Using Concept Lattices for Text Retrieval and Mining

Claudio Carpineto and Giovanni Romano

Fondazione Ugo Bordoni
Via Baldassarre Castiglione 59, 00142, Rome, Italy
carpinet,romano@fub.it

Abstract. The potentials of formal concept analysis (FCA) for information retrieval (IR) have been highlighted by a number of research studies since its inception. With the proliferation of small-size specialised text databases available in electronic format and the advent of Web-based graphical interfaces, FCA has then become even more appealing and practical for searching text collections. The main advantage of FCA for IR is the possibility of eliciting context, which may be used both to improve the retrieval of specific items from a text collection and to drive the mining of its contents. In this paper, we will focus on the unique features of FCA for building contextual IR applications as well as on its most critical aspects. The development of a FCA-based application for mining the web results returned by a major search engine is envisaged as the next big challenge for the field.

1 Introduction

The (short) history of the applications of FCA to information retrieval can be roughly split in three parts. In the 80's, some basic ideas were put forth - essentially that a concept can be seen as a query (the intent) with a set of retrieved documents (the extent) and that neighbour concepts can be seen as minimal query changes - and some preliminary study about the complexity of document lattices (i.e., the concept lattices built from collections of documents) was performed, mostly by Robert Godin and his co-workers ([28], [24]).

In the 90's, FCA has been integrated with basic IR techniques to build more comprehensive systems for information access. Concept lattices have been mainly used as a support structure for interactive subject finding tasks, with some explorations of the possibilities of FCA for text mining. We have seen several running prototypes, and some experimental evaluation comparing the performance of FCA-based IR with that of conventional IR methods (e.g., [27], [11], [10]).

Over the last few years, the range of functionalities has been expanded to include new tasks such as automatic text ranking and IR from semistructured data (e.g., [13], [16]); at the same time, new IR domains have been investigated including email messages, web documents, and file systems.

Although it might be argued that the impact made on mainstream IR has not been dramatic, the interest in using concept lattices for IR has grown both

within and outside of the FCA community. On the other hand, there is nowadays a much better awareness of the strengths and limitations of this technique for organizing and searching information. Furthermore, perhaps more importantly, we believe that the major changes faced by the information access world provide new unprecedented opportunities for FCA applications.

First of all, IR is no longer a boutique discipline for librarians, as the whole internet is being searched every day by billions of people around the world. Second, the availability of basic services for storing, networking, searching, and displaying information has led to a proliferation of specialized electronic databases for enterprises and individual users. Third, users may be interested in a variety of information searching tasks that go well beyond the capabilities of traditional IR systems dealing with the topic relevance task; examples include text data mining and several variants of the general topic relevance paradigm such as home page finding, question answering, IR from XML documents, news filtering, etc.

In the rest of the paper we will first discuss what makes FCA appealing to the development of more powerful search front ends. Next, we review how concept lattices have been used so far to improve specific traditional IR tasks or to handle new tasks that would be hardly dealt with by conventional systems. Then we will discuss the most difficult steps in the development of FCA-based IR applications, which may affect both the efficiency and the effectiveness of the overall system. Finally, we will argue for building a FCA-based system for visualizing web retrieval results on top of a major search engine.

2 Current search front ends and FCA

The growth of the web has favoured the emergence of new search applications, usage patterns, data formats, and interaction paradigms. To cope with the new requirements, more advanced search interfaces are being developed that provide the user with a range of access methods.

For instance, the Digital Library of the Association for Computing Machinery (http://www.acm.org) has been recently equipped with a new front end that not only allows the user to browse the documents contained in each category (e.g., journals, magazines, transactons, proceedings) and to search the full text (or title, or abstract) of publications, but also to use a number of browsable views into the literature, e.g., by Authors, by the ACM Computing Classification System (CCS), by Subjects, by Technical Interest.

Using the currently available tools, a number of interesting search tasks can be easily accomplished. For instance, a view by Author allows the user to drill down to a page that might be called an "author's virtual bibliographic home page", listing all the works by that author known to the system, or, to take another example, a view by CCS may be used to look into a specific domain, with the user drilling down from more general categories to narrower subjects and then to specific topics, thus progressively reducing the set of complying results.

One disadvantage of this system, as well as of most currently available front ends, is that the user must use a specific search functionality for each task, with little or no possibility of combining the results.

An even greater limitation is that while the present range of functionalities seem to support pretty well the user for the case when she is interested in finding those documents which are described by certain terms or categories, the same tools may be of little help if the user wants to disclose the content of specific sections of the Digital Library or to mine the concepts contained in a set of articles which have been filtered out by using some criterion.

This is an instance of the dichotomy between information retrieval and data view on one side and text mining on the other side. The latter involves the discovery of previously unknown information [31], as opposed to finding the best element among many other valid pieces of information, and can be seen as a form of exploratory data analysis [32].

The use of FCA can effectively complement the existing search systems to address some of their main limitations. Basically, FCA exploits the interdocument similarity between documents to offer an automatically-built support structure (i.e., the document lattice) in which to place the information searching process. The document lattice can be used to improve basic individual search strategies, as well as to host multiple integrated search strategies.

Having at her disposal a range of methods (e.g., querying, navigation, combination of data views, thesaurus climbing, pruning by search constraints), the user can select those that best fit the goal of the search, her knowledge of domain, her skills and preferences, and the results of the past interaction with the system. The user may then form hybrid strategies to make the search more accurate or fast.

The features discussed are naturally supported by concept lattices. Other approaches either require some manual coding, or do not allow for multiple retrieval strategies, or integrate multiple strategies in a loose manner.

