Order-theoretical ranking

CLAUDIO CARPINETO *
Fondazione Ugo Bordoni,
Via B. Castiglione 59, 00142 Rome, Italy
E-mail: carpinet@fub.it
Fax: +39-06-54804405
Tel: +39-06-54803426

GIOVANNI ROMANO
Fondazione Ugo Bordoni,
Via B. Castiglione 59, 00142 Rome, Italy
E-mail: romano@fub.it
Fax: +39-06-54804405
Tel: +39-06-54803423

Abstract

Current best-match ranking (BMR) systems perform well but cannot handle word mismatch between a query and a document. The best known alternative ranking method, hierarchical clustering-based ranking (HCR), seems to be more robust than BMR with respect to this problem, but it is hampered by theoretical and practical limitations. We present an approach to document ranking that explicitly addresses the word mismatch problem by exploiting interdocument similarity information in a novel way. Document ranking is seen as a query-document transformation driven by a conceptual representation of the whole document collection, into which the query is merged. Our approach is based on the theory of concept (or Galois) lattices, which, we argue, provides a powerful, well-founded, and computationally-tractable framework to model the space in which documents and query are represented and to compute such a transformation. We compared information retrieval using concept lattice-based ranking (CLR) to BMR and HCR. The results showed that HCR was outperformed by CLR as well as by BMR, and suggested that, of the two best methods, BMR achieved better performance than CLR on the whole document set while CLR compared more favorably when only the first retrieved documents were used for evaluation. We also evaluated the three methods' specific ability to rank documents that did not match the query, in which case the superiority of CLR over BMR and HCR (and that of HCR over BMR) was apparent.

* *To whom all correspondence should be addressed*

1. Introduction

Information retrieval is concerned with retrieving the documents of interest to a user from a natural-language document collection. The typical retrieval setting consists of a user submitting a query to the system, usually in free-text natural language, and the system returning a list of possibly relevant documents in ranked order. Most of the major ranking systems that are in use today, although working from a very different basis, can be seen as performing two main operations: building an internal representation of queries and documents first, and then scoring the query representations against the document representations to produce a ranked document list. In the first stage, queries and documents are usually represented as weighted term vectors, using such diverse weighting schemes as SMART's classical *tf* x *idf* approach (Salton, 1971; Salton and Buckley, 1988), Croft and Harper (1979)'s probabilistic indexing method, INQUERY's network inference model (Turtle and Croft, 1991; Callan et al., 1992), OKAPI's 2-Poisson model (Robertson and Walker, 1994; Walker *et al*., 1997), and the Cornell variant of the OKAPI algorithm (Singhal *et al*., 1995). Once the weighted term vectors have been computed, the matching of queries against documents is customarily performed in the same manner, i.e., by computing the dot product between corresponding weighted term vectors.

These best-match retrieval systems are highly efficient and have shown to perform well in many operational situations, including the TREC environment (Voorhees and Harman, 1998). However, they are limited by their inability of dealing with word mismatch. When translating an information requirement into a query for a document retrieval system, a user must convert concepts involved in his requirement into query terms which will not necessarily match the terms used by the authors to describe the same concepts in their documents. This is the well known vocabulary problem, described in (Furnas et al., 1987), two specific important aspects of which are polysemy (same word to describe different things) and synonymy (different words to describe the same thing). The severity of the vocabulary problem tends to decrease as queries get longer, but it may be exacerbated in applications where the queries are very short, as is usually the case in Web-based retrieval.

One traditional solution to the vocabulary problem is to automatically or interactively expand or refine the query using various knowledge sources, such as the relevance feedback provided by users (Harman, 1992), the top ranked documents retrieved by the original query (Xu and Croft, 1996 Carpineto *et al*., 1999), collection-specific lexical networks (Cooper and Byrd, 1997), and general purpose thesauri (Voorhees, 1993). The query modification approach attempts to extend the capabilities of conventional, best-match ranking systems to recover from word mismatch without modifying the assumptions of their underlying model. A more fundamental solution to word mismatch is to try to exploit the relationships in content which exist between the documents in the collection when deciding which documents are to be

retrieved in response to a query. Of this alternative strategy to best-match ranking, the most well known approach is cluster-based ranking, where a query is ranked not against individual documents but against a hierarchically grouped set of document clusters. The rationale for this method is the cluster hypothesis (van Rijsbergen, 1979; Hearst and Pedersen, 1996), which states that relevant documents tend to be more similar to each other than non-relevant documents. Hierarchical cluster-based ranking (hereafter often referred to as HCR) does go some way towards compensating for the vocabulary problem but does not solve it all, because this approach suffers from theoretical as well as practical limitations.

