A semantic approach to concept lattice-based information retrieval

Victor Codocedo, Ioanna Lykourentzou, Amedeo Napoli

Noname manuscript No.
(will be inserted by the editor)

A semantic approach to Concept Lattice-based Information Retrieval

Victor Codocedo · Ioanna Lykourentzou · Amedeo Napoli

the date of receipt and acceptance should be inserted later

Abstract The volume of available information is growing, especially on the web, and in parallel the questions of the users are changing and becoming harder to satisfy. Thus there is a need for organizing the available information in a meaningful way in order to guide and improve document indexing for information retrieval applications taking into account more complex data such as semantic relations. In this paper we show that Formal Concept Analysis (FCA) and concept lattices provide a suitable and powerful support for such a task. Accordingly, we use FCA to compute a concept lattice, which is considered both a semantic index to organize *documents* and a search space to model *terms*. We introduce the notions of cousin concepts and classification-based reasoning for navigating the concept lattice and retrieve relevant information based on the content of concepts. Finally, we detail a real-world experiment and show that the present approach has very good capabilities for semantic indexing and document retrieval.

1 Introduction

The increasing amount of information available nowadays implies more and more the ability to accurately retrieve documents relevant to user needs. Several approaches have been proposed, regarding this task, in the field of information retrieval (IR) [17]. Document retrieval for example, a sub-task of IR, focuses on how to exploit the basic elements of information that documents contain (terms, phrases, links, etc.) and their in-between relations, in order to construct a document index that users can browse and query. However, as information becomes more complex, high-dimensional and domain-specific, these information elements become too limited in their capacity to identify relevant documents for a given user need, and thus other factors, such as semantics, need to be considered for the purpose of document indexing and retrieval. Consequently, semantic indexing for document retrieval has gained importance in the IR literature [8]. In this work we present a novel technique to combine both, i.e. the relations among documents through the terms they share and the semantics of

Victor Codocedo · Ioanna Lykourentzou · Amedeo Napoli
LORIA - CNRS - INRIA - Univesité de Lorraine, BP 239, 54506 Vandœuvre-les-Nancy.

Ioanna Lykourentzou
Centre de Recherche Public Henri Tudor - 29, avenue John F. Kennedy L-1855, Luxembourg.

those terms, in order to improve the performance of document retrieval systems. To achieve this, we combine two typical document retrieval techniques, namely *"navigation"* among document relations and *"ranking"* of documents, using a notion of similarity between the semantics of document terms and the keywords of a user query. Both techniques, as shown in [3], can be naturally modelled and implemented in a document-term concept lattice computed from a document collection.

Formal Concept Analysis (FCA) is a theoretical as well as practical framework for classifying objects in a concept lattice based on the relations they have through the attributes they share [12]. As such, FCA can be used to understand and exploit the relations that documents have w.r.t. the terms they have in common. Therefore the concept lattice can be used to facilitate the navigation through a document space. Concept lattices have been used in a variety of domains on information sciences [27]. Particularly, on the domain of IR many approaches have demonstrated their usefulness regarding document indexing and retrieval [4,6,10,13,23,25,26]. Furthermore, they have proved to yield better or comparable results with regards to traditional document retrieval approaches, such as Hierarchical Clustering and Best-Match Ranking [3]. Despite this fact, as described in [4] and in [25], a few works exist in the area of semantic indexing using concept lattices.

More formally, in this paper we present a semantic indexing and document retrieval technique based on FCA. It relies on the general idea of constructing a document-term concept lattice used as an index to answer a given user query. The benefits of using a concept lattice as a query index are two-fold. Firstly, the concept lattice provides a structured support for the full query space (possible queries in a document collection), since it contains all possible modifications (or variations) of the original user query and their corresponding documents, organized in a partial order. Therefore it allows to consider the problem of document retrieval as a problem of navigation inside the lattice, starting from an original "query concept" and following the principles of classification-based reasoning [22]. Secondly, the lattice allows an easy incorporation of domain knowledge at attribute (term) level, thus significantly improving the semantic aspect of document retrieval that we need to address. For this, we develop a novel concept lattice exploration strategy based on the notion of "cousin concepts", as well as a new approach for ranking concepts based on their in-between semantic similarities [11], measured using an external lexical hierarchy. In the same way we anchor our document indexing and retrieval technique to the formal concept analysis definitions, we frame the entire process (comprising from document analysis to results presentation) to the knowledge discovery in databases (KDD) framework [1]. From a process design perspective, we take advantage of the robust and clear KDD process to guide the main steps to be completed in order to create an IR system which satisfies the users' information needs, in this case represented by small sub-sets of documents of vast document corpora. From a theoretical perspective, KDD is an accepted framework with a well established supporting community and an extensive literature regarding its relations with FCA [24].

To summarize, the main contributions of this paper are the introduction of cousin concepts and the use of semantic relations to enable document ranking on a concept lattice-based information retrieval system, built and described as a KDD-like process. Finally, we validate our approach using 4 typical IR document datasets and comparing it to 3 currently used document retrieval techniques. Results show that our approach achieves better performance for most of the ranking evaluation methods used.

The rest of this paper is organized as follows: Section 2 presents the basic technical background, comprising the notions of FCA, concept lattices and concept lattice-based information retrieval. Section 3 presents our document retrieval system CLAIR (concept lattices for information retrieval), built as a KDD-like approach that details the mains aspects

of this work (lattice navigation, cousin concepts and semantics-based ranking). Section 4 presents and discusses the experimental evaluation. Section 5 presents the related work. Finally Section 6 concludes the paper and presents perspectives.

2 Background

2.1 Information content as a semantic relation measure

We can measure the semantic relation between any two terms by using the measure known as *Lin similarity* [16]. This is a measure related to the amount of information carried by a term or a word within a context (piece of text, document or corpus). Consider for example that we have a medical document D_1 indexed by two terms: "arthroscopy"[1] and "complication" and that we are trying to find documents similar to D_1. In order to do so, we take one of D_1 terms (let us pick the term "arthroscopy") and look for documents that contain that term. Consider that we have two of such documents, the first containing the terms "arthroscopy" and "practice" and the second containing "arthroscopy" and "infection". The question now is how to identify which of these documents is more similar to the original one. Intuitively, we may choose the one with the term "infection" which supposes a kind of "complication" in the context of a surgery such as an arthroscopy (indeed, a very serious complication), while the term "practice" is much more general making the document with that term less similar to the original. This notion of information correlation between two terms or between the information content shared by them in a given context can be measured with Lin similarity which takes in consideration the actual frequency correlation in a text corpus as well as the commonalities those terms have in a lexical hierarchy (such as a dictionary). To formalize, given two terms m_1 and m_2, the Lin similarity between m_1 and m_2 is defined as:

$$lin(m_1, m_2) = \frac{2 \log p(m_s)}{\log p(m_1) + \log p(m_2)} \tag{1}$$

where $p(m_i)$ is the probability of the term m_i to appear in a corpus of documents and m_s is the "lowest common subsumer'" of terms m_1 and m_2 in the lexical hierarchy. In this work, we use the *Brown corpus*[2], as the corpus of documents to measure the probability of term appearance, and *WordNet* as the lexical hierarchy that will yield the lowest common subsumer of the terms. These resources were selected since they are widely used in IR systems. Nevertheless, our approach is not restricted to use them exclusively and they can be replaced for other similar resources related to a given domain. In the following, we provide a short description for these resources.

Brown Corpus. The Brown Corpus is a general text collection, which contains samples of 500 English language text documents, and approximately one million words, widely used in text linguistics. The Brown corpus was used in this work to calculate term frequencies ($p(m_i)$ in Equation 1).

WordNet. WordNet[3] is a well-known semantic dictionary, which associates terms with their meanings, called synsets [21]. Each term in WordNet may be associated with several synsets, where each synset corresponds to one specific meaning of the term. Synsets inside WordNet are organized into a hierarchical tree structure, based on their hypernym/hyponym

[1] Arthroscopy refers to a surgery on a joint using an arthroscope.

[2] http://khnt.aksis.uib.no/icame/manuals/brown/

[3] Wordnet is a widely-used free semantic dictionary organized in a hierarchical manner [21]

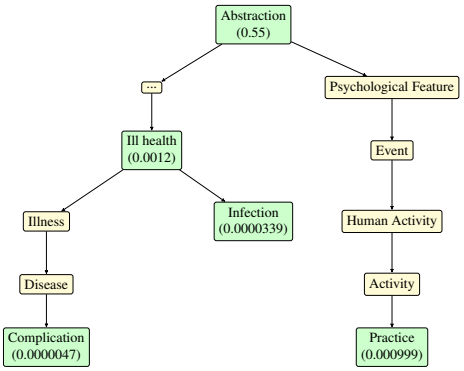


Fig. 1: Terms “complication”, “infection” and “practice” as positioned inside the lexical hierarchy Wordnet, shown in darker boxes together with their minimal common subsumers. The box with three dots represents 5 unimportant terms in the hierarchy.

relations. In this work we use Wordnet to obtain the lowest common subsumer m_s in Equation 1. The lowest common subsumer for two synsets is simply the lowest synset in the Wordnet hierarchy which is a hypernym for both of them.