3 Search functionalities enhanced by FCA

In this section we will examine which search functionalities or which combinations of search functionalities may be improved through a concept lattice. Most of the examined functionalities can be used both for text retrieval and text mining.

3.1 Query refinement

One of the most natural applications of concept lattices is query refinement, where the main objective is to recover from the null-output or the information overload problem.

This is not new, as some lattice representations were used in early IR [50] and even more recently [52] for refining queries containing Boolean operators. However, as these approaches typically rely on a Boolean lattice formalization of

the query, the number of proposed refinements may grow too large even for a very limited number of terms and they may easily become semantically meaningless to the user.

These limitations can be overcome by using concept lattices. One example is the REFINER system [12].

In response to a Boolean query, REFINER builds and displays a portion of the concept lattice of the documents being searched which is centered around a query concept. Such a query concept is found by computing the set of documents that satisfy the query and then by determining the set of terms possessed by all previously found documents. At this point, the most general concept containing these terms is chosen as the query concept; if there are no concepts that contain all the terms specified in the query (i.e., there are no documents exactly matching the query), REFINER adds a virtual concept to the lattice, as if the query represented a new document.

The potentials of this approach have been confirmed in an experiment on the CISI collection - a widely used, electronically-available bibliographical collection of 1460 information science documents described by a title and an abstract - showing that the effectiveness of information retrieval using REFINER was better than unrefined, conventional Boolean retrieval [12].

Concept lattices can be used also to refine queries expressed in natural language. The mapping of a query on to the lattice can be done by choosing the most general concept that contains all the query terms, similar to REFINER, or with some weaker criteria if such a concept coincides with the bottom of the lattice [54].

Once the query has been mapped on to a concept, the user may choose one of the neighbours of that concept, as in REFINER, or select one term from a list containing all the terms that are below that concept [36]. In the latter case, a substantial portion of the full concept lattice must be built.

3.2 Integrating querying and navigation

The effective integration of the query-based mode with the navigation paradigm has been the focus of much current research on information systems.

One typical choice is to maintain different retrieval methods in parallel (e.g., [39], [23]); in this case, the integrated system is, in practice, like a switch whereby the user may select either strategy. A tighter form of integration is achieved by cascading the two strategies, e.g., browsing prior to querying [42], or querying prior to browsing [38], or by having them coexist in the same search space ([1], [29]).

In these forms of integration the system may have to maintain several data structures possibly supporting different kinds of operations; when a single data structure is used consistency problems may arise.

Concept lattices take the hybrid searching paradigm one step further. As querying and navigation share the same data space and exchange their search results, they can be consistently integrated, without the need of mapping different representations on the part of the user. Furthermore, other search strategies

such as thesaurus climbing, space pruning, and partial views, can be easily combined in the same framework.

To characterize this state of affairs, in [9] the metaphor of the GOMS user's cognitive model [5] and user activity [40] is used. At any given time, the system is in a certain state, characterised by a current retrieval space (usually a subset of the original document lattice) and by a focus concept within it. In each state, the user may select an operator (browsing, querying, bounding, thesaurus climbing) and apply it. As a result, a transition is made to a new state, possibly characterised by a new retrieval space and/or new focus. The new state is evaluated by the user for retrieval, and then the whole cycle may be iterated.

Therefore each interaction sequence may be composed of several operators, connected in various orders. For instance, the user may initially bound the search space exploiting her knowledge about the goal, then query the system to locate a region of interest within the bounded space, then browse through the region; also, at any time during this process, the user may take advantage of the feedback obtained during the interaction to make a jump to a different but related region (e.g., by thesaurus climbing), or to further bound the retrieval space.

The merits and performance of using concept lattices for supporting hybrid search strategies have been described in a number of papers (e.g., [27], [10], [19],[16]). They can be summarized as greater flexibility, good retrieval effectiveness, and mining capabilities.

Among the various pieces of information that can be easily mined in a collection D using a concept lattice-based method, are the following: (i) Find the most common or uncommon subjects in D, (ii) Find which subjects imply, or are implied by, other subjects in D, (iii) Find novel and unpredictable subject associations in D, (iv) Find which subjects allow gradual refinement of subsets of D. Several detailed examples of mining information that would be difficult to acquire using the traditional information retrieval methods are provided in the cited papers.

3.3 Context-sensitive use of thesauri

In information retrieval applications, there often exist subsumption hierarchies on the set of terms describing the documents, in the form of a thesaurus.

A thesaurus can be integrated into a concept lattice either by explicitly expanding the original context with the implied terms or by taking into account the thesaurus ordering relation during the construction of the lattice ([8], [11]).

Using a thesaurus basically makes it possible to create new meaningful queries and guarantees that more general queries are indexed with more general terms, whereas in a standard concept lattice each query is strictly described by the terms present in the documents and possible semantic relationships between the terms themselves are ignored.

The user may thus locate the information of interest more effectively and quickly, partly because of enhanced navigation (the proximity of concepts in the lattice being related to semantic factors) and partly because of focused querying (as concept terms may be specialized/generalized using the thesaurus). An

experimental evaluation of the retrieval effectiveness of a thesaurus-enhanced concept lattice is described in [11].

As stated above, the common approach is to (explicitly or implicitly) add the implied terms to each document according to the thesaurus ordering relation. Uta Priss [44] discusses other possible ways in which a context and a thesaurus can be merged into an expanded context. She also suggests that the user should be given the possibility of interactively combinining concepts from multiple thesauri, or thesaurus facets, using Boolean operators [45].

