Journal of Universal Computer Science, vol. 10, no. 8 (2004), 985-1013
submitted: 22/3/04, accepted: 28/6/04, appeared: 28/8/04

Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO

Claudio Carpineto
Fondazione Ugo Bordoni, Italy
carpinet@fub.it

Giovanni Romano
Fondazione Ugo Bordoni, Italy
romano@fub.it

Abstract: The recent advances in Formal Concept Analysis (FCA) together with the major changes faced by modern Information Retrieval (IR) provide new unprecedented challenges and opportunities for FCA-based IR applications. The main advantage of FCA for IR is the possibility of creating a conceptual representation of a given document collection in the form of a document lattice, which may be used both to improve the retrieval of specific items and to drive the mining of the collection's contents. In this paper, we will examine the best features of FCA for solving IR tasks that could not be easily addressed by conventional systems, as well as the most critical aspects for building FCA-based IR applications. These observations have led to the development of CREDO, a system that allows the user to query Web documents and see retrieval results organized in a browsable concept lattice. This is the second major focus of the paper. We will show that CREDO is especially useful for quickly locating the documents corresponding to the meaning of interest among those retrieved in response to an ambiguous query, or for mining the contents of the documents that reference a given entity. An on-line version of the system is available for testing at *http://credo.fub.it*.
Key Words: concept lattices, information retrieval, CREDO, web mining
Category: H.3.3,H.5.4

1 Introduction

The potentials of FCA for IR have been highlighted by a number of research studies since its inception, since the document-term relation used in IR can naturally be seen as a formal .context of FCA. 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 ([Godin *et al.*, 1986], [Godin *et al.*, 1989]).

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., [Godin *et al.*, 1993], [Carpineto and Romano, 1996b], [Carpineto and Romano, 1996a]).

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., [Carpineto and Romano, 2000], [Cole *et al.*, 2003]); at the same time, new IR domains have been investigated including email messages, web documents, and file systems.

As a result, 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 considerably grown. Nowadays, with the recent advances in the theory [Ganter and Wille, 1999] and practice [Carpineto and Romano, 2004] of concept lattices and the advent of Web-based graphical interfaces, FCA has become even more appealing and practical for searching text collections.

Not only is FCA suitable for IR, the converse is also true. Modern IR is in need of more powerful tools for eliciting context and concepts from the raw data, as the proliferation of electronic text databases coupled with the widespread availability of basic tools for storing, networking, searching, and displaying information has led to a request for specialized search services. Both enterprises and individual users are increasingly interested in a variety of information searching tasks that go well beyond the capabilities of traditional IR systems dealing with the topic relevance task, such as text data mining, question answering, and IR from XML documents.

In this paper, we will argue that FCA has become a natural candidate for addressing some of the challenges posed by modern IR, most notably for accessing and mining Web retrieval results. The presentation is split in two main parts.

In the first, following a review of 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, we examine 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.

The second part is dedicated to CREDO, a concept lattice-based system for accessing and mining Web retrieval results. We first discuss the lack of effectiveness of current search engine interfaces and some solutions for overcoming this problem. Then we describe the design and implementation of CREDO, analysing its main blocks in detail. The operation and utility of the system is also illustrated using some example sessions. Finally, we provide some conclusions.

2 IR Applications of Concept Lattices

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.

2.1 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 [Soergel, 1967] and even more recently [Spoerri, 1994] 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. The two fundamental observations are stated below.

Proposition 1. A concept can be seen as a query (the intent) with a set of retrieved documents (the extent).

Proposition 2. Following edges departing upward (downward) from a query produces all minimal conjunctive refinements (enlargements) of the query with respect to the collection of documents from which the concept lattice has been built.

These properties have inspired the development of several query refinement systems based on concept lattices, of which REFINER [Carpineto and Romano, 1998] is a well known representative.. 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 [Carpineto and Romano, 1998].

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 [van der Merwe and Kourie, 2002].

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 [Lindig, 1995]. In the latter case, a substantial portion of the full concept lattice must be built.

2.2 Integrating Querying and Navigation

One major limitation of most IR systems is that while they support pretty well the user for the case when she is interested in finding those documents which are about certain subjects, the same tools may be of little help for the reciprocal task, i.e., finding which subjects are covered by certain documents. As the latter task is addressed by providing some form of navigation through the content of the database, 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., [Maarek *et al.*, 1991], [Gifford *et al.*, 1991]); 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 [Pedersen, 1993], or querying prior to browsing [Lucarella *et al.*, 1993], or by having them coexist in the same search space ([Agosti *et al.*, 1995], [Gopal and Manber, 1999]).

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 [Carpineto and Romano, 1995] the metaphor of the GOMS user's cognitive model [Card *et al.*, 1983] and user activity [Norman, 1986] 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., [Godin *et al.*, 1993], [Carpineto and Romano, 1996a], [Ferré and Ridoux, 2000],[Cole *et al.*, 2003]). 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.

2.3 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 ([Carpineto and Romano, 1994], [Carpineto and Romano, 1996b]).

The latter approach is based on an explicit characterization of the relationship between a thesaurus and the corresponding thesaurus-enhanced concept lattice.