Let us see an example of using both external knowledge sources and the Lin similarity measure, to calculate semantic similarity between terms. In Figure 1, the terms “complication”, “infection” and “practice” are shown along with their least common subsumers “ill health” and “abstraction” (darken boxes), as found in WordNet. The number under each term is the probability of appearance of that term in the Brown corpus. One may observe that actually the term “complication” is very close to the term “infection” (the sense of complication defined as: “any disease or disorder that occurs during the course of (or because of) another disease”). On the other hand, “practice” is very far from the term “complication” (14 steps in the tree compared to only 4 for “infection”) sharing the least common subsumer “abstraction” (defined as “a general concept formed by extracting common features from specific examples”). This is confirmed the Lin similarity value of the two candidate term pairs: 0.59 for the “complication-infection” and only 0.062 for the “complication-practice” term pair, which leads to the selection of “infection” as the term to replace “complication”.

2.2 Formal Concept Analysis and Concept Lattices

In order to present the rationale of our approach, it is essential to first present a brief description to Formal Concept Analysis (FCA). The basics of FCA are introduced in [12], but we recall some useful notions for the understanding of the paper.

Data is encoded in a formal context $\mathcal{K} = (G, M, I)$, i.e. a binary table where G is a set of objects, M a set of attributes, and $I \subseteq G \times M$ an incidence relation indicating by gIm that the object g has the attribute m. For $A \subseteq G$ and $B \subseteq M$, two derivation operators $(\cdot)'$ are defined as follows:

$' : \wp(G) \longrightarrow \wp(M)$ with $A' = \{m \in M \mid \forall g \in A, gIm\}$

$' : \wp(M) \longrightarrow \wp(G)$ with $B' = \{g \in G \mid \forall m \in B, gIm\}$,

where $\wp(G)$ and $\wp(M)$ respectively denote the powersets of G and M. The two derivation operators $'$ form a Galois connection[4] between $\wp(G)$ and $\wp(M)$ [12]. For a set of objects A, A' is the set attributes which are common to all objects in A. Analogously, for a set of attributes B, B' is the set of objects having all attributes in B. A *formal concept* is defined as a pair (A, B) where $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$, A being the *extent* and B the *intent* of the formal concept (in this case $A'' = A$ and $B'' = B$).

The set $\mathfrak{B}(G, M, I)$ of all concepts from $\mathcal{K}$ is ordered by extent inclusion, denoted by $\leq_{\mathcal{K}}$, i.e. $(A_1, B_1) \leq_{\mathcal{K}} (A_2, B_2)$ when $A_1 \subseteq A_2$ (or dually $B_2 \subseteq B_1$). In this case we say that (A_1, B_1) is the super-concept of (A_2, B_2) and inversely, (A_2, B_2) is the sub-concept of (A_1, B_1). The *concept lattice* of $\mathcal{K}$ is denoted by $\underline{\mathfrak{B}}(G, M, I)$.

For an object $g \in G$, the *object intent* is defined as $g' = \{m \in M | gIm\}$. Correspondingly, for $m \in M$, the *attribute extent* is defined as $m' = \{g \in G | gIm\}$. For a given object g, the *object concept* is defined by $\gamma(g) = (g'', g')$ (where g'' stands for $(g')'$). Dually, for a given attribute m, the *attribute concept* is $\mu(m) = (m', m'')$. Intuitively, the *object concept* is the smallest-extent concept in the lattice which includes the object. The *attribute concept* is the smallest-intent concept which contains the attribute. An example is given in the legend of Table 2.

2.3 Foundations of Information Retrieval based on Concept Lattices

2.3.1 The principles of Concept Lattice-based Ranking

The basic idea of current methods, hereby referred to as the CLR family methods [3] (concept lattice-based ranking methods), is that documents in a collection can be organized in a concept lattice according to the common terms that they share. For example, consider a set of 9 documents G annotated using a set of 12 terms M in the formal context $\mathcal{K} = (G, M, I)$ illustrated in Table 1 (white rows). The incidence relation set I indicates by gIm that document $g \in G$ is annotated with term $m \in M$. For this example, let us also assume a user query q_i containing the terms *"arthroscopy"* and *"complication"* (from hereafter we refer to the terms in a query as *keywords*) to be answered with a subset of documents.

In the CLR family approaches, as in many other information retrieval tasks [17], the query is considered as a *virtual object* and it can be included in the formal context as any other object (Table 1, grey row). Therefore, the original formal context is redefined to include the query $q = (q_e, q_i)$, where q_e is the virtual object and $q_i = \{m_1, m_2, ..., m_{|q_i|}\}$ where $m_1, m_2, ..., m_{|q_i|} \in M$ contains its keywords (i.e. the constraints associated to the query) and $|.|$ denotes set cardinality. The formal context is redefined as $\mathcal{K}_q = (G \cup \{q_e\}, M, I \cup \{(q_e, m_j)_{\forall m_j \in q_i}\})$ and its associated concept lattice is computed using a FCA algorithm. The concept lattice computed for the formal context of Table 1 (including the query) is illustrated in Figure 2. After constructing the lattice, the standard procedure in the CLR family approaches is to find the *object concept* of the virtual object q_e. This concept is usually called the *query concept* and it is the starting point to find documents satisfying a query in the lattice [20,19,3].

Let us continue with the above example. The *query concept* for the query with keywords q_i is concept 17 in the concept lattice illustrated in Figure 2. Its intent contains the terms *"arthroscopy"* and *"complication"*. Its extent contains the *virtual object* q_e and documents

[4] A Galois connection is based on a dual adjunction between partially ordered sets.

	patient	laparoscopy	scan	user	medicine	response	time	MRI	practice	complication	arthroscopy	infection
d_1	×	×	×							×		×
d_2			×	×	×	×	×		×	×	×	
d_3		×		×	×			×				
d_4	×				×			×				
d_5				×		×	×					
d_6									×		×	
d_7										×	×	
d_8										×	×	×
d_9											×	×
q_e *										×	×	

* Grey row represents the *query*.

Table 1: A term-document formal context including the query q.

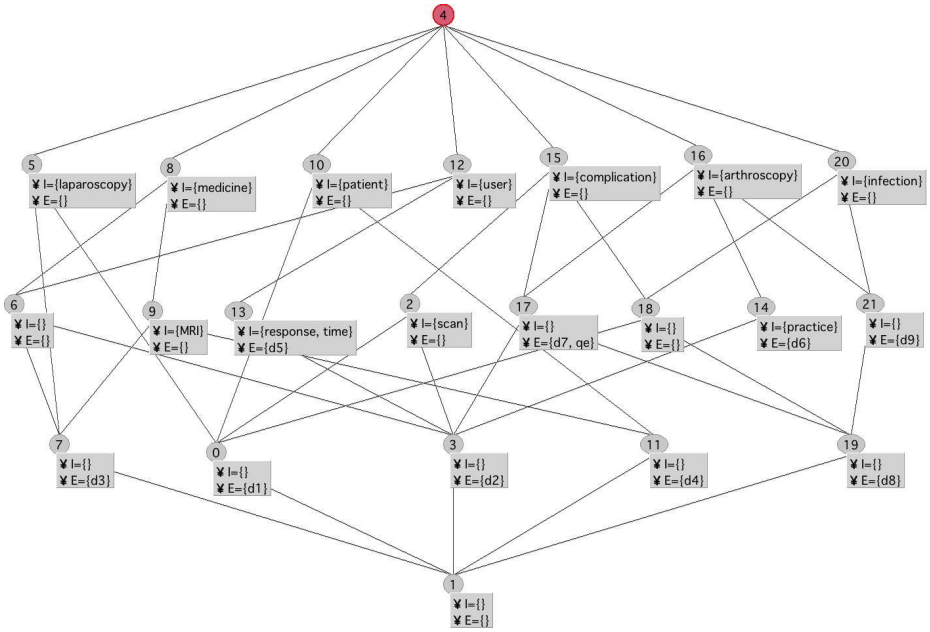


Fig. 2: Concept lattice in reduced notation derived from a document-term formal context including the query. The reduced notation of a lattice consists in labelling the extents/intents only with the first appearance of an object/attribute from top-to-bottom/bottom-to-top (respectively), i.e. objects are show in their *object concepts* and attributes in their *attribute concepts* (e.g. concept 19 is the *object concept* of document d_8 and concept 15 is the *attribute concept* of the attribute "complication").

d_2, d_7 and d_8 which satisfy a *conjunctive* version of query q_i, i.e. these documents include *all* of the query keywords. We refer to these documents as the *exact answer*.

However, very often documents relevant to the user may fail to meet the restrictions of a *conjunctive query*, due to reasons such as language ambiguity (e.g. synonymity or polysemy of terms), poor document descriptions, the lack of user's knowledge about how to effectively pose a question or create a query, etc. This is known as the *non-matching document problem* [3], which refers to the fact that documents relevant to the user query may not always exactly match its keywords and therefore they are not included in the exact answer. To overcome this issue, it is possible to use the lattice to satisfy *disjunctive* versions of the query, i.e. retrieve documents that contain only some of the query terms, by using the super-concepts of the *query concept*. For this example, concept 16, a super-concept of 17, contains in its extent document d_6 and in its intent the term *"arthroscopy"*, while concept 15, also a super-concept of 17, contains in its extent document d_1 and in its intent the term *"complication"*. We say that these documents "partially" meet the query and they provide a *close* or *partial answer*.