Improving the representation of the document collection at hand is not the only possible reason to use a thesaurus. One might integrate a thesaurus in a lattice with the goal of analyzing the appropriateness of the thesaurus classification for a specific collection of documents.

The latter approach draws an interesting analogy with the applications of concept lattices in object-oriented modelling (e.g., [25], [49]), where type or class hierarchies are merged into a lattice of software programs with the goal of restructuring the existing hierarchies.

3.4 Combining multiple views of semi-structured data

When the data can be classified along multiple axes (e.g., functional, geographical, descriptive), it may be convenient for the user to bring in new attributes in an incremental fashion, making decisions based on the information displayed by the system for the current choice of the attributes.

Think of a topic such as *Italian restaurants with a "dehors" near the Louvre Museum*. If there is no such restaurant, the user may find it useful to look for best matching restaurants by examining first the attributes that have higher priority to her and then moving on to the attributes with lower priority, e.g., geographical proximity first, then type of cuisine, and lastly possession of an open-air space.

In the FCA setting, this general approach has been implemented by a nesting & zooming technique, whereby the user may combine the lattices corresponding to each partial view and focus on the points of interest. To visualize the combination of partial views, a particular lattice visualization scheme is used, called nested line diagram, which will be discussed in Section 4.3.

Using partial views is most suitable for many-valued contexts, because it may be easier to identify valuable subcontexts. Indeed, in many cases, the lattices of certain subcontexts may be seen as conceptual scales of the given context, in the sense of [21]. In principle, partial views can be applied also to one-valued contexts by vertically slicing the context table (an example is described in [47]), but in the latter case it may be more difficult to select subcontexts that bear value, or just meaning. In fact, some of the most interesting applications have been developed in domains characterized by semistructured data, such as those for searching collections of emails ([17], [16]) or for analysing real-estate data extracted from the web [15].

3.5 Bounding the search space with user constraints

Bounding is one of the functionalities implemented in the ULYSSES prototype ([9], [10]) to help the user focus the search on the relevant parts of a large concept lattice. Bounding allows users to prune the search space from which they are retrieving information during the search. The user may dynamically apply constraints with which the sought documents have to comply and the current search space is bounded accordingly. The constraints are expressed as inequality relations between the description of admissible concepts and a particular conjunction of terms, and the partitions induced over the search space by the application of such constraints present useful properties from the point of view of the information retrieval performance.

There are four possible constraints: $\uparrow c_1$, $\downarrow c_1$, $\neg \uparrow c_1$, $\neg \downarrow c_1$, where c_1 is the intent of some concept in the lattice. The constraint $\downarrow c_1$, for example, causes the system to prune away from the concept lattice all the concepts whose intent is either greater than or incomparable with c_1 (in other terms, all the concepts which are not below c_1).

To implement this framework, in [8] it is described an efficient algorithm based on two boundary sets - one containing the most specific elements of the admissible space (i.e., the lower boundary set) and the other containing the most general elements (i.e., the upper boundary set) - that can incrementally represent and update the constrained space. As more and more constraints are added, the admissible space shrinks, and the two boundary sets may eventually converge to the target class.

3.6 Overcoming the vocabulary problem in text ranking

Current best-matching information retrieval systems are limited by their inability of retrieving documents which contain the same concept as the query but are expressed with different words. A common solution to alleviate this vocabulary problem is to create a richer query context, mainly based on the first documents retrieved by the original query [6] or based on some form of terminological knowledge structure [18].

A more fundamental solution to word mismatch relies on the exploitation of inter-document similarity, following van Rijsbergen's cluster hypothesis that relevant documents tend to be more similar to each other than non-relevant documents. The best known approach is to rank a query not against individual documents but against a hierarchically grouped set of document clusters [58]. This approach, however, may involve the use of some heuristic decisions both to cluster the set of documents and to compute a similarity between individual document clusters and a query. As a result, hierarchical clustering-based ranking may easily fail to discriminate between documents that have manifestly different degrees of relevance for a certain query.

The limitations of hierarchical clustering-based ranking can be overcome by using the concept lattice of the document collection as the underlying clustering structure. The concept lattice may then be used to drive a transformation

between the representation of a query and the representation of each document. This approach is described in [13].

Essentially, the query is merged into the document lattice and each document is ranked according to the length of the shortest path linking the query to the document concept. Of course, this is a quasi ordered retrieval output, because the documents that are equally distant from the query concept have the same score. We can think of the sets containing equally-ranked documents as concentric rings around the query node, the longer the radius, the lower the document score (of the associated documents).

An evaluation performed on two test document collections of small size, i.e., CACM (3204 documents) and CISI (1460 documents), showed that concept lattice-based ranking was comparable to best-matching ranking and better than hierarchical clustering-based ranking on the whole document set, whereas it clearly outperformed the other two methods when the specific ability to rank documents that did not match the query was measured.

4 Issues for FCA-based IR applications

Most FCA-based IR applications involve the following three steps: (a) extraction of a set of index terms that describe each document of the given collection, (b) construction of the concept lattice of the document-term relation generated at step (a), (c) visualization of the concept lattice built at step (b).

The solution to each step may crucially affect the efficiency and/or the effectiveness of the overall application. In the next subsections we will analyze each step in turn.

4.1 Automatic generation of index units

This step is not necessary if each document is already equipped with a set of index terms. In most situations of interest, however, the index terms are not available and their manual generation is often impractical or even unfeasible (think of large text databases that change frequently over time).

Automatic indexing has long been studied in information retrieval. To automatically extract a set of index terms describing each document, the following steps can be followed.