Proposition 3. Let (G, M, I) be a context and $(M^*, \leq_{M^*})$ be an ordered set, with $M^* \supseteq M$. Assuming that the following *compatibility condition* holds:

$$\forall g \in G, m_1 \in M, m_2 \in M^* : [(g, m_1) \in I, m_1 \leq_{M^*} m_2] \Rightarrow (g, m_2) \in I,$$

the ordering relation $\leq$ between the concepts in the thesaurus-enhanced concept lattice satisfies the following property:

$$(X_1, Y_1) \leq (X_2, Y_2) \;\Leftrightarrow\; \forall\, m_2 \in Y_2,\; \exists\, m_1 \in Y_1,\; m_1 \leq_{M^*} m_2$$

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 [Carpineto and Romano, 1996b].

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 [Priss, 1997] 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 [Priss, 2000].

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., [Godin and Mili, 1993], [Snelting and Tip, 1998]), where type or class hierarchies are merged into a lattice of software programs with the goal of restructuring the existing hierarchies.

2.4 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 3.3.

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 [Ganter and Wille, 1999]. In principle, partial views can be applied also to one-valued contexts by vertically slicing the context table (an example is described in [Rock and Wille, 2000]), 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 ([Cole and Stumme, 2000], [Cole *et al.*, 2003]) or for analysing real-estate data extracted from the web [Cole and Eklund, 2001].

2.5 Bounding the Search Space with User Constraints

Bounding is one of the functionalities implemented in the ULYSSES prototype ([Carpineto and Romano, 1995], [Carpineto and Romano, 1996a]) 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 [Carpineto and Romano, 1994] 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.

2.6 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 [Carpineto *et al.*, 2001] or based on some form of terminological knowledge structure [Efthimiadis, 1996].

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. In *latent semantic indexing* [Deerwester *et al.*, 1990], for instance, the relationships between documents are exploited to transform the representation of the documents themselves before ranking.

The inter-document similarity can also be directly used to determine the ranking of documents, without modifying their representation. In this case, a query is ranked not against individual documents but against a hierarchically grouped set of document clusters [Willet, 1988]. This simple and well known 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 [Carpineto and Romano, 2000]. 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.

The operation of concept lattice-based ranking can be characterized more precisely through the following proposition.

Proposition 4. Let $(\mathcal{C}(G, M, I);\, \succ\prec)$ be the ordered set formed by the set of concepts of the context (G, M, I) along with the nearest neighbour relation $(\succ\prec)$, i.e., for $x, y \in \mathcal{C}(G, M, I)$, $x \succ\prec y$ if $x \succ y$ or $y \succ x$, and define the *distance* between concepts x and y as the least $n \in \mathcal{N}$ for which the following condition holds:

$$\exists z_0,\, z_1,\, \ldots,\, z_n \;\in\; \mathcal{C}(D, T, I);\, \succ\prec) \text{ such that } x = z_0 \succ\prec z_1 \ldots \succ\prec z_n \;=\; y.$$

It follows that for each query q and $g_1, g_2 \in G$, g_1 is ranked ahead of g_2 with respect to q if and only if the distance between the query concept (q'', q') and the object concept (g_1'', g_1') is less than the distance between (q'', q') and the object concept (g_2'', g_2').

Of course, the output of concept lattice-based ranking is a quasi ordered set, 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.

3 Issues for Concept Lattice-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.

3.1 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., [Karp *et al.*, 1992]), or via some rule-based stemmer such as Porter's [Porter, 1980].

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 5; it may be also useful to discriminate between the documents that belong to a same concept, e.g., for automatic text ranking.

The typical approach to word weighting is as follows. 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 modelled by the well known "term frequency inverse document frequency" scheme, or $tf{\cdot}idf$. Term frequency is given by the ratio of the number of times a term occurs in a document to the total number of terms in that document. Inverse document frequency is the total number of documents in a collection over the number of documents in which the term occurs.

The two 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) - are usually extended through a third length normalization assumption, which states that for the same quantity of term matching, long documents are less important than short documents.

This three-component framework has been implemented using several approaches, the best-known of which is probably the vector space model [Salton and McGill, 1983]. More recent weighting functions are BM25 [Robertson *et al.*, 1998], statistical language modeling (SLM) [Zhai and Lafferty, 2001], and deviation from randomness (DFR) [Amati and van Rijsbergen, 2002]. The latter approaches have been shown to perform very well on large, heterogeneous test collections, such as those used at TREC and CLEF. In particular, BM25 has been used by most participants in TREC and CLEF in recent years.

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 [Carpineto *et al.*, 2002].

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 but is very important for FCA-based systems to facilitate the subsequent process of lattice construction.

This problem can be 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 [Carpineto and Romano, 2000] or the value corresponding to one standard deviation above the mean [Carpineto and Romano, 1996a]. 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, the selected index terms ultimately control the retrieval performance of the overall system. Reducing the set of features usually has a negative effect but this is not necessarily the case. For a discussion of the effects of feature selection on FCA-based text ranking see [Carpineto and Romano, 2000].