In this work we present a novel approach to document ranking that has clear potentials to deal with word mismatch. In the same vein as HCR, it is based on building a cluster structure from the set of documents, but the clustering methodology is completely different. Instead of grouping the set of documents using some similarity metrics, it is based on the recognition and ordering of set inclusion relations between the terms describing the documents. The ranking strategy of our approach is also different from HCR. Rather than computing a similarity between individual document clusters and a query, we use a clustered representation of the whole document collection to drive a transformation between the representation of a query and the representation of each document. In practice, the query is merged into a conceptually-clustered document space and the similarity between the query and each document is seen as a function of the length of the shortest path linking the query to the document. The mathematical tool used to implement this approach is the concept (or Galois) lattice associated with a term-document relation (Wille, 1984; Davey and Priestley 1990).

The theory of concept lattices has already been used in information retrieval applications (Godin et al., 1989 and 1993; Carpineto and Romano, 1996a and 1996b), but it has been customarily employed to support user interface design. In fact, one of the motivations of our research was a desire to extend the scope of such a theory to fully-automatic retrieval tasks. We show that concept lattice-based ranking (CLR) has a clear and sound semantics and that CLR helps overcome some limitations of HCR related to the dissatisfying theoretical assumptions and operational implementations of the latter method. We then evaluate the retrieval effectiveness of CLR in contrast to BMR and HCR. The results are promising. In particular, they support the view that CLR performs better than HCR, and that CLR may be seen as an alternative to BMR, especially for some retrieval tasks and at least for collections of small-medium size. In addition, our experiment indicates promising synergistic combinations between CLR and BMR for large databases.

This article is organized as follows. Section 2 examines some inherent limitations of HCR. Section 3 provides an abstract, but intuitive, view of document ranking as a representation transformation in a conceptual space. Section 4 formally introduces concept lattice-based

ranking, provides a detailed illustrative example, and examines implementation and computational requirements. Section 5 reports the results of comparing the three ranking methods on different retrieval tasks, including the specific ability of ranking documents that do not match a given query. Section 6 considers the scaling issue. Section 7 contains a discussion of related work, including cluster-based ranking, earlier applications of concept lattices to information retrieval, and other approaches based on document similarity. Section 8 concludes the paper with a summary and some directions for future work.

2. Limitations of hierarchical cluster-based ranking systems

Hierarchical clustering methods take as input a matrix of document-to-document distances, based on some similarity function, and iteratively merge the most similar pair of distinct clusters, using some clustering strategy (e.g., single link, complete link, group average, Ward's method), until there is only one cluster. Once the clustering hierarchy has been built, an incoming query is ranked against neighborhoods of this structure, using some search strategy (top down, bottom up, or optimal search) and some query-cluster similarity function**.** The result of HCR is a partially-ordered sets of documents, because the documents in each cluster are equally ranked; however, a totally-ordered list can be easily obtained from it by individually ranking the documents in each cluster against the query.

From a conceptual point of view, HCR seems to be more robust than BMR with respect to word mismatch, because it takes into account both the interdocument similarity and the similarity between a query and the individual documents. Still, it appears too limited to reveal the richness and diversity of relevance relationships between queries and documents with different surface representations. For one thing, this is due to some well known empirical weaknesses of hierarchical clustering methods for information retrieval applications, such as the relatively small number of free parameters that they have (essentially, n clusters for n documents) and their incapability of performing multiple or crossed classifications (Deerwester et al., 1990; Carpineto and Romano, 1996a). In addition, perhaps more importantly, we believe that a major problem with this approach is the lack of firm theoretical foundations by which to characterize the document ranking in terms of the query and document descriptions. Shaw *et al*. (1997) pointed out that clustering algorithms may not reveal the natural structure of a set of documents and that search strategies exploiting the topical relatedness of queries and clusters may not select the most effective clusters of documents. We elaborate on this, showing some inherent limitations involved in both of these two steps.