1. Text segmentation. The individual words occurring in a text collection are extracted, ignoring punctuation and case.

2. Word stemming. Each word is reduced to word-stem form. This may be done by using some large morphological lexicon that contains the standard inflections for nouns, verbs, and adjectives (e.g., [34]), or via some rule-based stemmer such as Porter's [43].

3. Stop wording. A stop list is used to delete from the texts the (root) words that are insufficiently specific to represent content. The stop list included in the CACM dataset, for instance, contains 428 common function words, such as

"the", "of", "this", "on", etc. and some verbs, e.g., "have", "can", "indicate", etc.

4. Word weighting. This step is necessary to perform word selection, described in step 6; it may be also useful to discriminate between the documents that belong to a same concept, e.g., for automatic text ranking.

For each document and for each term, a measure of the usefulness of that term in that document is derived. The goal is to identify words that characterize the document to which they are assigned, while also discriminating it from the remainder of the collection. This has long been modeled by the well known $tf \cdot idf$ weighting scheme, which is now a bit outdated.

The two typical assumptions of the $tf \cdot idf$ scheme - namely that multiple appearences of a term in a document are more important than the single appearence (tf) and that rare terms are more important than frequent terms (idf) - have been extended through a third length normalization assumption stating that for the same quantity of term matching, long documents are less important than short documents.

These three assumptions have been implemented using several approaches, most notably using Robertson's probabilistic model [46], statistical language modeling [60], and deviation from randomness [2]. These recent models have been shown to perform much better than the classical $tf \cdot idf$ scheme on large, heterogeneous test collections, such as those used at TREC (Text REtrieval Conference, http://trec.nist.gov).

When the documents to be indexed are obtained in response to a query, it might be more effective to use term scoring functions that are based on the difference between the distribution of the terms in the set of retrieved documents and the distribution of the terms in the whole collection. In this way, the scores assigned to each term may more closely reflect the relevance of the term to the specific query at hand rather than the general importance of the term in the collection. Several term-scoring functions of this kind and possible ways of combining them to improve the quality of the generated terms are discussed in [14].

Also, for semi-structured or web documents, text-based indexing might be complemented with other techniques that take advantage of additional sources of knowledge, such as document fields, incoming or outgoing links, anchor texts, and url structure.

5. Word selection. This last step is not necessary for IR systems performing full-text indexing (in fact, it has not been included in the classic blueprint for automatic indexing suggested by Salton [48]), but it is very important for FCA-based systems to facilitate the subsequent process of lattice construction.

This problem is customarily addressed by using some heuristic threshold which restricts the index set. Among others, one can use as selection criterion the mean of weights in the document [13] or the value corresponding to one standard deviation above the mean [10]. A more elaborate approach is to choose the feature subset that maximizes the performance of a certain retrieval task or

minimizes some involved error, but this might be too difficult or expensive in many cases.

Clearly, reducing the set of features may affect the retrieval effectiveness, although this does not necessarily result in performance degradation. The effects of feature selection on FCA-based text ranking are discussed in [13].

4.2 Efficient lattice construction

It is well known that the size of a concept lattice may grow exponentially with the number of objects. However, this situation occurs rarely in the information retrieval domain, as witnessed by a number of theoretical and experimental findings.

To gain some deeper insights into the actual order of magnitude of document lattices, one can hypothesize that the document description obeys some simple distribution of probability (estimated by term frequency). If each index term is assigned to each document with constant, independent probability $p = k/|M|$, the number of keywords per object follows a binomial distribution with a mean value of k and the mean number of concepts in the lattice, derived by Godin *etal.* [28], is given by:

$$|C| = \sum_{i=0}^{|G|} \sum_{j=0}^{|M|} \binom{|G|}{i} \binom{|M|}{j} p^{ij} (1-p^i)^{|M|-j} (1-p^j)^{|G|-i} \qquad (1)$$

Here we plot Equation 1 for four values of k (5, 10, 20, and 50), choosing $|G| = 10000$. Figure 1 clearly shows that the number of concepts varies from linear to quadratic with respect to the number of documents, at least for the chosen parameter values. The size of the lattice grows as the number of terms per document increases; the upper bound is reached for $k = 50$, corresponding to a probability of assigning a term to a document equal to $50/1000 = 0.05$. This latter value accounts for a relatively dense context table, at least for information retrieval applications.

These findings agree with experimental observations. For instance, for the test collection CACM (3204 documents), it has been reported that the concept lattice contained some 40,000 concepts [13] , whereas for the test collection CISI (1460 documents), characterized by a larger number of terms per document (about 40), the size of the lattice grew to 250,000 ([10], [12], [13]).

Several algorithms have been developed for building the concept lattice of an input context (G, M, I) (e.g., [20], [4], [7], [26]). Usually, the efficiency of such algorithms critically depend on the number of concepts present in the lattice.

The best theoretical worst time complexity is $O(|C||M|(|G|+|M|))$, exhibited by the algorithm presented by Nourine and Raynaud [41]; in practice, the behaviour may significantly vary depending on a number of factors including the relative sizes of G and M, the size of I, and the density of the context, i.e., the size of I relative to the product $|G||M|$ (see [35] for an experimental comparison).