Note that the extraction of a good set of index terms is one of the most difficult steps when producing a useful document lattice, because an optimal strategy should consider the characteristics of the collection and the queries.

3.2 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 findings which suggest that the number of concepts varies from linear to quadratic with respect to the number of documents (e.g., [Godin *et al.*, 1986], [Carpineto and Romano, 1996b]).

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 [Carpineto and Romano, 2000] , 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 ([Carpineto and Romano, 1996a], [Carpineto and Romano, 1998], [Carpineto and Romano, 2000]).

Several algorithms have been developed for building the concept lattice of an input context (G, M, I) (e.g., [Ganter, 1984], [Bordat, 1986], [Carpineto and Romano, 1993], [Godin *et al.*, 1995]). Usually, the efficiency of such algorithms critically depend on the number of concepts C 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 [Nourine and Raynaud, 1999]; 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 [Carpineto and Romano, 2004] for a comprehensive presentation and analysis of the algorithms for lattice construction and [Kuznetsov and Obiedkov, 2002] for an experimental comparison).

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 ([Bordat, 1986], [Lindig, 2000]) and to find just the portion of lattice centered around that concept [Carpineto and Romano, 1998]. 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 [Nourine and Raynaud, 1999].

Figure 1 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,

FindLowerNeighbours
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)

$lowerNeighbours := \emptyset$
$lNCandidates := \emptyset$
for each $m \in M \backslash Y$
    $X_1 := X \cap \{m\}'$
    $Y_1 := X_1'$
    if $(X_1, Y_1) \notin lNCandidates$
        then
        Add (X_1, Y_1) to $lNCandidates$
        $count(X_1, Y_1) := 1$
        else
        $count(X_1, Y_1) := count(X_1, Y_1) + 1$
    if $(|Y_1| - |Y|) = count(X_1, Y_1)$ **then**
        Add (X_1, Y_1) to $lowerNeighbours$

Figure 1: Find Lower Neighbours algorithm.

both from a theoretical and an experimental point of view. We will return to this in the conclusion.

3.3 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 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 [Godin *et al.*, 1989], [Godin *et al.*, 1993], [Carpineto and Romano, 1996b], and the REFINER system discussed earlier [Carpineto and Romano, 1998].

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 ([Carpineto and Romano, 1995], [Carpineto and Romano, 1996a]).

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.

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 [Cole *et al.*, 2003] 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 ([Wille, 1984], [Vogt *et al.*, 1991],[Vogt and Wille, 1995], [Stumme, 1998]). 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 [Gershon *et al.*, 1998], 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.

4 Accessing and Mining Web Retrieval Results with CREDO

In the preceeding sections, we have argued that there is a lot of scope for application of FCA to interactively exploring the content of text collections. One particularly important form of text collection is represented by the Results Page returned by Web search engines. Now we address the issue of mining Web retrieval results using a concept lattice approach. We start by discussing other approaches, then present the system CREDO.

4.1 Approaches to Visualising Web Retrieval Results

Recent research has focused on the lack of effectiveness of Web search engine interfaces, which suffer from a lack of a concise representation of the content of all retrieved documents. The unmanageably large response sets of Web search engines compounded with their low precision often make the perusal of document summaries ineffective, time-consuming, and frustrating for the user.

A number of approaches have been presented, including query-biased summaries [Tombros and Sanderson, 1998], hierarchical query-biased summaries [Sweeney *et al.*, 2002], query term hits between documents [Berenci *et al.*, 2000], and arrangement of results using an auditorium seating metaphor [Terveen and Hill, 1998]. The best-known method is, perhaps, to use hierarchical clustering (e.g., [Zamir and Etzioni, 1999]), which is also offered by commercial search engine services such as Vivisimo (*http://vivisimo.com*) or Clustered Hits (*http://www.clusteredhits.com*).

The clusters built from Web documents are labeled using the document descriptions and the resulting structure is offered to the user for browsing, starting from the top clusters. The effect for the user is a combination of query-based (e.g., Google) and category-based (e.g., Yahoo!) Web search methods. It allows the user to focus on some weakly-specified subject (by a query), and then drill down through the hierarchy that has been created on the fly in response to that particular query.

However, hierarchical clustering of Web results also presents some drawbacks. First, valuable or relatively rare clusters can be omitted in the resulting hierarchy due to the use of similarity metrics and heuristic choices during cluster formation [Carpineto and Romano, 2000]. Second, once the hierarchy has been constructed, a label for each cluster must be found. This step is very important for the task at hand, because the labels guide the process of cluster selection and subsequent refinement on the part of the user. However, it is often difficult to find a good description for a set of documents. Finally, the user navigates through a strict hierarchy, which does not easily permit recovery from bad decisions.

Most of these limitations can be overcome using concept lattices: the set of clusters is complete and each element is formally justified, cluster labeling is integrated with cluster formation, the structure is a lattice instead of a hierarchy. In addition to being theoretically appealing, on-line mining of Web results using concept lattices is technically feasible, as demonstrated by the the system CREDO.