As it can be observed from Figure 2, each formal concept in the concept lattice contains a possible conjunctive query and a set of documents which satisfy that query while combinations of formal concepts (in the form of unions) work analogously for disjunctive queries. In fact, the concept lattice configures the *global query space* of the document collection, where the query concept represents the original user query and its super and sub-concepts represent the immediate modifications that can be performed over the query to find "partial-matching documents". Notice that only super-concepts contain different documents than those that could be found on the *query concept*, since sub-concept's extents will always be subsets of the query concept's extent. Following the idea of disjunctive queries, apart from the query concept and its super and sub-concepts, other concepts of the concept lattice may also be found to include some terms of the query and some new terms, and therefore to also represent query modifications. Finally, some other concepts of the lattice do not share any terms with the user query and therefore do not constitute query modifications. It is important to notice that, given that the lattice forms the global query space of the document collection, the retrieval of those concepts that represent meaningful query modifications can be considered as a matter of: 1) *navigating* the lattice starting from the query concept and then 2) *ranking* the retrieved concepts in w.r.t. their relations with the query concept.

2.3.2 Current lattice navigation and ranking approaches

Two main different navigation strategies have been proposed in the literature. The neighbourhood expansion strategy [3] is based on the idea of visiting concepts, in an "expanding ring" order, starting from the query concept. This strategy does not make a distinction between visiting super or sub-concepts, since in the same ring there may be super and sub-concepts of the query concept. The hierarchical exploration strategy [19] navigates the lattice by exploring the super-concepts of the query concept. These super-concepts contain more documents than those found in the query concept thus allowing to work with a disjunctive approach.

Both strategies assume a topological distance measure in order to *rank* the concepts reached by *navigation*. In this work, the topological distance in a lattice is defined as the minimal path length between two given formal concepts (considered as nodes in a graph [28], see also the nearest neighbour relation in [3]). This notion is straightforward in the sense that nearer concepts from the query concept are considered "more related" and hence, they receive a better ranking. However, both strategies differ in that using the neighbourhood expansion it is possible to reach many more concepts within the lattice than in the case of hierarchical exploration.

In terms of *query modification*, the hierarchical exploration strategy works by modifying the original conjunctive query to a set of disjunctive queries represented by the intents of the super-concepts of the query concept. From these super-concepts it is possible to obtain a set of documents used as an answer for the original query. For example, Figure 3 presents the section of a lattice containing 4 concepts including a query concept (concept 3 in white) for the conjunctive query *"arthroscopy"* **and** *"complication"* **and** *"practice"* (notice that in this case the marker q_e is on concept 3) using the hierarchical exploration strategy. Both super-concepts of the query concept (concept 14 and 17) receive the same ranking (they are both at distance 1 from the query concept) and hence the answer is the union of the extents of both super-concepts. Actually, this is the answer for the disjunctive query (*"arthroscopy"* **and** *"complication"*) **or** (*"arthroscopy"* **and** *"practice"*) or more shortly, *"arthroscopy"* **and** (*"complication"* **or** *"practice"*). In this manner, hierarchical exploration searches in the query space for *relaxed* versions of the original query and rank them for how *relaxed* they are (notice that the concept 16 ranked at distance 2 answers the very relaxed query containing only the keyword *"arthroscopy"*).

The notion of query modification is not explicitly present in the neighbourhood expansion strategy since in the same "ring" of topological distance different types of concepts are considered and ranked equally. For example, in Figure 4 the same conjunctive query *"arthroscopy"* **and** *"complication"* **and** *"practice"* is represented along with 6 other concepts obtained through neighbourhood expansion. There are 4 "rings" represented by the topological distances included in arcs between concepts (e.g. ring 1 contains concepts 14 and 17). It can be appreciated that concepts with different intent cardinalities receive the same ranking since they are in the same "ring" (e.g. concepts 16 and 19 in the ring 2). Moreover, it is difficult to assess the modification in the query represented in ring 4 (concept 20) containing the keyword "infection". Nevertheless, this characteristic also gives neighbourhood expansion its potential since it is able to find many more documents than the hierarchical exploration (for example, in hierarchical exploration the concept with the term "infection" is not a possible query modification and document d_1 is never considered as an answer). As such, there is not an actual notion of query modification, but an idea that closer concepts in the lattice will contain closer document descriptions and hence, closer relevant documents.

3 CLAIR - Concept Lattices for Information Retrieval

3.1 Motivation for a new approach for Information Retrieval based on Formal Concept Analysis

As described in the previous section, the main difference between hierarchical exploration (HE) and neighbourhood expansion (NE) strategies is how the notion of *query modification* is applied. Since HE is based on a clear *query relaxation* process where documents are ranked according to how much relaxed is the query they satisfy (w.r.t. the original query), we can expect that the answers it provides, compared to those obtained from NE, are of better quality in terms of relevant documents. On the other hand, since NE is based on a continuous expansion of the lattice region used to retrieve documents, we can expect that the answers it provides contains a larger quantity of relevant documents compared to the answers provided by HE (along with a larger quantity of irrelevant documents).

The trade-off between quality and quantity in document retrieval systems has always been an active issue in the IR domain, mainly reflected by the two most common retrieval

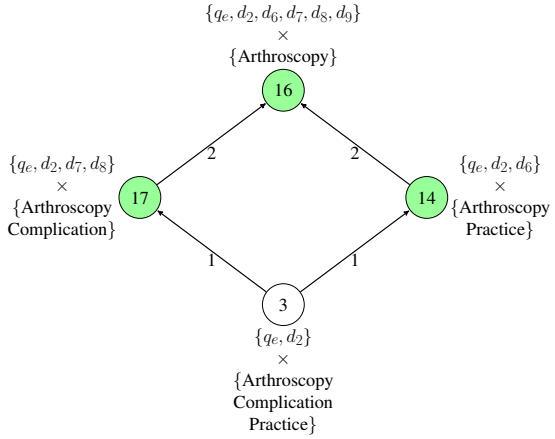


Fig. 3: Section of a lattice showing 4 concepts obtained by hierarchical exploration. Arrows represent the navigation direction with their correspondent topological distance from the query concept of query "arthroscopy", "complication" and "practice" (represented in white).

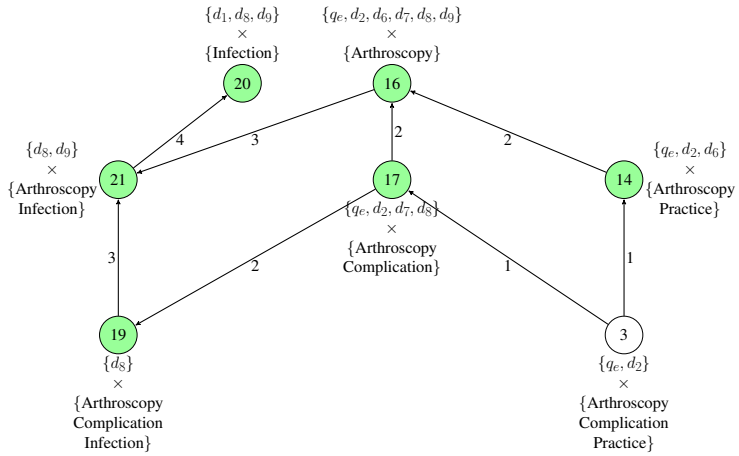


Fig. 4: Section of a lattice showing 7 concepts obtained by neighbourhood expansion. Arrows represent the navigation direction with their correspondent topological distance from the query concept of query "arthroscopy", "complication" and "practice" (represented in white).

evaluation measures: *precision* and *recall* [17] (further described in Section 4.2). Furthermore, it is hard to compare a system with high quality in the answers versus one with high quantity of answers since, in many cases, this depends on the application intended for the system. For example, a beginner student will not be interested in all documents related to a given field but only on those more relevant (interest focus on quality), while an expert will be more interested on monitoring the field trying to find as many documents as he can (interest focus on quantity). Nevertheless, it is accepted that a good retrieval system should have a good quality/quantity balance which is our goal in this work. To achieve this, we

take lessons from HE and NE strategies considering a careful design in the evaluation of the *query modification* process (*ranking*) but also considering an expansion to concepts other than those in the super-hierarchy of the query concept (*navigation*). These two elements are reflected by two of the three aspects of our work, namely classification, navigation and ranking. Regarding navigation, we define a new relation for two given concepts within the lattice which we call *cousin concepts*. Regarding ranking, we consider a semantic-based formal concept similarity measure introduced in [11]. In the following, we describe and detail the three main aspects of our approach called CLAIR (Concept Lattices for Information Retrieval).

3.2 The principles of CLAIR

Our approach, hereby referred to as CLAIR, focuses on the following three aspects.