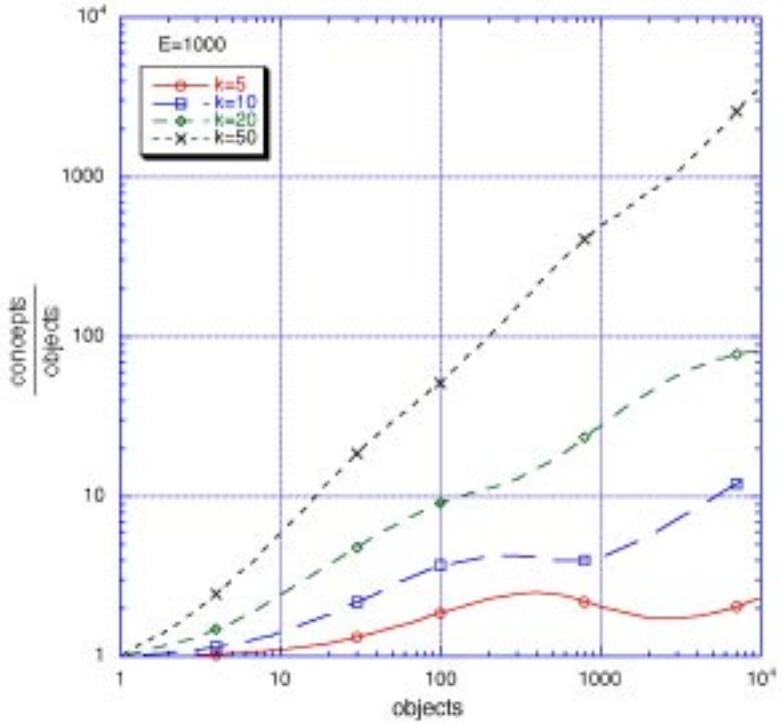

Fig. 1. Theoretical space complexity of document lattices. Both scales are logarithmic.

As the size of the document lattice may largely exceed the number of documents and because of the inherent complexity of the lattice-building algorithms, the full document lattice may be constructed only for small to medium size collections, usually up to thousands of documents. For larger test collection, such as those containing millions of documents used at TREC, it is just unfeasible to build the complete associated concept lattice.

Fortunately, in many applications it is enough to compute a very small portion of the lattice, typically consisting of a focus concept and its neighbours. Such a focus concept, for instance, might be selected by the user through a point-and-click graphical user interface showing a partial lattice, or, as seen earlier, it might be computed by mapping a natural language or Boolean query on the document lattice. In this case, the system returns just the neighbours of a focus concept in the lattice.

The problem of generating all the nearest neighbours of a given concept has been addressed both to build a full lattice ([4], [37]) and to find just the portion of lattice centered around that concept [12]. As this is a very general and useful algorithm, we describe it here in a detailed manner.

Our version follows the same general strategy as the works cited above but differs in two main details, namely the generation of the candidate extent and the choice of the admissible candidates. To solve the latter subtasks, we borrow the more efficient procedures presented in [41].

Figure 2 describes the algorithm for determining the set of lower neighbours of a given concept; the determination of the upper neighbours is a dual problem and can be solved by easily adapting the given algorithm. The theoretical time complexity of the computation of the lower neighbours is $O(|G||M|^2)$; the time complexity of the algorithm for finding both the lower and upper neighbours is $O(|G||M|(|G|+|M|))$. The use of Nourine and Raynaud's procedures does not affect the theoretical complexity of the algorithm but they may produce a substantial efficiency gain in practical situations.

Although the possession of a fast algorithm for computing the underlying concept lattice or part of it may be an essential prerequisite for IR applications as well as for applications concerning rule mining or software analysis, the issue of an optimal selection of the available algorithms has not been adequately addressed. More research is needed on the evaluation of competing algorithms, both from a theoretical and an experimental point of view. We will return to this in the conclusion.

4.3 Effective lattice visualization

Except for automatic tasks such as document ranking, most of the IR applications based on concept lattices require some form of exploration of the graph diagram on the part of the user. However, forming useful visualizations of graph structures is notoriously difficult due to the conflicting issues of size, layout, and legibility on limited screen area. The problem is further compounded by the fact that the concept lattices of real applications are usually very large. The common

Find Lower Neighbours
Input: Context (G, M, I), concept (X, Y) of context (G, M, I)
Output: The set of lower neighbours of (X, Y) in the concept lattice of (G, M, I)

1. $lowerNeighbours := \emptyset$
2. $lNCandidates := \emptyset$
3. **for** each $m \in M \backslash Y$
4. $\quad X_1 := X \cap \{m\}'$
5. $\quad Y_1 := X_1'$
6. $\quad$ **if** $(X_1, Y_1) \notin lNCandidates$
   $\quad\quad$ **then**
7. $\quad\quad$ Add (X_1, Y_1) to $lNCandidates$
8. $\quad\quad$ $count(X_1, Y_1) := 1$
   $\quad\quad$ **else**
9. $\quad\quad$ $count(X_1, Y_1) := count(X_1, Y_1) + 1$
10. $\quad$ **if** $(|Y_1| - |Y|) = count(X_1, Y_1)$ **then**
11. $\quad\quad$ Add (X_1, Y_1) to $lowerNeighbours$

Fig. 2. Find Lower Neighbours algorithm

approach is to show or hide parts of the lattice via interactive specification of a focus concept and/or subsets of terms.

One simple method consists of showing just the neighbours of a focus concept. Simple graphical interfaces of this kind have been suggested or implemented in several works, including [24], [27], [11], and the REFINER system discussed earlier [12].

To show a larger portion centered around a focus concept, we can resort to focus+context visualization tehniques. Focus+context viewers use as a general metaphor the effects observed when looking through fisheye lenses or magnifying glasses. A simple way to implement a fish eye view is to display the information contained in a lattice in varying levels of details depending on the distance from the focus; the size of the information at the focal point are increased whereas the information placed further away are reduced in scale.

In practice, a specific display format for each subset of concepts placed at the same distance from the focus concept can be used, the distance being the length of the shortest paths between the concepts. Such displays may involve different combinations of sizes, fonts, and types of information. A similar approach has been adopted in the ULYSSES prototype ([9], [10]).