4.2 Design and Implementation of CREDO

Figure 2 shows the architecture of CREDO. CREDO, which stands for Conceptual REorganization of DOcuments, takes as input a user query. The query is forwarded to an external Web 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 terms that describe each returned document is generated, from which a conceptual structure representing all retrieval results is built. Such a conceptual structure, as will be explained below, can be seen as a variant of a concept lattice; we will refer to it as a *CREDO hierarchy*. The last steps consist of visualizing the CREDO hierarchy and managing the subsequent interaction with the user, who may browse through the concepts and display the associated documents.

CREDO has been implemented in *Shark*, a Lisp-like language being developed by the second author of this paper that is automatically translated into ANSI C and then compiled by gcc compiler. CREDO runs on a Web server as a CGI application that can be invoked from any Web browser; it is available for testing at *http://credo.fub.it*. In the following sections we detail the working of CREDO's main blocks.

4.2.1 Interaction with the Search Engine

The interaction between CREDO and the search engine is handled through SOAP (*http://www.w3.org/TR/SOAP*), an XML-based protocol widely used for

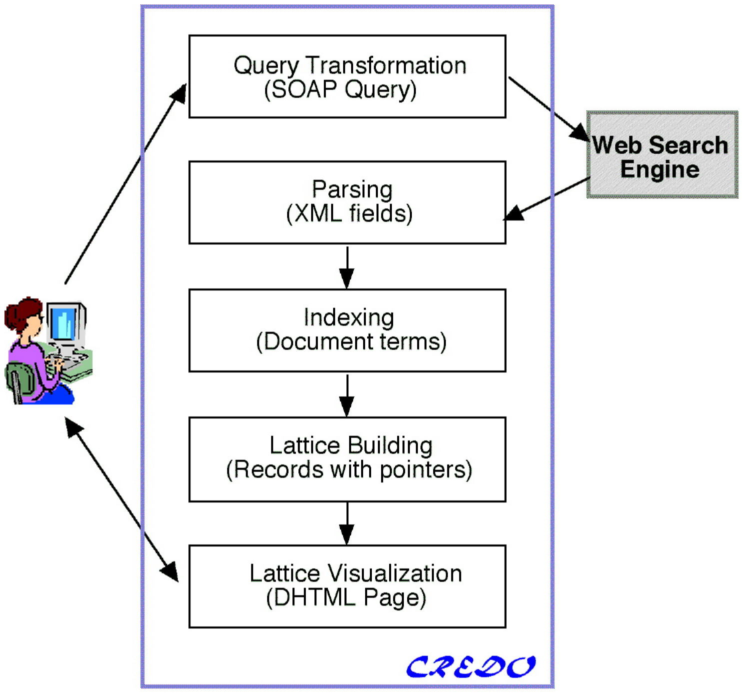

Figure 2: CREDO's architecture.

exchanging structured and typed information over the Web. An example of a search engine that accepts queries from an authorized computer using this protocol is Google, through its Web APIs service (*http://www.google.com/apis/*).

In our case, we need to encode the information concerning the query and the retrieval results. After the query has been typed in, a SOAP call to the search engine is generated that will contain the query terms as well as other pieces of information such as the number of result pages required.

CREDO collects the first 100 documents retrieved by the search engine. This number is a good compromise between having few documents (which can be better examined sequentially and are possibly too homogeneous in content to give rise to useful subgroups) and a large number of documents (which are possibly not relevant and cannot be processed with reasonable response times).

The output of the search engine is returned as a SOAP message, analogous to that received as an input, which CREDO transforms into a tree-based internal representation. Several attributes of interest for the extraction of a subset of index terms are present, including a "title" and "snippet" for each retrieved document.

It is also possible for CREDO to query a search engine in a direct manner via the HTTP protocol, and then acquire the results by parsing the returned HTML pages. This approach has the advantage that the utilization of the search engine may not be subject to external constraints. However, the parsing process is prone to error and is sensitive to changes in the search engine interface.

4.2.2 Indexing of Retrieval Results

One possibility is to use the method described in Section 3.1, considering as the collection at hand the first retrieved documents. This method, however, requires that the description of each document should be detailed enough to have a rich representation for the document's language model.

This is not the case in our application, because the document summaries are very short, whereas the full-text documents are not available unless we download the original documents from the Web (which may take too long). On the other hand, we can take advantage of the document structure.

CREDO considers just the information contained in the results returned by the search engine, focusing on the elements that best describe the content of the documents. Each document is indexed by two sets of terms, one for the title and one for the snippet, extracted from the values of the corresponding attributes in the SOAP message returned by the engine. The extraction of the terms is preceeded by simple document cleaning, which is the same for both attributes and consists of (i) identifying all tokens formed by alphanumerical characters, (ii) converting upper case characters to lower case characters, (iii) stripping numbers, and (iv) removing stop words.

4.2.3 Construction of the CREDO hierarchy

One of the main difficulties is that the concept lattice of the retrieved documents may contain many irrelevant concepts resulting from spurious combinations of the document terms. This problem is especially relevant to the set of coatoms (i.e., the lower neighbours of the top element of the lattice), which must be shown to the user at the beginning of the interaction and should give an immediate idea of the main subjects into which the results can be grouped.