1. *Classification:* In this work, classification is used with two meanings, namely the *operation* of classification and the *product* of this operation which is also called "classification". Firstly, we use FCA for building a concept lattice which is considered as a semantic index for document retrieval (the concept lattice as a result of a classification operation). Then, given a user query, we rely on the principle of *classification-based reasoning* for inserting the query in the lattice and identifying concepts that provide possible answers to the query (the classification operation applied to the query). This method of query insertion differs from the one used in the CLR family approaches. In CLAIR, as detailed previously, the query concept is not appended to the lattice through an incremental FCA algorithm (e.g. Galois in the case of CLR [2]), but it is classified by the lattice through classification-based reasoning.
2. *Navigation:* We propose a new navigation strategy of the concept lattice which is tailored to the needs of the ranking method proposed using a semantic similarity measure. Navigation is here used in the same sense as in CLR-like approaches, i.e. the identification of relevant concepts given an initial query concept. Our navigation approach is based on the notion of *cousin concepts*. The rationale behind the use of cousin concepts is that in order to identify additional, partial-matching documents we need to modify the original user query but in a manner that the query and the modified query are closely related. We achieve this by the generalization of the query concept in the concept lattice to its super-concepts (which we call *query generators*) and their posterior specialization to what we call *cousin concepts* of the query concept (i.e. the sub-concepts of the query generators). Since query generators are immediate super-concepts of the query concept, cousin concepts retain some keywords (more precisely, those in the query generators) while including some other terms. For example, consider the query concept with intent $\{m_1, m_2\}$ and its query generator with intent $\{m_1\}$. Through the specialization of the query generator we can obtain the concepts with intents $\{m_1, m_3\}$ and $\{m_1, m_4\}$ which are considered as modifications of the original query (i.e. replacing m_2 with m_3 or m_4, respectively).
3. *Ranking:* Since many cousin concepts (or query modifications) can be obtained from the concept lattice for a single query concept, there is a need to evaluate how *close* are these modifications from the original query. For the previous example the question is whether we should replace m_2 with m_3 or with m_4. We answer this by measuring the *semantic similarity* of the terms included in the intent of each retrieved concept w.r.t. the keywords. In this way, we also address the problem of retrieving documents related to the user query in a semantical way, rather than only based on topological characteristics

of the concept lattice. We use a measure introduced in [11] which considers external knowledge sources to evaluate "semantic closeness".

3.3 The implementation of CLAIR as a Knowledge Discovery in Databases process

Following the rationale described above, here we describe the proposed CLAIR approach for document retrieval, which considers *classification*, *navigation* and *ranking* based on the notions of *classification-based reasoning*, *query modification* and *semantic similarity*, respectively. We formulate our approach following the lines of a knowledge discovery in databases (KDD) process [1] which allows us to define well differentiated tasks and provides us with a robust framework to implement the document retrieval process (Figure 5). In particular, our process is defined as a sequential three steps KDD-like process to reflect the three aspects of our work (*classification*, *navigation* and *ranking*). The first step of our approach is "Document Classification", related to the data pre-filtering step of a KDD process. The second step is knowledge "Lattice Navigation" related to the mining/knowledge discovery KDD-process step. The last step of our process is "Concept Ranking", related to the interpretation KDD-process step.

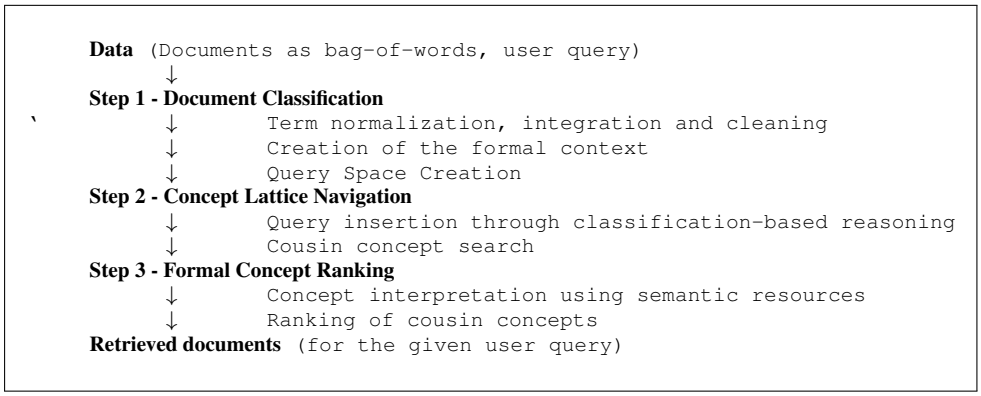


Fig. 5: 3-step KDD-like document retrieval process.

3.4 Step 1 - Document Classification

In this step we construct a formal context $\mathcal{K} = (G, M, I)$ as defined in Section 2.3 consisting of a set of documents G, a set of terms M and the annotations expressed in the incidence relation table as pairs $(g, m) \in I$ where $g \in G$ and $m \in M$. Depending on the nature of the collection of documents, different tasks should be performed in order to construct the formal context (e.g. parsing, tokenizing, stop-word filtering etc. [18]). In order to simplify and standardize the approach, we assume that the documents in the collection are in the form of a bag-of-words (i.e. each document consists of a set of terms). We argue that this is

a safe assumption since most of document corpora are already provided in this format and in the other case, the transformation of text to bag-of-words is a straightforward process. Additionally, a normalization of the terms is required in order to reduce sparsity and integrate the representation of documents. Three basic natural language analysis techniques can be used [18]. *Stemming* is a technique that normalizes a set of words to their morphological root (e.g *retrieval, retrieves, retrieve* are normalized to *retriev*). Thus, it greatly reduces the sparsity and the number of attributes in the context, however it does not maintain the original meaning of the terms (e.g. *retriev* is not an actual word). To maintain the meaning of the terms it is possible to use a *semantic element mapping* which normalizes a set of words to a semantic element definition using an external knowledge source (e.g. *recover, retrieve, find, regain* are mapped to the definition *"Get or find back; recover the use of"*)[5]. This technique reduces sparsity but produces an explosion in the number of attributes in the context, since each term can be mapped to more than one definition (e.g. *recover* maps to 4 different definitions). Finally, through the use of *lemmatization* it is possible to normalize a set of inflected forms of a word (e.g. *retrieval, retrieves, retrieve* are normalized to *retrieve*). Thus, it slightly reduces the sparsity of annotations and the number of attributes in the context, while maintaining the original meaning of the terms.

Given the complexity of calculating a concept lattice and the fact that we need to maintain the original meaning of the terms in order to measure their in-between semantic similarity, we normalize document terms using the technique of word lemmatization.

Finally, a concept lattice representing the query space is created based on the document-term formal context obtained from the previous step. Different algorithms exist to compute a set of formal concepts from a formal context and to build a concept lattice [15]. For this work, we rely on the AddIntent algorithm [30] because of its performance.

3.5 Step 2 - Concept Lattice Navigation

The second step corresponds to navigating the constructed "document index" or "query space"[6] in order to retrieve documents for a given user query. For convenience, we propose a model-based document retrieval approach, i.e. the document index is built a-priori and not for each given user query like in usual CLR-like approaches (described in Section 2.3). This is done given the complexity of building a concept lattice from a formal context of significant size. Instead, in our approach the concept lattice is constructed once and the query is simply inserted in the lattice when required.

3.5.1 Query insertion through classification-based reasoning

This sub-step assumes the existence of the concept lattice and a user query in the form of a set of keywords. Its output is a query concept and set of related formal concepts which are used to retrieve a set of documents. A user query q is considered as a *query concept* $q = (q_e, q_i)$ where q_e is a "dummy variable" to be instantiated by retrieved documents and $q_i = \{m_1, m_2, ..., m_{|q_i|}\}$ is a set of keywords.

The query concept is not "inserted" in the lattice, but rather "classified" by it using *classification-based reasoning* as introduced in [22]. Classification-based reasoning is based

[5] Dictionary definition of the first sense of *recover* given by Wordnet.

[6] Notice that "document index" or "query space" are both a dual view of the same concept lattice from the point of view of extents or intents, respectively. Here after we refer equally to "document index", "query space", "concept lattice" or "semantic index".

```
1 function compare (q, C)
2     if C is marked
3       then return {}
4     else mark C
5     if C does not subsume q
6        then return {}
7     else MSS ← {}
8       for every descendant D of C do
9         MSS ← MSS ∪ compare (D, q)
10    if MSS = {}
11       then return {C}
12    else return MSS
```

Fig. 6: Classification-based Reasoning algorithm: Searching for the most specific subsumers of q: comparison of the current object C with q.

on a depth-first traversal of the lattice and consists in, given a query concept q, searching for the *most specific subsumers (MSS)* and the *most general subsumees (MGS)* of q. Actually, the search for the most general subsumees here is useless. The search for the most specific subsumers is illustrated in the algorithm in Figure 6 and works as follows. The classification-based reasoning algorithm receives a concept to classify q and a MSS candidate concept C (line 1). C is firstly checked for if it was previously visited and if not, then it is marked (lines 2-4). This is done to ensure that concepts are visited only once. A subsumption test is performed for checking whether C subsumes q, where the subsumption test relies on intent inclusion: C subsumes q as long as the intent of C is included in the intent on q (line 5). If C does not subsume q, the sub-lattice rooted in C is cut and no more considered (line 6). If C subsumes q, a recursive call of the classification-based reasoning algorithm is called over the sub-lattice rooted in C (line 7-9). The traversal continues with the first descendant of C and so on in the same way. The traversal ends when there are no more concepts to visit and returns the set MSS of most specific subsumers. In the case that no MSS were found in the sub-lattice of C, then C becomes a MSS (lines 10-12) In the present case, these most specific subsumers are called *query generators*.

In case there exists in the lattice a formal concept (A, B) such as $B = \{m_1, m_2, ..., m_{|q_i|}\}$, then the algorithm identifies (A, B) as the query concept, and those documents in A constitute the *exact answer*. The existence of an *exact answer* is not always guaranteed, especially for large and complex queries. The worst case scenario appears when no *query generators* are found except for the top concept of the lattice (which includes all the documents and no terms). Actually, this can only be the case if no keyword provided by the user can be found in the query space and in this case, the query is considered to be *unsuccessful*.

