. 1982. Restructuring lattice theory: An approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered Sets*, volume 83 of *NATO Advanced Study Institutes Series*, pages 445–470. Springer Netherlands.