To address this problem, one can use a very limited number of terms per document (e.g., the title of the document). The advantage is that in this way we reduce the possibility that two documents that are different in content share some word by chance. However, by doing so it is likely that many documents will not share any term with the other documents, thus remaining ungrouped (i.e., their document concept will be directly linked to the top and the bottom elements of the concept lattice).

To avoid this drawback, CREDO takes a hybrid approach, in which the lower levels of the CREDO hierarchy are built using a larger set of terms than those used to build the top level. First, the top element of the concept lattice along with the coatoms (i.e., the lower neighbours of the top element), are built, using only the title terms. If some query term is not contained in the title, which is not usually the case for Web searches, it is added to the title to make sure that there is a natural starting point for browsing the document lattice. Then the extent of each coatom is increased by including the documents that contain the concept terms in their snippet. This amounts to expanding the description of each document with those snippet terms which are also concept terms. The top element with the modified coatoms are the top level of the CREDO hierarchy. At this point, the lower levels are built using the expanded document representation.

This can be seen as a two-step classification procedure, in which the first layer identifies the main topics and the other layers contain the subtopics of each main topic. Clearly, the resulting clustering structure is not a true concept lattice, in the sense that it cannot be seen as the concept lattice of a specific context. In particular, it is not equivalent to the concept lattice which would be obtained if we indexed each document by taking the union of the terms contained in the title and in the snippet.

The implementation of the hybrid strategy described above makes use of the *Find Lower Neighbours* algorithm described in Section 3.2, which is applied iteratively to generate the levels of the CREDO hierarchy in a top-down, breadth-first manner. The only caution is that different contexts must be used as input to the *Find Lower Neighbours* algorithm to take into account the expansion of the document representation. The coatoms are found using only the title terms as document intents; the other invocations of the *Find Lower Neighbours* algorithm are performed using the expanded context.

The pseudo-code of the algorithm, called *Find CREDO Hierarchy*, is shown in Figure 3. For the sake of generality, the CREDO hierarchy is seen as a set of concepts C and of a set of edges E, where the edges are ordered pairs of concepts (c_1, c_2) such that $c_1 \prec c_2$, i.e., c_1 is a lower neighbour of c_2. In practice, however, it is convenient to implement each concept as a record with pointers to its neighbours.

4.2.4 Visualization of CREDO's Results and Interaction with the CREDO hierarchy

After constructing the hierarchy, CREDO presents the user with the results. CREDO's interface is illustrated in Figure 4, relative to the query "jaguar". The top frame contains a search box with the submitted query, the lower left frame shows the lattice top along with its most numerous children using a hierarchical folders representation, and the lower right frame shows the documents associated with the currently selected concept (at the outset, the top level of the lattice). All the documents of one concept that are not covered by its displayed children

FindCREDOHierarchy
Input: Context (G, M, I_{titles}), document snippets
Output: The CREDO hierarchy $CH = (C, E)$

```
C := {(G, G')}   /* Using (G, M, I_titles) */
E := ∅
coatoms := FindLowerNeighbours((G, M, I_titles), (G, G'))
I_exp := I_titles
for each (X, Y) ∈ coatoms
    for each g ∈ G
        if g ∉ X and Y ⊆ snippet(g) then
            X_exp := X ∪ {g}   /* Expand extent of coatom */
            I_exp := I_exp ∪ {g, Y}   /* Expand object representation */
    C := C ∪ {(X_exp, Y)}   /* Create child of top element of CH */
    Add edge (G, G') → (X_exp, Y) to E
    /* The following statements build the lower levels iteratively */
currentLevel := C\{(G, G')}
while currentLevel ≠ ∅
    nextLevel := ∅
    for each (X, Y) ∈ currentLevel
        lowerNeighbours := FindLowerNeighbours((G, M, I_exp), (X, Y))
        for each (X_1, Y_1) ∈ lowerNeighbours
            if (X_1, Y_1) ∉ C then
                C := C ∪ {(X_1, Y_1)}
                nextLevel := nextLevel ∪ {(X_1, Y_1)}
            Add edge (X_1, Y_1) → (X, Y) to E
    currentLevel := nextLevel
```

Figure 3: *Find CREDO Hierarchy* algorithm.

are grouped in a dummy concept named "other" (in Figure 4, "other" contains 38 out of the 100 documents associated with its parent).

Note that the sum of the documents covered by the children may exceed the number of documents contained in their parent, because the same document can be assigned to multiple children. In Figure 4, for instance, the top element of the CREDO hierarchy contains the 100 documents returned by the search engine, whereas the sum of the documents contained by its children is equal to 114.