As we have previously described, in a concept lattice, for a user query represented by a query concept, query generators represent "relaxed versions" of the original user query, i.e. they include less keywords than the query intent. Any other sub-concept of a query generator –except the *query concept*– induces a modified user query, since it includes a part of the query determined by the query generator, plus a set of terms that are not contained in the original query. Based on this observation, we introduce a *navigation strategy* allowing to find "successful modifications" of an original user query, i.e. they do include answers, which are closely related one to the other. In order to find and reuse these query modifications, we define hereafter the notion of *cousin concepts*.

3.5.2 Cousin Concept Definition:

Two formal concepts (A_1, B_1) and (A_2, B_2) which are not comparable for $\leq_{\mathcal{K}}$ are said to be *cousin concepts* iff there exists $(A_3, B_3) \neq \top$ such that:

- $(A_1, B_1) \leq_{\mathcal{K}} (A_3, B_3)$.
- $(A_2, B_2) \leq_{\mathcal{K}} (A_3, B_3)$.
- $d_{\mathcal{K}}((A_2, B_2), (A_3, B_3)) = 1$ and $d_{\mathcal{K}}((A_1, B_1), (A_3, B_3)) = 1$.

where $\top$ is the top concept and $d_{\mathcal{K}}$ measures the "topological distance" between two formal concepts in the lattice $\mathcal{K}$. The distance $d_{\mathcal{K}}$ is analogous to the "minimal path length" for two nodes in a graph as defined in [28]. Intuitively, this means that (A_1, B_1) and (A_2, B_2) do not subsume each other and that (A_3, B_3) is the upper bound $(A_1, B_1) \sqcup (A_2, B_2)$. Actually, (A_3, B_3) represents a *query generator* of queries (A_1, B_1) and (A_2, B_2). This also means that the query in (A_1, B_1) is considered as a modification of the query in (A_2, B_2) and vice versa. We restrict *query generators* not to be the top concept, since the empty query can be considered as the generator of the whole *query space*. For example, in Figure 2, concept 18 is a cousin of 17 because of concept 15, concept 6 is a cousin of 13 because of concept 12 and so on. However, concept 10 is not a cousin concept of 12 since that would mean that the query generator should be concept 4 which is the top.

For a given query and its *query concept*, the querying process aims at traversing the lattice to extract all its *cousin concepts* (A_i, B_i). For simplicity reasons we have restricted the cousin concepts to be at a topological distance 2 from each other. This restriction can be relaxed in order to increase the number of documents retrieved by the process if necessary.

As an example of the above-described *lattice navigation* strategy, consider Figure 7, which contains part of the lattice in Figure 2. Specifically, Figure 7 displays concepts 14, 16, 17 and 21, where concept 17 (in white) represents the *query concept* for the query with keywords "arthroscopy" and "complication". Its extent contains the *exact answer*, i.e. documents d_7 and d_8. Concept 8 contains in its intent only "arthroscopy"and provides a relaxed version of the original query and works as a *query generator*. From this concept we can obtain the cousin concepts of concept 17, i.e. concepts 14 and 21. These provide two different query modifications where the term "complication" in the original query is replaced with the terms "practice" or "infection", allowing to choose whether document d_2 and d_6 or document d_9 should be ranked first. The decision on the ranking of the retrieved concepts, and the documents that they include is a matter of *interpretation* of the results and it is described in the following step.

3.6 Step 3 - Formal Concept Ranking

Given the query concept and its *cousin concepts*, the output of a concept ranking process is a sorted list of documents retrieved to the user. As we recall from the previous step, a cousin concept represents both a query modification (in its intent) and a set of documents that satisfy that modification (its extent). In this step we interpret these query modifications in the sense of semantic similarity w.r.t. the original user query considering that those modifications which deviates less from the original query (and hence, are more similar) should yield documents more relevant. To achieve this, we use a semantic similarity measure defined for two formal concepts within a concept lattice.

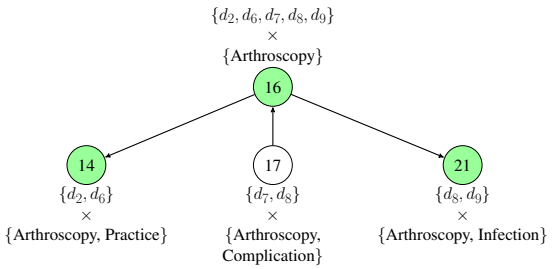


Fig. 7: Example of lattice navigation. Starting from the query concept (17) we navigate to its cousin concepts (14,21) and retrieve documents d_2, d_6 and d_9. Concepts are shown with their extents and intents. Arrows show the direction of the navigation inside the lattice.

3.6.1 Computing the similarity between concepts

In our framework, the ranking of the candidate concepts is performed using the semantic similarity metric proposed by Formica [11]. Given two formal concepts $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$ the similarity between C_1 and C_2 is defined as:

$$sim(C_1, C_2) = \frac{|A_1 \cap A_2|}{max(|A_1|, |A_2|)} * w + \frac{\mathcal{M}(B_1, B_2)}{max(|B_1|, |B_2|)} * (1 - w) \quad (2)$$

where $0 \leq w \leq 1$ is a weighting parameter and $\mathcal{M}(B_1, B_2)$ is the maximization of the sum of the *information content* similarities between each possible pair of terms created using one term from B_1 and another from B_2. *Information content* similarity between two terms is measured using their distance in a lexical hierarchy and/or their co-occurrence in a text corpus (see Section 2.1).

As an example of how concept ranking is performed using Formica's similarity, let us consider Figure 7. Given the query concept 17, we navigate the lattice and find two cousin concepts, i.e. concept 14 (containing documents d_2 and d_6) and 21 (containing document d_9). We need to rank these concepts, in order to decide the order in which the retrieved documents will be presented to the user. In order to do so, we compare the similarity of each of the cousin concepts, to the query concept, using Formica's semantic similarity metric defined in Equation 1 with $w = 0.5$, Wordnet as the external lexical hierarchy and the Brown corpus as the body of text to locate term frequencies. We observe that $sim(17, 21) = 0.6137$, while $sim(17, 14) = 0.225$, because the pair *(complication, infection)* has a higher semantic relation than the pair *(complication, practice)*, as explained in Section 2.1, and the intersection between concept 17 and 14 is empty, while for 17 and 21 the intersection contains one element. Therefore, we may rank and retrieve document d_9 before documents d_6 and d_2. Differentiations in the weight value w allow for differentiation in the preference between the structural (from the extents) and the semantic (from the intents) similarities of the compared concepts.

4 Experimental Evaluation

To test the capabilities of our approach, we applied it on four datasets of the SMART collection[7] which is a well known benchmark collection used in text mining and retrieval. Each dataset is provided as a collection of documents from different domains. Additionally, a list of queries (in different formats) is given for each dataset. A query has an associated set of valid answers, i.e. documents that have been labelled by human experts to answer this query.

4.1 Experimental setting

All datasets are preprocessed by parsing, stop word removal and lemmatization using the natural language toolkit (NLTK) library[8] for Python. Table 2 details each dataset in terms of the number of document, terms, annotations (the number of document-term relations), formal context density (#annotations / (#documents $\times$ #terms)) and the number of queries with provided answers used in the experiments.

For each dataset a formal context containing all documents and lemmatized terms is created and a concept lattice is derived from it using an AddIntent algorithm implementation [30]. Preprocessed datasets are stored in a relational database for further operations. Concept lattices were modelled as directed graphs using the networkX library[9] for handling large graphs. Each received query is processed to construct a bag-of-words using its lemmatized keywords. A query concept is created including all the lemmatized keywords in its intent and an empty extent. Using classification-based reasoning we look for the *query generators* of the query concept. To compute Formica's similarity, the query concept extent is considered as including the *union of the extents of all its query generators*. This heuristic greatly improves the performance of the posterior ranking in the four datasets used. We provide a further explanation in the following discussion. The sub-concepts of the *query generators* (cousin concepts) are ranked using Formica's similarity measure described in Section 3. Finally, a list of sorted documents is created using the extent of the cousin concepts. A document is only inserted in the list once, from the cousin concept with the highest rank. The order of the documents inserted in the list from the same cousin concept is disregarded.

Name	#documents	#terms	#annotations	Ctx. density	# queries
CISI	1460	8169	68827	0.05%	30
CACM	3204	7466	67502	0.2%	53
MED	1033	11207	57370	0.4%	26
CRAN	1398	5964	77743	0.9%	100

Table 2: Dataset characteristics

In order to compare the results of our approach, we have implemented three retrieval methods, namely exact matching, BM25 and CLR which work directly on the databases yielded from the preprocessed dataset. The *exact matching* (EM) method is a naive approach which searches the database for documents with at least one keyword provided in the query. Additionally, documents are ranked according to how many terms they have in common

[7] ftp://ftp.cs.cornell.edu/pub/smart/

[8] `http://nltk.org`

[9] `http://networkx.github.com`

w.r.t. the query. Documents with more terms in common are ranked first. The BM25 function [17] (also known as Okapi BM25) uses a probabilistic approach to rank documents considering collection size and document length normalization. Each document is *scored* w.r.t. the query using a modified and parametric version of term-frequency and inverse-document-frequency (TF.IDF). BM25 is within the BM (Best Match) family of retrieval methods that are less restrictive than EM methods. CLR (concept lattice-based ranking) is a standard lattice-based approach presented in [3] and explained in Section 2.3.