In some cases, we are mainly interested in the portion of the lattice placed *below* a focus concept. A simple and useful approach is to use a tree, by making the focus concept the root and associating each sequence of concepts below the focus with a path. The tree representation has several advantages. As the metaphor of hierarchical folders is used for storing and retrieving files, bookmarks, menus items, etc., most users are familiar with it and hence no training

on the part of the user is required. Furthemore, it takes little space on the screen and it may be drawn efficiently.

The main disadvantage is that there may be a considerable amount of duplication of information when the concepts have multiple parents. On the other hand, this is not very likely to happen if only some levels of the hierarchies are visualized. The tree-like representation surfaces in some more recent prototypes based on concept lattices such as HierMail [16] and its commercial follow-up Mail-Sleuth (*http://www.mail-sleuth.com*).

An alternative approach to lattice visualization is based on combining multiple partial views of the data represented in the context. A particular scheme termed nested line diagram has been developed within the FCA community and first implemented in the Toscana system ([57], [55],[56], [53]). In essence, (i) two or more subsets of attributes are chosen by the user, (ii) the concept lattices of the subcontexts identified by the attribute subsets of step 1 are found, and (iii) the full concept lattice is embedded in the direct product of the lattices of subcontexts as a join-semilattice. The overall effect is that of having several complete lattices of partial contexts nested into one another rather than a partial lattice of a complete context.

One advantage of nested line diagrams is that the size of each local diagram cannot exceed the number of possible combinations of the attribute values present in the corresponding subcontext, regardless of the number of objects in the database. Hence, it is possible to draw the full lattices of each subcontext even for large databases, provided that the subcontexts are sufficiently small. Clearly, this approach is effective when the number of scales to combine is limited.

Before concluding this section we would like to emphasize that the fast advances in the field of graphical web interfaces may spur a renewed interest in the techniques for lattice visualization. In addition to exploring the use of alternative visual layouts proposed in the information visualization field [22], whether focused on more complex inherent graph substructures or on richer interactive or linking mechanisms, it would be useful to compare relative merits and drawbacks of each visualization scheme for specific performance tasks.

5 The next challenge

Current Web retrieval interfaces are limited by a lack of a concise representation of the content of all retrieved documents; conventional textual displays take much perusal time and screen space and do not enable inspection of more documents at a time. A related drawback is represented by the inability of providing good refinement terms for narrowing down the large set of results that are typically returned in response to a query.

Current research is addressing these shortcomings by attempting to provide visual or terminological cues for interpreting and manipulating retrieval results (e.g., [30], [3], [59], [33]), but most proposed approaches are still hampered by theoretical and practical limitations.

Concept lattices are a good candidate for extending user control over presentation and selection of web retrieval results. Among the anticipated advantages are the following. As the refinement terms in the lattice are based on all the words in the query concept (rather than on single words) and they are driven by the content of the documents being searched (rather than on predefined term-term associations), the suggested terms may work well not only for simple, popular topics - as with the "related keyword" feature provided by some commercial search engines - but also for specialized or ambiguous topics. Furthermore, concept lattices are more flexible than hierarchical clustering approaches, because it is easier for the user to recover from bad early decisions while traversing the structure. Finally, several search strategies can be integrated in the same framework.

Searching the web results using FCA is technically feasible. Now we sketch a possible architecture. The system takes as input a user query. The query is forwarded to a selected search engine, and the first pages retrieved by the search engine in response to the query are collected and parsed. At this point, a set of index units that describe each returned document is generated; such indices are next used to build the concept lattice corresponding to the retrieved results. The last steps consists of showing the lattice to the user and managing the subsequent interaction between the user and the system.

There are a number of design decisions involved here, which may affect both the efficiency and the effectiveness of the whole system. Following the data flow, the main decisions are: analyzing a small/large number of retrieved documents, using document snippets or full text documents, performing single- or multi-keyword indexing, constructing partial or full concept lattice, using simple or sophisticated visualization schemes, allowing single or multiple interaction modes. In order to ensure fast response times and good overall retrieval effectiveness, each component should be carefully designed and engineered; also, the interactions with the other components should be studied.

In spite of such difficulties, building a system of this kind can help to address the issue of retrieving and mining web documents in a more principled and effective manner. We believe that the challenge is quite realistic, given the current state of the art of FCA and IR techniques, and that this might be a big success for the whole field of concept data analysis.

6 Conclusions

The advances in the methods for constructing and searching document lattices coupled with the unprecedented need for contextual text processing techniques make FCA a strong tool for building modern IR applications. In order to grasp this opportunity, some bigger effort and a few cautions are required on the part of FCA developers. Here we would like to make some recommendations that can help build successful applications.

Focus on appropriate IR tasks. The chosen tasks must be suitable for FCA and should not be easily solved by conventional IR techniques. For instance,

natural language processing techniques could hardly demonstrate their usefulness as long as they were employed to improve the classical topic relevance task, whereas they have recently become an essential component of systems performing question answering on large text collections.

Integration with advanced IR techniques. To solve any nontrivial task, it may be necessary to integrate FCA methods with existing IR techniques. As the IR field is moving on fastly, it is important to pick up the most updated techniques. For instance, using the classical $tf \cdot idf$ weighting scheme rather than the much more effective methods that have been developed lately may seriously degrade the performance of the whole IR application.

Adoption of IR evaluation metrics. The effectiveness of the application should be measured using recognized evaluation metrics. This holds both for automatic and interactive tasks. Evaluation studies of the latter type of tasks, which is more relevant to FCA applications, are not frequent in the literature probably due to a combination of methodological, technological, organizational, and economical issues, although there are some significant exceptions (e.g., [51], [3]).