The user can click on one concept and see its children, thus narrowing down

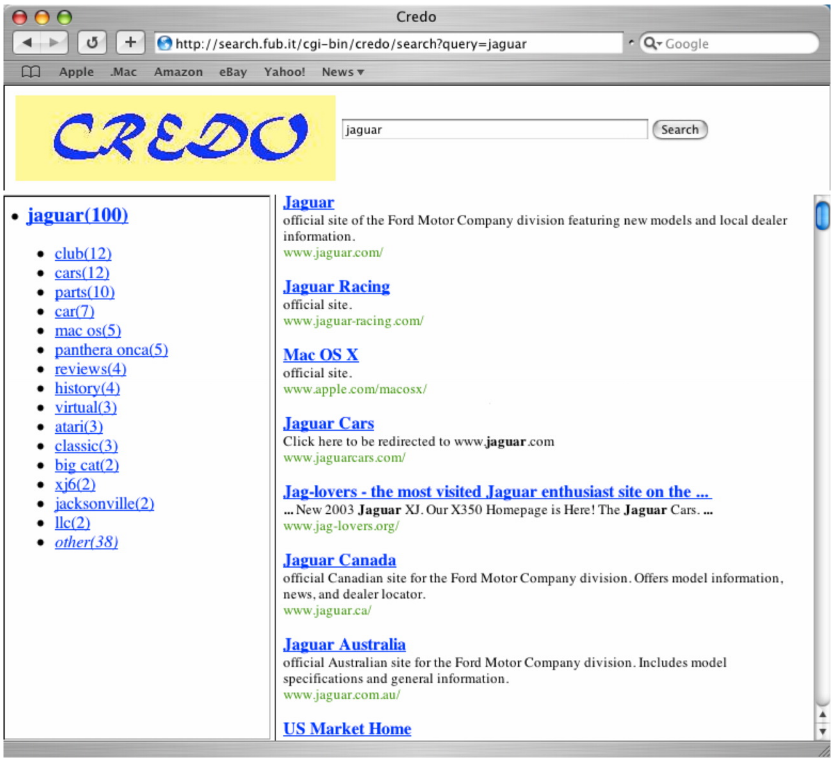

Figure 4: Results of the query "jaguar".

the scope of the search. This operation can be repeated on the newly-displayed concepts to further narrow the scope of the search, or it can be performed on other top concepts to browse unexplored branches.

To manage the interaction between the user and CREDO through a Web connection different technical solutions are conceivable. One possibility is to feed the client with just the top level and then provide the lower levels on demand, but this increases the number of interactions between client and server and requires proper handling of the *state* on the part of the server. An alternative approach is to compute the whole lattice and send out the results of all possible concept selections at once. This requires more bandwidth but is simpler to implement and allows faster response times during the interaction. CREDO adopts the latter solution.

As CREDO allows the user to browse through the query results, this can be seen as a form of query refinement, in which the emphasis is on the efficient construction of a small but informative concept lattice. Furthermore, CREDO can be seen as a tool for query disambiguation and content categorization. This is especially useful for Web searches, because as there are a huge number of highly rich and heterogeneous documents, the retrieval results may easily contain potentially relevant documents that cover entirely different subjects. This point is clearly illustrated in the next section.

4.3 CREDO at Work

Consider again the query “jaguar”. This is an inherently ambiguous word on the Web. The output of CREDO (see Figure 4) clearly reflects this fact, with concepts such as *cars*, *parts*, *atari*, and *xj6* (referring to the Jaguar car brand), or *mac os* (one of the latest operating systems of MAC), or *panthera onca* and *big cat* (i.e., the animal), or *jacksonville* (i.e., the American football team), or even *clubs* (there are in fact plenty of clubs called jaguar).

This example shows the utility of CREDO to disambiguate a user query and to quickly focus on the documents relevant to the intended meaning.

As a further illustration, consider the query “xml”. The top concepts created by CREDO (see Figure 5) show several useful main topics referring to it, such as *markup*, *software*, *tools*, *free*, *rdf*, *editor*, *1.0*, *apache project*, *python processing* etc.. An example of a secondary topic (of the primary topic *editor*) is *windows free*, which contains a document about a free XML editor for Windows. Note that the same document could be reached through the path: *xml* - *free* - *windows editor*. Neither path is better than the other, but one may better fit a particular user’s paradigm or need.

This example illustrates the greater flexibility of navigating CREDO’s results, as compared to using a strict hierarchy, because the same piece of information can be reached through multiple paths.

The last example concerns the query “claudio carpineto” (in quotation marks), shown in Figure 6. There emerge many interesting concepts, some of which are multi-word concepts; e.g., *fondazione ugo bordoni* (the affiliation of Claudio Carpineto), *gianni amati* (the name of a colleague, with whom he has authored several papers), *machine learning* (one of his research interest). In fact, in the limited context represented by the results of “claudio carpineto”, each word in any of those pairs always co-occurs with the other word in the pair. For instance, “gianni” univocally determines and is univocally determined by “amati”, “machine” by “learning”, etc.

This is an interesting feature of concept lattice-based mining of Web results, because it permits the discovery of deterministic or causal associations between words that hold in the set of results.