4.2 Evaluation measures

In the following, we provide a description of the evaluation measures used in this work as described in [17]. Precision and Recall are measures to assess the relevance of documents answered by a retrieval system. Formally, given a query q, we define precision and recall in Equations 3 and 4 (respectively) where the set $retrieved$ contains all documents found for q and the set $relevant$ contains the documents which constitute the actual answer for q (ground truth). The set $positive = |retrieved \cap relevant|$ represents the correct subset of the retrieved documents for query q.

$$precision(retrieved) = \frac{|positive|}{|retrieved|} \tag{3}$$

$$recall(retrieved) = \frac{|positive|}{|relevant|} \tag{4}$$

The $relevant$ set (or "ground truth") is usually constructed by a single or a group of domain experts which are able to distinguish within the document collection which are the relevant documents for a given query (sometimes checking documents one by one). Having the ground truth, the calculation of precision and recall is straightforward. For example, consider the query with keywords "arthroscopy" and "complication" answered by CLAIR with documents shown in list 1 at Table 3 as to be the $retrieved$ set (example illustrated in Figure 7). The $relevant$ set is defined in list 2 at the same Table. From this we obtain that CLAIR is able to retrieve 3 out 4 documents that domain experts considered relevant and as such, our query was answered with a recall of 0.75. However, along with those 3 relevant documents, our system also found 2 documents not considered by domain experts which are regarded as mistakes. Hence, our system finds 3 correct documents out of 5 which yields a precision of 0.6.

List	Set Name	Documents	Precision	Recall
1	Ground Truth	d_4, d_6, d_7, d_9	-	-
2	System answer	d_2, d_6, d_7, d_8, d_9	0.6	0.75
3	Low Quality/High Quantity	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9$	0.44	1
4	High Quality/Low Quantity	$_7$	1	0.25
5	System answer (ranked)	d_7, d_8, d_9, d_2, d_6	0.6	0.75

Table 3: Examples of precision and recall

In a nutshell, *precision* measures the proportion of correct answers among the answers found by a retrieval system. Hence, precision takes into consideration also the incorrect answers of the system. On the other hand, *recall* measures the proportion of correct documents

Row	Document	Relevant?	Precision	Recall
1	d_7	✓	1	0.25
2	d_8	✗	0.5	0.25
3	d_9	✓	0.66	0.5
4	d_2	✗	0.5	0.5
5	d_6	✓	0.6	0.75

Table 4: Values of precision and recall calculated considering the first 1,2,3,4 and 5 ranked elements in list 5 at Table 3. Precision and recall are calculated with a list which considers all previous documents in decreasing order of ranking (for example, precision and recall in the third row were calculated for documents d_7, d_8 and d_9. Columns "Relevant?" shows if the document in the corresponding ranking is relevant (in list 1 at Table 3).)

over the whole collection of correct answers for a given query and does not consider incorrect answers. Precision and recall are often considered as a trade-off between the quality and the quantity of the answers where a high quality is achieved with a high precision value, and a high quantity of answers is achieved with a high recall value. Actually, a high recall value for a given user query can be easily achieved by retrieving the whole set of documents in the collection. However, this comes at the cost of a low precision (low quality/high quantity). For example, consider in Table 3 list number 3 with recall of 1 and precision of 0.44. Inversely, it is easy to achieve a high precision by retrieving a few documents which are likely to be correct at the cost of a low recall (high quality/low quantity). In Table 3, this is the case of list 4 with precision of 1, but recall of 0.25.

As discussed in Section 3.1, it is well accepted that a good retrieval system should maintain a balance between quality and quantity (precision and recall), however if we want to compare CLAIR w.r.t. other systems, we need something more robust to draw conclusions. Another aspect of precision and recall is that they do not take into consideration the document ranking in the answer of a retrieval system. For example, consider in Table 3 lists 2 and 5 (not ranked and ranked, respectively) which have the same values of precision and recall. Thus, we not only need an evaluation for the answers but also for the ranking applied to the answers.

Regarding these drawbacks, some other evaluation measures have been proposed considering precision and recall in the context of ranking. Table 4 shows the ranked answer in list 5 at Table 3 where precision and recall were calculated considering only the first 1, 2, 3, 4 and 5 documents of the ranking. We can appreciate that different values of precision are obtained depending on how many documents are considered in the answer (usually, because of the quality/quantity trade-off, the tendency is that as more documents are considered, less precision is achieved).

In the following, we define the evaluation measures for a ranking process considered in this work. These measures are used as standard IR evaluation techniques and their rationale is out of the scope of this article [17].

Considering the precision and recall values in Table 4, we define the *interpolated precision* at a recall interval as the maximum value of precision for a given interval of recall. For example, the interpolated precision at the recall interval $[0.5, 1]$ is the maximum among $0.66, 0.5, 0.6$ (for rows 3,4 and 5 in Table 4, respectively), and hence is 0.66. Similarly, the interpolated precision in the recall interval $[0, 0.5]$ is 1. The *interpolated average precision* (IAP) is the mean of the interpolated precisions for all recall intervals. In this example, is the mean of the interpolated precisions for $[0, 0.5]$ and $[0.5, 1]$ and hence, it is 0.83. For

two intervals, we say 2-point interpolated average precision or IAP@2. The usual approach considers 11 intervals and it is called 11-point interpolated average precision or IAP@11.

Analogous to IAP, we calculate the *mean average precision* (MAP) which is the mean of the precision values in Table 4 where the column "Relevant?" is marked as correct (✓). This is, it only considers the precision values where the last document of the list (d_x) is a relevant document[10]. The MAP in this example is the average of the precision values of rows 1,3 and 5 which is 0.753.

To formalize these measures, let us define the set $rank = \{d_1, d_2, ..., d_n\}$ as the set of documents answered by a retrieval system for a given query sorted by ranking in a descending manner . We define $rank^{d_x} \subseteq rank$ as a sub-list of $rank$ which contains from the first element until element $d_x \in rank$. Equation 5 is the interpolated precision in a given interval defined by the edges r_1 and r_2. Equation 6 defines the 11-point interpolated average precision and Equation 7 describes the *mean average precision* as used in this work.

$$ip(r_1, r_2) = \underset{\forall d_x \in rank}{\operatorname{argmax}} \{precision(rank^{d_x}) \iff recall(rank^{d_x}) \in [r_1, r_2[\} \quad (5)$$

$$IAP@11 = \frac{\sum_{i=0}^{10} ip(\frac{i}{10}, \frac{(i+1)}{10})}{11} \quad (6)$$

$$MAP = \frac{\sum_{\forall x \in positive} precision(rank^x)}{|positive|} \quad (7)$$

$$(8)$$

Finally, in this work we also provide the precision calculated in the first five documents of the ranking or P@5. This is not a measure that evaluates the ranking, but it gives an insight on the practicability of the approach, given that users tend to evaluate the retrieved set of documents by the relevance of the few first elements in the ranking. In the case of Table 4, the P@5 is given by the precision on list 5 and it is 0.6.

4.3 Results

Table 5 shows the results for 3 measures, namely interpolated average precision at 11-points (IAP@11), mean average precision (MAP) and precision in the first 5 ranked documents (P@5) on the four datasets and the four approaches. Values in boldface indicate the best value obtained for an approach for each dataset. From these results it can be appreciated that CLAIR surpasses the other three approaches with a score of CLAIR: 8, CLR: 1, EM: 3 and BM25: 1 (the tie for P@5 in the CRAN dataset is considered as a point for CLR and EM). CLAIR always win in the values of IAP and MAP which are actually the measures that consider document positions and same precision/recall conditions, whereas it never wins in the precision in the first five ranked documents.

This can be explained by the manner used by CLAIR to rank documents according to the semantic similarity among the terms shared by a subset of documents and the terms in a query. A subset of documents may have a few terms in common w.r.t. the query, but several *similar* terms which will rank them high in the retrieved list compared with documents with more terms in common w.r.t. the query but a few *similar* terms. It seems that having more

[10] To be strictly correct, this is called the "average precision" (AP), while MAP is the mean of AP over a set of queries. Since we are presenting all of our results averaged over a set of queries, in this work we refer to AP as MAP.

terms in common w.r.t. the query is more important than having more *similar* terms. On the other hand, most of the documents do not have many terms in common w.r.t. the query. This means that approaches like EM and BM25 are very good at ranking a few documents (those that have more terms in common with the user query) and bad at ranking many documents where their discriminatory power is low (those that have more similar terms rather than the same terms in common with the user query). The discriminatory power of CLAIR does not decrease in this manner making it robust w.r.t. documents with different terms than those of the query, mainly by the use of semantic similarity among terms.

This is further supported by Figures 8 which present the interpolated precision at 11 different recall points. It can be seen that, with exception of the CRAN dataset, EM always has a best value in the first point 0.0, while CLAIR quickly surpass EM (and the other approaches) for the rest of the recall points.

Dataset	MED				CACM			
Approach	CLAIR	CLR	EM	BM25	CLAIR	CLR	EM	BM25
IAP@11	**0,5451**	0,4619	0,5116	0,5015	**0,2756**	0,1504	0,2135	0,0848
MAP	**0,5029**	0,402	0,4686	0,4804	**0,2524**	0,1697	0,1939	0,0724
P@5	0,48	0,336	0,5524	**0,5714**	0,1608	0,0769	**0,2154**	0,1038
Dataset	CISI				CRAN			
Approach	CLAIR	CLR	EM	BM25	CLAIR	CLR	EM	BM25
IAP@11	**0,3527**	0,2978	0,17	0,1762	**0,0279**	0,0199	0,0154	0,0181
MAP	**0,3234**	0,259	0,1444	0,1499	**0,0262**	0,0181	0,013	0,0159
P@5	0,2303	0,1886	**0,28**	0,2571	0,0122	**0,0043**	**0,0043**	0,0106

Table 5: Measures for each domain and each approach.