Engineering test collections. It would be very useful to have a set of test databases on which to run rigorous experimental comparisons. Test collections could be used to evaluate the efficiency (and perhaps the correctness) of the algorithms for constructing the document lattice and also to perform more controlled IR experiments. Engineering test collections may be an important step to take for the whole research community on FCA to encourage systems implementations and to measure advances.

Deployment of tools. Although a number of FCA papers have been published in major IR forums, the awareness of the utility of concept lattices for IR is still limited outside of the FCA community. The free availability of an on-line, concept lattice-based tool for mining Web retrieval results would probably greatly increase the scope of FCA for IR.

7 Acknowledgments

We would like to thank Gerd Stumme for his helpful comments on an earlier version of this paper.

References

1. M. Agosti, M. Melucci, and F. Crestani. Automatic authoring and construction of hypertexts for information retrieval. *ACM Multimedia Systems*, 3:15–24, 1995.
2. G. Amati, C. Carpineto, and G. Romano. FUB at TREC-10 Web Track: A Probabilistic Framework for Topic Relevance Term Weighting. In *Proceedings of the 10th Text REtrieval Conference (TREC-10), NIST Special Publication 500-250*, pages 182–191, Gaithersburg, MD, USA, 2001.
3. E. Berenci, C. Carpineto, V. Giannini, and S. Mizzaro. Effectiveness of keyword-based display and selection of retrieval results for interactive searches. *International Journal on Digital Libraries*, 3(3):249–260, 2000.

4. J.P. Bordat. Calcul pratique du treillis de Galois d'une corrispondance. *Math. Sci. Hum.*, 96:31–47, 1986.
5. S. Card, T. Moran, and A. Newell. *The psychology of human-computer interaction.* Lawrence Erlbaum Associates, London, 1983.
6. C. Carpineto, R. De Mori, G. Romano, and B. Bigi. An information theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19(1):1–27, 2001.
7. C. Carpineto and G. Romano. An order-theoretic approach to conceptual clustering. In *Proceedings of the 10th International Conference on Machine Learning*, pages 33–40, Amherst, MA, USA, 1993.
8. C. Carpineto and G. Romano. Dynamically bounding browsable retrieval spaces: an application to Galois lattices. In *Proceedings of RIAO 94: Intelligent Multimedia Information Retrieval Systems and Management*, pages 520–533, New York, New York USA, 1994.
9. C. Carpineto and G. Romano. ULYSSES: A lattice-based multiple interaction strategy retrieval interface. In Unger Blumenthal, Gornostaev, editor, *Human-Computer Interaction, 5th International Conference, EWHCI, Selected Papers*, pages 91–104. Springer, Berlin, 1995.
10. C. Carpineto and G. Romano. Information retrieval through hybrid navigation of lattice representations. *International Journal of Human-Computer Studies*, 45(5):553–578, 1996.
11. C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):1–28, 1996.
12. C. Carpineto and G. Romano. Effective reformulation of Boolean queries with concept lattices. In *Proceedings of the 3rd International Conference on Flexible Query-Answering Systems*, pages 83–94, Roskilde, Denmark, 1998.
13. C. Carpineto and G. Romano. Order-Theoretical Ranking. *Journal of the American Society for Information Science*, 51(7):587–601, 2000.
14. C. Carpineto, G. Romano, and V. Giannini. Improving retrieval feedback with multiple term-ranking function combination. *ACM Transactions on Information Systems*, 20(3):259–290, 2002.
15. R. Cole and P. Eklund. Browsing semi-structured web texts using formal concept analysis. In *Proceedings of the 9th International Conference on Conceptual Structures*, pages 319–332, Stanford, CA, USA, 2001.
16. R. Cole, P. Eklund, and G. Stumme. Document retrieval for e-mail search and discovery using formal concept analysis. *Applied Artificial Intelligence*, 17(3):257–280, 2003.
17. R. Cole and G. Stumme. CEM: A Conceptual Email Manager. In *Proceedings of the 8th International Conference on Conceptual Structures*, pages 438–452, Darmstadt, Germany, 2000.
18. E. Efthimiadis. Query expansion. In M. E. Williams, editor, *Annual Review of Information Systems and Technology, v31*, pages 121–187. American Society for Information Science, Silver Spring, Maryland, USA, 1996.
19. S. Ferré and O. Ridoux. A file system based on concept analysis. In *Proceedings of the 1st International Conference on Computational Logic*, pages 1033–1047, London, UK, 2000.
20. B. Ganter. Two basic algorithms in concept analysis. Technical Report FB4–Preprint No. 831, TU Darmstadt, Germany, 1984.
21. B. Ganter and R. Wille. *Formal Concept Analysis - Mathematical Foundations.* Springer, 1999.