We first focus on the process of cluster formation, arguing that the method used by HCR to group the documents into clusters may easily involve at some point some heuristic decision to choose between equivalent cluster hierarchies. In particular, we will see that even for a very

simple set of documents, the resulting hierarchy may contain certain clusters while failing to produce other equally good clusters. We then consider the ordering of the generated clusters in response to a query, which is usually based on a best-match function between the query and some clusters representation. The result of this step may also vary depending on how we choose the main parameters involved here. Overall, we show that the reliance of HCR on several and loosely-coupled similarity or distance measures may make the output result difficult to control and prone to error. In fact, the use of HCR may easily result, even for simple tasks and in the presence of typical assumptions and choices, in a failure to discriminate between documents that have manifestly different degrees of relevance for a certain query. In the rest of this section we make these points more precise.

Consider the simple document-term relation described in Table 1. Clearly, BMR methods would always fail to discriminate between documents that have no terms in common with a query. For instance, if the query is equal to "T1", documents D2 and D3 would be equally ranked by BMR, whereas D2 appears to be more relevant than D3 for the given query; a dual situation holds for documents D2 and D1 with respect to query "T4".[1] Let us see what happens if we use a HCR method. Clusters D1-D2 and D2-D3 are equivalent, no matter which interdocument similarity function and which cluster strategy we have chosen. Thus, we may have two possible cluster hierarchies for the set of documents at hand, depending on how we break the tie (see Figure 1). Suppose, for simplicity and generality, that incoming queries are then scored against the clusters using optimal search. For query "T1", the left hierarchy would produce a correct document ranking, because the best matching cluster would be D1, followed by cluster D1-D2. For query "T1", however, the right hierarchy would rank cluster D1-D2-D3 right after cluster D1, which would cause documents D2 and D3 to be equally ranked. A dual situation holds for query "T4", with the first hierarchy behaving wrongly and the second correctly. This behavior will be observed for any choice of the query-cluster similarity function.

Not only can HCR fail to discriminate between documents which do not match the query (hereafter often referred to as non-matching documents), i.e., D2 versus D3 for query "T1", D1 versus D2 for query "T4", but it can also fail to discriminate between matching and non-matching documents. Assume without loss of generality that the query-cluster similarity function is computed by taking the inner product with cosine normalization between the query term vector and a cluster term vector formed by the set of terms contained in all cluster's documents, weighted with their frequency in the cluster itself. These are typical choices for the normalization factor and the cluster representative (e.g., Griffiths *et al*., 1986). If the incoming query is equal to "T2", then, for the left hierarchy, the best matching cluster is D1-D2;

[1] One should be aware that in a more complex context these considerations may represent a rough approximation of what determines relevance. Spink and Saracevic (1997), for instance, show that no matter how judicious their choice, the majority of search terms will retrieve both relevant and non-relevant documents.

documents D1 and D2 are thus ranked equally, and ahead of D3, which is correct. However, for the same query "T2", using the right hierarchy would result in selecting the cluster D1-D2-D3 (with a query-similarity score of $2 / (10)^{1/2}$) before cluster D2-D3 (whose query-similarity score is equal to $1 / (6)^{1/2}$), in which case the documents D1, D2, and D3 would be equally ranked. A dual situation holds for query "T3", with the right hierarchy behaving correctly and the left hierarchy behaving wrongly.

In order to overcome these limitations it seems thus necessary to attack the fundamental representational limitations of current HCR systems, including the limited expressive power of the structure representing the interdocument similarity and the need of mapping entities with different representations. Our approach addresses both these issues using a richer and unique framework for representing documents and query and comparing them.

Table 1. A simple document-term relation

	T1	T2	T3	T4
D1	x	x		
D2		x	x	
D3			x	x

Figure 1. Equivalent cluster hierarchies derived from the set of documents shown in Table 1.