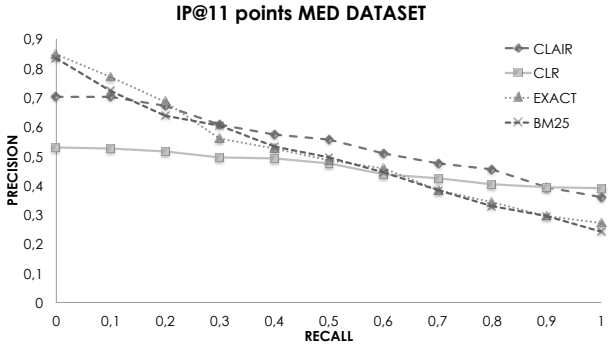
Finally, regarding the use of Formica's similarity, in these experiments we did not seek for an optimal value of w since our goal was to show the feasibility of our approach and to prove that better or similar results could be obtained with CLAIR compared to standard IR techniques. Nevertheless, it is worth mentioning that for most of the queries (specially the large ones), the query concept extent is empty, i.e. it is very hard that a document contains exactly all the terms provided in the user query. In these cases the left (additive) term of Formica's similarity is not informative (equal to 0). To avoid this, we use the heuristic of considering the query concept extent as containing the union of all its query generators' extents. This has shown to greatly improve the results in all the four datasets used in the experiments.

4.4 Query analysis

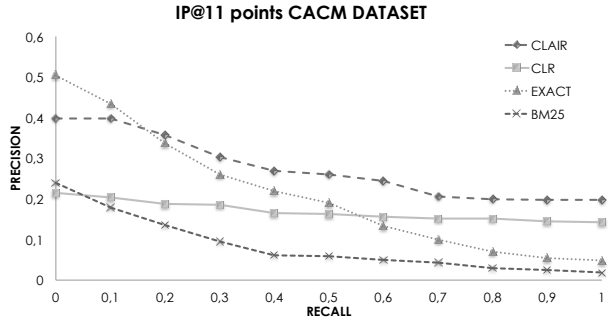
In the following we present a brief analysis for some queries of the CISI dataset in order to provide a deeper understanding on the how CLAIR is able to obtain better results than the rest of the approaches and describe further improvements for the approach.

The sunny case of query 6: Query 6 contains the terms *communication, verbal, possibility, word, computer* and *human*, however it is only mapped to one relevant document (out of 1460). Our approach is able to find 30 documents in the query generators shown in Table 6.

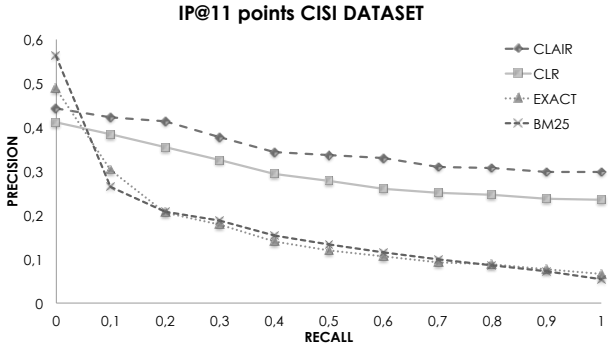
These three query generators led us to 35 different cousin concepts (query modifications) from which the top 10 in the ranking are shown in Table 7 where the query concept is also



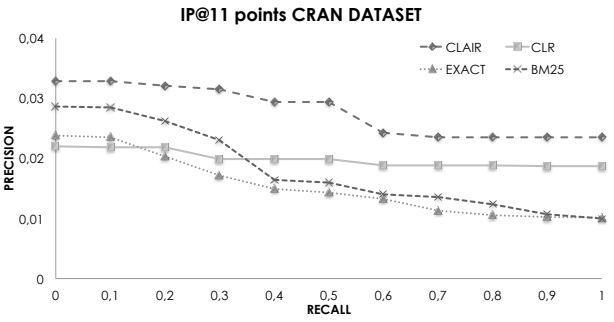
(a) MED dataset



(b) CACM dataset



(c) CISI dataset



(d) CRAN dataset

Fig. 8: 11-point interpolated precision for each dataset

ID	Intent
$G6.1$	word, computer, human
$G6.2$	computer, communication
$G6.3$	possibility, human

Table 6: Query generators for query 6.

Formica Sim	Intent	Extent support
1	communication, verbal, possibility, word, computer, human *	0
0.637	word, computer, human, information	4
0.634	machine, word, experiment, based, computer, human, text	3
0.603	research, word, computer, human	3
0.602	index, word, computer, human	3
0.595	word, computer, make, human	3
0.493	concept, possibility, human, analysis	3
0.484	computer, form, communication	4
0.470	possibility, human, information	3
0.470	computer, information, communication	15
0.449	computer, part, communication	4

* Grey row represents the *query concept.*

Table 7: Ranked results for query 6.

illustrated in grey. The concept with the highest similarity to the query concept (second row) contains in its extent 4 documents including the only correct answer for the query. In this case, the query modification is created by replacing the term *communication* (in the sense of "Something that is communicated by or to or between people or groups") by the term *information* (in the sense of "A message received and understood"). Thus, thanks to the search through cousin concepts, the system is able to find this unique relevant document showing the capabilities of our approach.

The infamous case of query 8: The case of query 8, consisting of the terms *language, indexing, information, retrieval* and *science*, is of special interest since we are able to find 76 different query modifications with very high similarity values w.r.t. the original query (shown in Table 9) leading us to 31 documents which include none of the possible 18 correct answers.

The problem is due to the fact that the query generators (in Table 8) do not contain any correct answers making any possible query modification useless. One possible way to overcome this issue corresponds to the inclusion of more documents by relaxing the definition of cousin concepts allowing query generators to be at a distance 2 of the query concept, however this induces an explosion on the query modifications obtained from the lattice (from 76 to 504) and a consequent lower performance of the document retrieval process.

ID	Intent
$G8.1$	language, information, retrieval, indexing
$G8.2$	science, information, retrieval, indexing
$G8.3$	science, language, information, indexing

Table 8: Query generators for query 8.

Formica Sim	Intent	Extent cardinality
1	language, indexing, information, retrieval, science *	0
0.987	science, word, information, retrieval, indexing	2
0.977	subject, language, information, retrieval, indexing	5
0.977	language, field, information, retrieval, indexing	5
0.972	science, information, term, retrieval, indexing	4
0.959	science, subject, information, retrieval, indexing	4
0.936	method, language, information, retrieval, indexing	7
0.927	method, science, information, retrieval, indexing	4
0.926	system, language, information, retrieval, indexing	13
0.921	science, research, information, retrieval, indexing	4
0.916	concept, language, information, retrieval, indexing	3

* Grey row represents the *query concept.*

Table 9: Ranked results for query 8.

Finding more documents: As stated above, it is possible to increase the *recall* of our approach (the number of relevant documents retrieved over the total number of relevant documents) by relaxing the definition of cousin concepts allowing the query generators to be at distances farther than 1 from the query concept. However, this comes with a great cost in terms of computation since the number of query modifications obtained from the query space (the concept lattice) which should be compared to the query concept will greatly increase. It also impacts negatively the precision of the answers (the number of relevant documents retrieved over the total amount of documents retrieved).

Applicability: It is worth mentioning the applicability of our approach given the limitations in the computation of a concept lattice. With the state-of-the-art FCA algorithms, it is uncertain that CLAIR may be applied in document collections such as the entire Web or even some subsets of it proposed as standard datasets for testing IR tasks[11]. Indeed, the applicability of CLAIR is restricted to smaller datasets, usually personal data collections where the number of documents is not larger than 100.000 documents, such as personal picture collections, research references, mail archives, music albums, etc. These datasets share three characteristics: they are real-life datasets, they are numerous since mostly any person creates several of them; and more importantly, there is a real necessity to improve the performance of searching within them.

5 Related Literature

Formal concept analysis (FCA [12]) is a data representation, organization and management technique with applications in many fields of information science, ranging from knowledge representation and discovery to logic and description logics [27]. In the past years, FCA has been applied to document indexation (an Information Retrieval task [17]) since it proposes a robust and formal framework to exploit the relations that documents (objects) have through the terms they share (attributes). The capabilities of FCA in the standard IR model are numerous and range from query representation and expansion, document browsing and ranking to faceted navigation and visualization [29]. For example, the work of Priss [26] uses concept lattices to improve the representation of a document collection by merging

[11] The TReC competition provides several datasets for different IR tasks. Some of them contain billions of documents. `http://trec.nist.gov/`

it with information from thesauri and thus creating a multi-faceted extended context. In a similar approach, the work of Carpineto et al. [5] presents CREDO, a system that queries Google to construct a faceted browser from a concept lattice to help the user on its search experience.

Apart from browsing support and automatic facet construction, some approaches use the concept lattice directly as a document index and propose different strategies to navigate it in order to find those documents relevant for a given user query. In general, given a document-term concept lattice and a conjunctive user query, these approaches work by identifying a formal concept in the lattice that best represents the query. Other than the approaches based on hierarchical exploration [20] and neighbourhood expansion [3] deeply described in Section 2.3, several systems have been proposed for the exploitation of concepts lattices for IR. For example, in [9], the author presents the system Camelis as a logical information system for the organization, indexation and retrieval of different types of documents. Camelis allows searching using metadata elements of documents as their attributes in a "logical context". The tool also supports browsing, navigation and the inclusion of external information sources as automatic face recognition on images.