The “claudio carpineto” example is also interesting for other reasons. As shown in Figure 6, the coatom with the largest number of documents is *romano*, the surname of the second author of this paper, with whom Claudio Carpineto

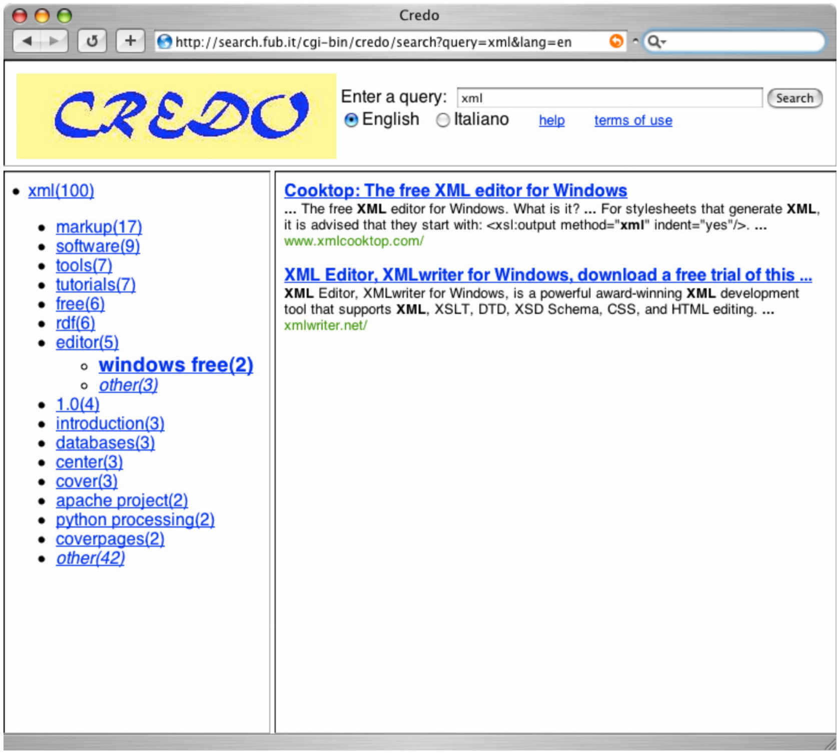

Figure 5: Selecting *editor* and *windows free* in the "xml" results.

has authored most of his paper. Somewhat surprisingly, a concept *giovanni romano* has not been generated, nor a concept *giovanni* as a subconcept of *romano*, in spite of the fact that the word *giovanni* co-occurs with the word *romano* in each of the 34 documents associated with the concept *romano*. On closer inspection, we found that there is a nonrelevant document in the retrieved results with the words *valerio romano* in the title which prevents CREDO from forming the coatom *giovanni romano*. The coatom *romano* is created instead, and at this point the word *giovanni* is no longer considered for generating the subsequent concepts.

Although in this case the failure in producing the concept *giovanni romano* or *giovanni* can be attributed to a spurious document in the set of retrieved results, it may happen that a very frequent term such as *giovanni* does not give rise to any concept because it is not contained in any title. To account for this anomalous situation, one could use a more sophisticated policy for determining the set of index terms, for instance by also accomodating no-title terms that occur comparatively more frequently in the set of retrieved documents than in

Figure 6: Results of the query "claudio carpineto" (in quotation marks).

a reference corpus (see Section 3.1).

It should also be noted that the second and third child of the top concept are *information* and *retrieval*, respectively, with 25 and 24 documents. One might think that, as in the *giovanni romano* case, the concept *information retrieval*, which is one of the research interests of Claudio Carpineto, has not been erroneously generated due to some unfortunate circumstance. In fact, the concept *information* is rightly more general than *information retrieval* because there are retrieved documents that reference the *Information Systems* group at Fondazione Ugo Bordoni, of which Claudio Carpineto has been the head, as well as the use of *information theoretic* approaches in some of the papers authored by Claudio Carpineto.

5 Conclusions

Building on the properties of a document lattice, we have developed CREDO, a system for inspecting Web retrieval results. We have shown that CREDO is useful for quickly retrieving the items with the intended meaning in the case of an ambiguous query, as well as for highlighting the contents of the documents that reference a given entity.

Compared to other clustering approaches to mining Web retrieval results, CREDO has a stronger theoretical basis and a greater flexibility. In addition, it permits the extraction of relevant multi-word concepts that are specific to the query at hand.

We plan to improve CREDO along two main directions. The first is the use of more sophisticated policies for document indexing, e.g., by scoring the importance of terms or by taking advantage of word proximity. The second is an investigation of the main trade-offs involved in the implementation of CREDO, such as analyzing a small or large number of retrieved documents, using document snippets or full text documents, performing single- or multi-keyword indexing, constructing a partial or full concept lattice, using simple or sophisticated visualization schemes, allowing single or multiple interaction modes.

We believe that the development of CREDO will help increase the scope of FCA for IR. Although several 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 on-line, concept lattice-based tools for performing Web searches such as CREDO is a step in this direction. To enable a greater and lasting impact of FCA on IR, the following points also deserve attention.

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., [Spink and Saracevic, 1997], [Berenci *et al.*, 2000]).

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 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.

Acknowledgements

We would like to thank two anonymous reviewers for many helpful comments and suggestions.

References

[Agosti *et al.*, 1995] M. Agosti, M. Melucci, and F. Crestani. Automatic authoring and construction of hypertexts for information retrieval. *ACM Multimedia Systems*, 3:15–24, 1995.

[Amati and van Rijsbergen, 2002] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.