3 Viewing document ranking as a query transformation in a conceptual document space

The starting point is that the matching between a query and a document can be described in terms of a sequence of operations that transforms one into the other. We envisage that the basic operations to transform the term vector representing a query into the term vector representing a document are a form of term addition and term deletion, while the length of the sequence that accomplishes the desired transformation is a measure of the similarity between the two vectors. Note that one apparently straightforward way to implement this idea is to simply count the terms that are not shared by the query vector and the document vector, or, alternatively, to use some related similarity measure, such as Dice or Jaccard coefficient (Salton and McGill, 1983). Such an approach, however, would amount to a simplistic form of best-match retrieval, with very limited ranking capabilities. Neither would it be able, in general, to discriminate between

non-matching documents. In the simple example considered in Table 1, for instance, it could not rank D2 ahead of D3, for query “T1”, because the two documents would get the same distance score. Thus, viewing the distance between a query and a document as a transformation process between the corresponding representations is not sufficient alone.

The second essential ingredient of our approach is that the query-document transformation should be driven by an internal conceptual structure of the database being searched. The intuition behind this view is the following. Assume for a moment that we are able to extract from a document collection a set of concepts contained in it, where such concepts can be thought of as the queries that can be naturally satisfied by some document in the collection. Assume also that, for each concept, it is available some procedure, say a nearest neighbors operator, that determines which are its most related concepts. The set of concepts together with the nearest neighbors operator provide us with a tool for analyzing the query-document transformation, provided that the query can be mapped onto some concept. The set of concepts defines the space of admissible query transformations; the nearest neighbors operator tells us how to make a transition from a query to another. At this point the problem of computing a sequence of operations that transforms a query into a document can be cast as a breadth-first search through the space of admissible queries, as determined by the query and the collection at hand. The initial state is a concept corresponding to the query, while the successor states are specified by the nearest neighbor operator. The search ends as soon as a concept is reached that corresponds to the document representation. A document score is given by the length of the shortest path linking the initial and the final state, while the final ranked document list is obtained by arranging the documents in the increasing order of score.

The next question is: is there any conceptual representation of a document collection that satisfies these requirements? We argue that concept (or Galois) lattice is a good candidate. The concept lattice associated with a document collection contains a particular set of concepts inherent to the collection that can be ordered by generality. Roughly, each concept is a complete set of co-occurrences, where by this we mean a set of terms appearing jointly in a set of documents and such that those documents do not have other terms in common. Concept lattices have the following desirable features: (i) the concepts have a natural interpretation from the point of view of characterizing the set of queries that can be satisfied by the collection, (ii) a user query can be consistently mapped onto the lattice, (iii) the concept ordering tells us how to gradually pass from one query to another, and (iv) the set of concepts in the lattice is sufficiently numerous to ensure a rich representation while being still tractable. In addition, the concept lattice can be automatically built from a document-term matrix, and its construction is computationally feasible.

Before we proceed, it is useful to consider again the example given in Section 2 as a simple illustration. As will become clearer in the following of the paper, the set of relevant concepts

contained in the concept lattice associated with the set of documents described in Table 1 and with query "T1" is shown in Figure 2. The highest ranked document is D1, because there is a link connecting the query to it, followed by document D2, through the path T1-T1T2-T2-T2T3, and by document D3, through the path T1-T1T2-T2-T2T3-T3-T3T4.

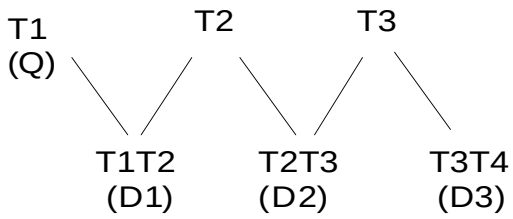

Figure 2. The set of ordered concepts derived from the collection in Table 1 and from query "T1".

In the next section we make the above considerations more precise. We start by introducing order-theoretical ranking formally; in particular, we define the concept lattice that can be generated from a document collection, and provide an algebraic treatment that yields the document ranking for an incoming query. Basic notions and formalisms are borrowed and adapted from the concept lattice theory (Wille 1984) as well as from the more general ordered set theory (Davey and Priestley 1990).