In [6], the authors present SearchSleuth as a document organization and search system using a concept lattice built from the results obtained from a Web search engine (like CREDO, but with Yahoo instead of Google). It is based on three different techniques, namely *generalization*, *specialization* and *categorization*. The later is based on looking for the *sibling concepts* of the query concept which, as *cousin concepts* are direct sub-concepts of the direct super-concepts of the query concept, with the additional restriction that they also should be direct super-concepts of the direct sub-concepts of the query concept. In this approach, concept's rank is calculated through the average of the Jaccard similarity between the extents and the intents of the query concept and its siblings concepts (as in [20]). It is worth mentioning that Formica's similarity is also based in the Jaccard similarity of the extents but it uses an optimization approach to find the best semantically similar terms within the intents of two concepts. Semantics are not considered in SearchSleuth, nor in its predecessor ImageSleuth [7].

Other systems worth mentioning are FooCA [13], which presents the user with a formal context of documents and terms obtained from a Web search engine. The user manually filters and manages the formal context to obtain a concept lattice for navigation and browsing in a posterior step. CreChainDo [23] is presented as a concept lattice-based IR system designed to support explicit relevance feedback from users. In this case, the user is presented with the intent of the query concept and he can manually add or remove terms (relevance feedback) which in turns allows him to navigate the lattice or to recalculate it (in the case he adds a terms which is not in the lattice).

A thorough study on the incidence of information retrieval focused articles in the domain of formal concept analysis can be found in [25]. To the authors knowledge, this is the first work engaged in the use of semantics on a FCA-based IR system. An exception worth mentioning would be the extension of Camelis (called Camelis 2 or Sewelis [10]) which deals with faceted search and SPARQL[12]. While we refer to "semantics" as the actual meaning of terms, "semantics" in the "semantic web" are given by a collection of "semantic definitions" called "ontologies" or "meta-schemas' and hence, they are not the same (ironically, in this case the term "semantics" also has different *semantics*). Finally, this extension of Camelis is based on the definition of a language to query RDF[13] graphs instead of lattices.

[12] SPARQL is the query language definition for linked data in the semantic web domain.

[13] Resource Description Framework

6 Conclusions

In this paper we present a concept lattice-based for information retrieval approach (CLAIR) to support semantic indexing and document retrieval. We define our approach as a knowledge discovery in databases-like (KDD-like) process involving the formal concept analysis framework (FCA).

The proposed process uses the concept lattice, constructed from a document-term formal context, as the semantic index of the document collection. Given a user query, it applies a classification-based reasoning algorithm to insert into the lattice a new formal concept corresponding to the query (the *query concept*) and to find a set of *query generators* corresponding to disjunctive versions of the query. The concept lattice is thus considered as the *query space* for the document collection, where each formal concept is a possible query and consequently, the *query generators* are used to obtain *modifications* of the original query. These modifications are used to find partial-matching documents. The navigation from the original query concept to the concepts containing the modified queries is achieved using the notion of *cousin concepts*, which is another original aspect of this work. Finally, external knowledge sources are used to measure the "semantic closeness" of the query modifications with the original user query.

We validate the feasibility and capabilities of our approach by applying it on four classical datasets used in information retrieval and by comparing it to three different IR techniques. We discuss the flexibility of the proposed process, which allows the improvement of the *recall* of the answers by changing one single parameter of the navigation strategy.

There are several directions for future investigations. The first one is related to the improvement of the performance of the approach, using parallelization and computer clusters. Algorithmic adaptations are then needed. This will also allow to draw some comparisons with different methods used in the information retrieval field, as well as to experiment with different datasets. A second one is related to the data structure. Mining complex datasets rather than binary formal contexts allows us to consider weighted *document* $\times$ *term* tables and a more sophisticated document indexing. In this way, we are planning to adopt the use of pattern structures [14] to improve the present approach. Finally, the present approach could also be coupled with more numerical approaches for semantic retrieval, such as latent semantic analysis [8]. It could be interesting to study and build the basis for such a hybrid semantic retrieval approach.

7 Acknowledgements

The work of Ioanna Lykourentzou in the present project is supported by the National Research Fund, Luxembourg, and cofunded under the Marie Curie Actions of the European Commission (FP7-COFUND). The work of Victor Codocedo is part of the Quaero Programme, funded by OSEO, French State agency for innovation.

References

1. Ronald J Brachman and Tej Anand. The Process of Knowledge Discovery in Databases. In *Advances in Knowledge Discovery and Data Mining*, pages 37–57. 1996.
2. C. Carpineto and G. Romano. GALOIS : An order-theoretic approach to conceptual clustering. In *Proceedings of the 10th International Conference on Machine Learning (ICML'90)*, pages 33–40, July 1993.

3. Claudio Carpineto and Giovanni Romano. Order-theoretical ranking. *Journal of the American Society for Information Sciences*, 51(7):587–601, May 2000.
4. Claudio Carpineto and Giovanni Romano. *Concept Data Analysis: Theory and Applications*. July 2004.
5. Claudio Carpineto, Giovanni Romano, and Fondazione Ugo Bordoni. Exploiting the potential of concept lattices for information retrieval with credo. *Journal of Universal Computer Science*, 10:985–1013, 2004.
6. Jon Ducrou and Peter W. Eklund. Searchsleuth: The conceptual neighbourhood of an web query. In *CLA*, volume 331 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
7. Jon Ducrou, Björn Vormbrock, and Peter Eklund. Fca-based browsing and searching of a collection of images. In *Conceptual Structures: Inspiration and Application*, volume 4068 of *Lecture Notes in Computer Science*, pages 203–214. Springer Berlin Heidelberg, 2006.
8. S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88*, pages 281–285, New York, New York, USA, May 1988. ACM Press.
9. S. Ferré. Camelis: a logical information system to organize and browse a collection of documents. *International Journal of General Systems*, 38(4), 2009.
10. Sébastien Ferré and Alice Hermann. Semantic search: Reconciling expressive querying and exploratory search. In *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 177–192. Springer Berlin Heidelberg, 2011.
11. Anna Formica. Concept similarity in Formal Concept Analysis: An information content approach. *Knowledge-Based Systems*, 21(1):80–87, February 2008.
12. Benrhard Ganter and Rudoph Wille. *Formal Concept Analysis*. Springer, Berlin, 1999.
13. Bjoern Koester. *FooCA: web information retrieval with formal concept analysis*. Beiträge zur begriffflichen Wissensverarbeitung. Verlag Allgemeine Wissenschaft, Mühltal, 2006.
14. Sergei O. Kuznetsov. Pattern Structures for Analyzing Complex Data. In *Proceedings of the 12th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, volume 5908 of *Lecture Notes in Computer Science*, pages 33–44. Springer Berlin Heidelberg, December 2009.
15. Sergei O Kuznetsov and Sergei A Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.
16. Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
17. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
18. Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.
19. Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smaïl-Tabbone. Using Domain Knowledge to Guide Lattice-based Complex Data Exploration. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 847–852, 2010.
20. Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smaïl-Tabbone. Querying a bioinformatic data sources registry with concept lattices. In *Proceedings of ICCS 2005*, pages 323–336. LNCS 3596, Springer, 2005.
21. George A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, November 1995.
22. A. Napoli, C. Laurenço, and R. Ducournau. An object-based representation system for organic synthesis planning. *International Journal of Human-Computer Studies*, 41(1/2):5–32, 1994.
23. Emmanuel Nauer and Yannick Toussaint. CreChainDo: an iterative and interactive Web information retrieval system based on lattices. *International Journal of General Systems*, 38(4):363–378, 2009.
24. Jonas Poelmans, Paul Elzinga, Stijn Viaene, and Guido Dedene. Formal concept analysis in knowledge discovery: a survey. In *Proceedings of the 18th international conference on Conceptual structures: from information to intelligence*, ICCS'10, pages 139–153, Berlin, Heidelberg, 2010. Springer-Verlag.
25. Jonas Poelmans, Dmitry I. Ignatov, Stijn Viaene, Guido Dedene, and Sergei O. Kuznetsov. Text mining scientific papers: a survey on FCA-Based information retrieval research. In *Proceedings of the 12th Industrial conference on Advances in Data Mining: applications and theoretical aspects*, volume 7377 of *Lecture Notes in Computer Science*, pages 273–287, Berlin, Heidelberg, July 2012. Springer Berlin Heidelberg.
26. Uta Priss. Lattice-based information retrieval. *Knowledge Organization*, 27:132–142, 2000.
27. Uta Priss. Formal concept analysis in information science. *Annual Review of Information Science and Technology.*, 40(1):521–543, December 2006.
28. Dan A. Simovici and Chabane Djeraba. *Mathematical Tools for Data Mining: Set Theory, Partial Orders, Combinatorics*. Springer Publishing Company, Incorporated, 1 edition, 2008.

29. Francisco J. Valverde-Albacete and Carmen Peláez-Moreno. Systems vs. methods: an analysis of the affordances of formal concept analysis for information retrieval. In *Proceedings of Formal Concept Analysis meets Information Retrieval (FCAIR), worlshop co-located with ECIR-2013*, Moscow, 2013.
30. Dean van der Merwe, Sergei Obiedkov, and Derrick Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In Peter Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 205–206. Springer, Berlin/Heidelberg, 2004.