[Berenci *et al.*, 2000] 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.

[Bordat, 1986] J.P. Bordat. Calcul pratique du treillis de Galois d'une correspondance. *Math. Sci. Hum.*, 96:31–47, 1986.

[Card *et al.*, 1983] S. Card, T. Moran, and A. Newell. *The psychology of human-computer interaction.* Lawrence Erlbaum Associates, London, 1983.

[Carpineto and Romano, 1993] 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.

[Carpineto and Romano, 1994] 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.

[Carpineto and Romano, 1995] 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.

[Carpineto and Romano, 1996a] C. Carpineto and G. Romano. Information retrieval through hybrid navigation of lattice representations. *International Journal of Human-Computer Studies*, 45(5):553–578, 1996.

[Carpineto and Romano, 1996b] C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):1–28, 1996.

[Carpineto and Romano, 1998] 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.

[Carpineto and Romano, 2000] C. Carpineto and G. Romano. Order-Theoretical Ranking. *Journal of the American Society for Information Science*, 51(7):587–601, 2000.

[Carpineto and Romano, 2004] C. Carpineto and G. Romano. *Concept Data Analysis: Theory and Applications*. John Wiley & Sons, 2004.

[Carpineto *et al.*, 2001] 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.

[Carpineto *et al.*, 2002] 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.

[Cole and Eklund, 2001] 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.

[Cole and Stumme, 2000] 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.

[Cole *et al.*, 2003] 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.

[Deerwester *et al.*, 1990] S. Deerwester, S. T. Dumais, W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[Efthimiadis, 1996] 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.

[Ferré and Ridoux, 2000] 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.

[Ganter and Wille, 1999] B. Ganter and R. Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999.

[Ganter, 1984] B. Ganter. Two basic algorithms in concept analysis. Technical Report FB4–Preprint No. 831, TU Darmstadt, Germany, 1984.

[Gershon *et al.*, 1998] 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.

[Gifford *et al.*, 1991] 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.

[Godin and Mili, 1993] 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.

[Godin *et al.*, 1986] R. Godin, E. Saunders, and J. Jecsei. Lattice model of browsable data spaces. *Journal of Information Sciences*, 40:89–116, 1986.

[Godin *et al.*, 1989] 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.

[Godin *et al.*, 1993] 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.

[Godin *et al.*, 1995] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on Galois lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[Gopal and Manber, 1999] 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.

[Karp *et al.*, 1992] 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.

[Kuznetsov and Obiedkov, 2002] 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.

[Lindig, 1995] 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.

[Lindig, 2000] 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.

[Lucarella *et al.*, 1993] 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.

[Maarek *et al.*, 1991] 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.

[Norman, 1986] 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.

[Nourine and Raynaud, 1999] L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71:199–204, 1999.

[Pedersen, 1993] 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.

[Porter, 1980] M. F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[Priss, 1997] 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.

[Priss, 2000] U. Priss. Lattice-based information retrieval. *Knowledge Organization*, 27(3):132–142, 2000.

[Robertson *et al.*, 1998] 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.

[Rock and Wille, 2000] T. Rock and R. Wille. Ein Toscana-Erkundungssystem zur Literatursuche. In G. Stumme and R. Wille, editors, *Begriffliche Wissensverarbeitung. Methoden und Anwendungen*, pages 239–253. Springer, Berlin, Germany, 2000.

[Salton and McGill, 1983] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[Snelting and Tip, 1998] 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.

[Soergel, 1967] D. Soergel. Mathematical analysis of documentation systems. *Information storage and retrieval*, 3:129–173, 1967.

[Spink and Saracevic, 1997] 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.

[Spoerri, 1994] 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.

[Stumme, 1998] G. Stumme. Local scaling in conceptual data systems. In *Proceedings of the 6th International Conference on Conceptual Structures*, pages 308–320, Montpellier, France, 1998.

[Sweeney *et al.*, 2002] S. Sweeney, F. Crestani, and A. Tombros. Mobile delivery of news using hierarchical query-biased summaries. In *Proceedings of ACM SAC 2002, ACM Symposium on Applied Computing*, pages 634–639, Madrid, Spain, 2002.

[Terveen and Hill, 1998] L. Terveen and W. Hill. Finding and visualizing intersite clan graphs. In *Proceedings of ACM CHI'98: Human Factors in Computing Systems*, pages 448–455, Los Angeles, CA, USA, 1998.

[Tombros and Sanderson, 1998] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Reasearch and Development in Information Retrieval*, pages 3–10, Melbourne, Australia, 1998.

[van der Merwe and Kourie, 2002] 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.

[Vogt and Wille, 1995] 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.

[Vogt *et al.*, 1991] 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.

[Wille, 1984] R. Wille. Line diagrams of hierarchical concept systems. *Int. Classif.*, 11(2):77–86, 1984.

[Willet, 1988] P. Willet. Recent trends in hierarchic document clustering: a critical review. *Information Processing & Management*, 24(5):577–597, 1988.

[Zamir and Etzioni, 1999] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *WWW8/Computer Networks*, 31(11–16):1361–1374, 1999.

[Zhai and Lafferty, 2001] 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.