22. N. Gershon, S. K. Card, and S. G. Eick. Information visualization tutorial. In *Proceedings of ACM CHI'98: Human Factors in Computing Systems*, pages 109–110, Los Angeles, CA, USA, 1998.
23. D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. Jr O'Toole. Semantic file systems. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pages 16–25, 1991.
24. R. Godin, J. Gecsei, and C. Pichet. Design of a browsing interfaces for information retrieval. In *Proceedings of the 12th Annual International ACM SIGIR Conference on Reasearch and Development in Information Retrieval*, pages 32–39, 1989.
25. R. Godin and H. Mili. Building and Maintaining Analysis Level Class Hierarchies Using Galois Lattices. In *Proceedings of the 8th Annual Conference on Object Oriented Programming Systems Languages and Applications*, pages 394–410, Washington, D.C., USA, 1993.
26. R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on Galois lattices. *Computational Intelligence*, 11(2):246–267, 1995.
27. R. Godin, R. Missaoui, and A. April. Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. *International Journal of Man-Machine Studies*, 38:747–767, 1993.
28. R. Godin, E. Saunders, and J. Jecsei. Lattice model of browsable data spaces. *Journal of Information Sciences*, 40:89–116, 1986.
29. B. Gopal and U. Manber. Integrating content-based access mechanisms with hierarchical file systems. In *Proceedings of 3rd Symposium on Operating Systems Design and Implementation*, pages 265–278, New Orleans, Louisiana, USA, 1999.
30. M. Hearst. User interfaces and visualization. In R. Baeza-Yates and B. Ribeiro-Neto, editors, *Modern Information Retrieval*, pages 257–322. ACM Press, New York, New York, USA, 1999.
31. M. A. Hearst. Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, MD, USA, 1999.
32. D. C. Hoaglin, F. Mosteller, and J. W. Tukey. *Understanding robust and exploratory data analysis*. John Wiley & Sons, Inc., 1983.
33. H. Joho, M. Sanderson, and M. Beaulieu. Hierarchical approach to term suggestion device. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 454, Tampere, Finland, 2002.
34. D. Karp, Y. Schabes, M. Zaidel, and D. Egedi. A freely available wide coverage morphological analyzer for English. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, pages 950–955, Nantes, France, 1992.
35. S.O. Kuznetsov and S.A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2–3):189–216, 2002.
36. C. Lindig. Concept-based component retrieval. In *Working notes of the IJCAI-95 workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs*, pages 21–25, Montreal, Canada, 1995.
37. C. Lindig. Fast concept analysis. In *Working with conceptual structures - Contribution to the 8th International Conference on Conceptual Structures*, pages 152–161, Darmstadt, Germany, 2000.
38. D. Lucarella, S. Parisotto, and A. Zanzi. MORE: Multimedia Object Retrieval Environment. In *Proceedings of ACM Hypertext'93*, pages 39–50, Seattle, WA, USA, 1993.

39. Y. Maarek, D. Berry, and G. Kaiser. An information retrieval approach for automatically constructing software libraries. *IEEE Transactions on software Engineering*, 17(8):800–813, 1991.
40. D. Norman. Cognitive engineering. In D. Norman and S. Draper, editors, *User centered system design*, pages 31–61. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.
41. L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71:199–204, 1999.
42. G. Pedersen. A browser for bibliographic information retrieval based on an application of lattice theory. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 270–279, Pittsburgh, PA, USA, 1993.
43. M. F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.
44. U. Priss. A graphical interface for document retrieval based on Formal Concept Analysis. In *Proceedings of the 8th Midwest Artificial Intelligence and Cognitive Science Conference*, pages 66–70, Dayton, Ohio, USA, 1997.
45. U. Priss. Lattice-based information retrieval. *Knowledge Organization*, 27(3):132–142, 2000.
46. S. E. Robertson, S. Walker, and M. M. Beaulieu. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC, and Interactive track. In *Proceedings of the 7th Text REtrieval Conference (TREC-7), NIST Special Publication 500-242*, pages 253–264, Gaithersburg, MD, USA, 1998.
47. T. Rock and R. Wille. Ein toscana-erkundungssystem zur literatursuche. In G. Stumme and R. Wille, editors, *Begriffliche Wissensver-arbeitung. Methoden und Anwendungen*, pages 239–253. Springer, Berlin, Germany, 2000.
48. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison Wesley, 1989.
49. G. Snelting and F. Tip. Reengineering class hierarchies using concept analysis. In *Proceedings of ACM SIGSOFT 6th International Symposium on Foundations of Software Engineering*, pages 99–110, Lake Buena Vista, FL, USA, 1998.
50. D. Soergel. Mathematical analysis of documentation systems. *Information storage and retrieval*, 3:129–173, 1967.
51. A. Spink and T. Saracevic. Interaction in information retrieval: selection and effectiveness of search terms. *Journal of the American Society for Information Science*, 48(8):741–761, 1997.
52. A. Spoerri. InfoCrystal: Integrating exact and partial matching approaches through visualization. In *Proceedings of RIAO 94: Intelligent Multimedia Information Retrieval Systems and Management*, pages 687–696, New York, New York USA, 1994.
53. G. Stumme. Local scaling in conceptual data systems. In *Proceedings of the 6th International Conference on Conceptual Structures*, pages 308–320, Montpellier, France, 1998.
54. F. J. van der Merwe and D. G. Kourie. Compressed pseudo-lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2–3):229–254, 2002.
55. F. Vogt, C. Wachter, and R. Wille. Data analysis based on a conceptual file. In H.-H. Bock, W. Lenski, and P. Ihm, editors, *Classification, Data Analysis and Knowledge Organization*, pages 131–140. Springer, Berlin, 1991.
56. F. Vogt and R. Wille. TOSCANA - A graphical tool for analyzing and exploring data. In R. Tammassia and I. G. Tollis, editors, *Graph Drawing'94*, pages 226–233. Springer, Berlin, 1995.
57. R. Wille. Line diagrams of hierarchical concept systems. *Int. Classif.*, 11(2):77–86, 1984.

58. P. Willet. Recent trends in hierarchic document clustering: a critical review. *Information Processing & Management*, 24(5):577–597, 1988.
59. O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *WWW8/Computer Networks*, 31(11–16):1361–1374, 1999.
60. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, New Orleans, LA, USA, 2001.