4. A ranking method based on the concept lattice theory

4.1. Formal description

As input data, we consider a binary relation between a set of documents (D) and a set of terms (T), usually called *context* in the concept lattice theory. In the information retrieval field, this is the usual document by term relation.

Definition 1 A context is a triple (D, T, I) where $I \subseteq D \times T$. We write *dIt*, meaning the document *d* has the term *t*.

To derive from a context the set of concepts, the following Galois connections are useful.

Definition 2 Let (D, T, I) be a context. For $X \subseteq D$ and $Y \subseteq T$, define:
$X' = \{t \in T \mid (\forall d \in X)\ dIt\}$
$Y' = \{d \in D \mid (\forall t \in Y)\ dIt\}$.

In other words, X' is the set of terms common to all documents in X and Y' is the set of documents possessing all the terms in Y.

Definition 3 Let (D, T, I) be a context. A concept is a pair (X, Y) where $X \subseteq D$, $Y \subseteq T$, X' = Y, and Y' = X; X and Y are called the *extent* and the *intent* of the concept, respectively.

Only the pairs (X,Y) that are complete with respect to I according to the given definition represent admissible concepts. In particular, a subset Y of T is the intent of some concept if and only if Y''=Y, in which case the unique concept of which Y is an intent is (Y',Y); a dual statement holds for the extent of a concept. From the point of view of document retrieval, each concept can be seen as a conjunctive query (the intent) along with the set of documents that satisfy it (the extent).

The set of all concepts of the context (D, T, I) is denoted by $\mathbf{C}$(D, T, I). It is possible to order this set of concepts by generality (specificity), in such a way that each concept contains a subset (superset) of the terms of its more specific (general) concepts .

Definition 4 Let $\mathbf{C}$(D, T, I) be the set of concepts of the context (D, T, I) and let (X_a, Y_a), (X_b, Y_b) $\in \mathbf{C}$(D, T, I). Then (X_a, Y_a) is subsumed ($\leq$) by (X_b, Y_b) if $X_a \subseteq X_b$, or, equivalently, if $Y_a \supseteq Y_b$.

$\mathbf{C}$(D, T, I) along with $\leq$ form a partially ordered set, that turns out to be a complete lattice[2] (Wille 1984).

Definition 5 Let ($\mathbf{C}$(D, T, I) ; $\leq$) be the concept lattice corresponding to the context (D, T, I) and let a, b $\in$ ($\mathbf{C}$(D, T, I) ; $\leq$). Then *a* is a nearest neighbor (>-<) of *b* if either $a < b$ and $a \leq z < b$ implies $z = a$, or $b < a$ and $b < z \leq a$ implies $z = a$. The latter conditions are demanding that there be no element z of ($\mathbf{C}$(D, T, I) ; $\leq$) with $a < z < b$ or $b < z < a$.

Observe that the subsumption relation determines, but is not determined by, the nearest neighbor relation. The nearest neighbor relation defines for each concept all its minimal conjunctive refinements (enlargements) with respect to the database at hand, in the sense that there is no choice of term addition (deletion) which would produce an intermediate concept.

---

[2] Recall that, given a non-empty ordered set P, if for all $S \subset P$ there exists a least upper bound and a greatest lower bound, then P is called a *complete lattice*.

Definition 6 Let $\mathbf{C}(D, T, I ; >\!\!-\!\!<)$ be the set of concepts of the context (D, T, I) together with the nearest neighbor relation and let $a, b \in (\mathbf{C}(D, T, I) ; >\!\!-\!\!<)$. Then the transitive closure, $>\!\!-\!\!<^*$, of $>\!\!-\!\!<$ is defined by $a >\!\!-\!\!<^* b$ if and only if

$(\exists n \in \mathbb{N})\ (\exists z_0, z_1, ..., z_n \in (\mathbf{C}(D, T, I) ; >\!\!-\!\!<))$ such that $a = z_0 >\!\!-\!\!< z_1 >\!\!-\!\!< z_2 .. >\!\!-\!\!< z_n = b$.

The transitive closure identifies a sequence of minimal refinements or enlargements by which to derive one concept from another. In particular, it can be applied to a query and a document, provided that we specify a concept node for the query and a concept node for the document. The former can be obtained by merging the query into the lattice as a pseudo-document. The latter is the lattice node with intent equal to the set of terms describing the document in question; such a node exists, by definition, and its extent contains at least that document. The transitive closure of a query can then be used to order the set of documents that can be derived from it.

Definition 7 Let q a user query, (D_q, T_q, I_q) the context (D, T, I) augmented with q, and $\mathbf{C}(D_q, T_q, I_q ; >\!\!-\!\!<)$ the set of concepts of the augmented context along with the nearest neighbor relation[3]. For each $d_1, d_2 \in D$, let $a_1, a_2 \in \mathbf{C}(D, T, I)$, such that the intents of a_1 and a_2 are equal to the descriptions of d_1 and d_2, respectively. Then d_1 is ranked ahead of d_2 with respect to q if and only if $n_1 < n_2$, where n_1 is the least $n \in \mathbb{N}$ such that $q >\!\!-\!\!<^* a_1$, and n_2 is the least $n \in \mathbb{N}$ such that $q >\!\!-\!\!<^* a_2$.

From the above definitions, it can be easily proved the following result.

Proposition 1 If a document $d_1 \in D$ is ranked ahead of a document $d_2 \in D$ for a user query q, according to Definition 7, then the set of terms contained in d_1 can be derived from the set of terms contained in q by a smaller number of admissible minimal transformations, with respect to the collection at hand, than the set of terms contained in d_2.

A complete ranked document list can be obtained by arranging the whole set of documents in the increasing order of minimal transformations that are necessary to derive each document from the query. Of course, this is a partially 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).

We now present a more elaborate example that illustrates the approach we have described. Then we address the issue of automatic determination of order-theoretical ranking.

---

[3] Note that both the user query and the documents are elements of $\mathbf{C}(D_q, T_q, I_q; >\!\!-\!\!<)$

4.2. An example

We will refer to a simple bibliographic database consisting of seven documents described by eight index terms (see Table 2). All documents are about computerized decision support systems, with two broad application domains: economics and environment. Suppose we are interested in the former class of documents, and let us assume that the database is queried by questioning "Neural-Network-Systems, Finance". The concept lattice built from the set of documents and from this query, treated as if it were a pseudo-document, is illustrated in Figure 3 by a Hasse diagram.

Table 2. A simple database containing seven documents described by eight index terms

	D1	D2	D3	D4	D5	D6	D7
Neural-Network-Systems	x	x	x				x
Knowledge-Based-Systems				x	x	x	x
Credit				x	x		x
Finance	x			x			x
Account	x		x				
Bank	x	x	x			x	
River		x					
Waters						x	

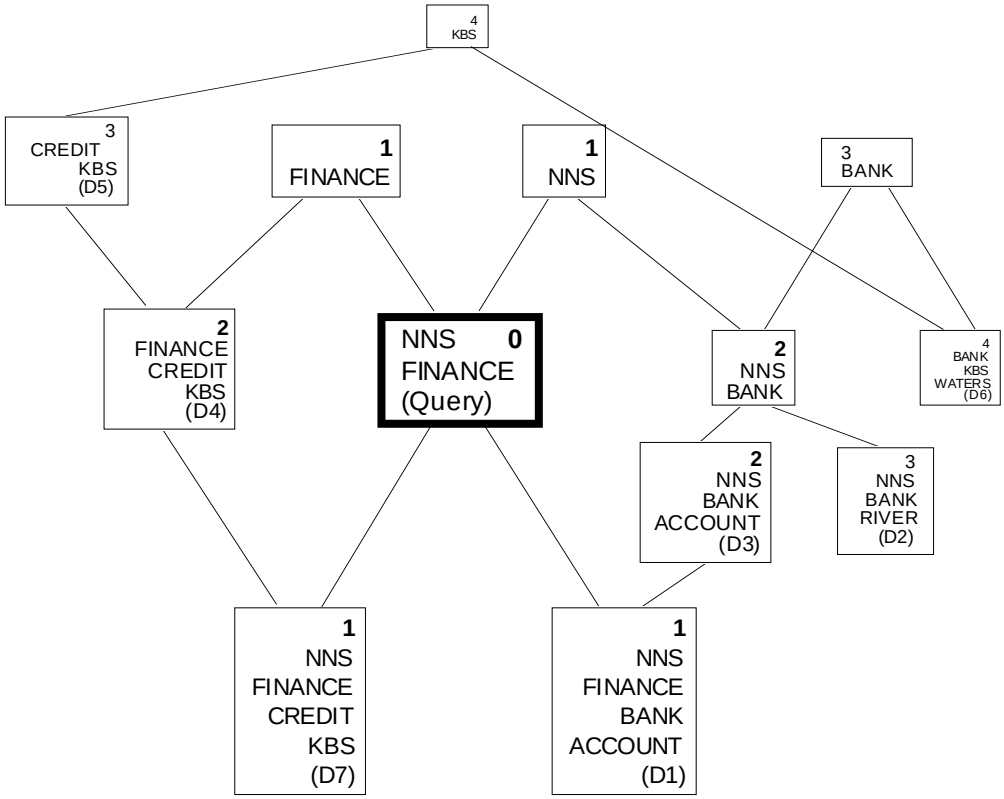

Figure 3. The concept lattice associated with the database in Table 2 and with the query “Neural-Network-Systems, Finance”. For each concept, it is also shown its ring number (right upper corner).

The Hasse diagram shows the set of concepts along with the nearest neighbor relation (see Definition 4), implying that there is an edge between two nodes if and only if they represent comparable concepts (the ascending paths can be seen as representing the subconcept/superconcept relation) and there is no other intermediate concept in the lattice. The labeling of the diagram in Figure 3 is such that the intents are shown for every concept, while an extent is attached only to the smallest concept that has that extent (i.e., when the description of documents in the extent coincides with the node's intent). It should also be noted that in Figure 3 we do not show the lattice top and the lattice bottom. By definition, the top concept contains all documents and is described by their common terms (if any), the bottom concept is described by the set of all terms and contains the documents that have those terms. Except for the case when the lattice top is described by a non-empty set of attributes (i.e., all documents have those attributes), or, dually, when the lattice bottom contains a non-empty set of documents (i.e., some document has all attributes), it is convenient to omit both the lattice top and the lattice bottom in the graph used to compute the document ranking, for these two nodes cannot be seen as proper queries with associated documents.

In order to visually highlight the transition from relevant to non-relevant information (to keep with our terminology, from near to distant rings), the diagram in Figure 3 has been depicted using a simple instantiation of the fisheye view technique (Sarkar and Brown, 1994). The focus of interest is the query node, and the other nodes are displayed in decreasing levels of detail (with respect to the font size) and at increasing graphical distance, depending on the topological distance from the focus, where the topological distance between a given node and the focus node is the length of the shortest path between the two nodes.

As stated in Definition 3, not every term subset is a lattice concept. For instance, in the lattice relative to the context in Table 1 there cannot be any concept having an intent equal to "Credit", in that all documents having "Credit" have also "Knowledge-Based-Systems". The completeness constraint limits the number of admissible concepts by favoring maximally specific descriptions of the extent's documents. In other terms, it is assumed that if one term always appears jointly with other terms, the single terms do not refer to distinct concepts while their tuple does convey a useful meaning. In a sense, every complete set of co-occurrences determines a semantic "context" specific to the collection at hand, and the closeness between two semantic "contexts" is completely determined by the distribution of terms in the collection's documents. One consequence of this assumption is that in the lattice there may well be two near concepts that differ by a larger number of terms than more distant concepts do. For instance, the query concept Q and the concept with extent D7 are connected by a single-link path although they differ by two terms (i.e., "Credit" and "Knowledge-Based-Systems"), because creating an intermediate concept between Q and D7 with either "Credit" or "Knowledge-Based-Systems" would be of no help in discriminating between the given set of documents. At the same time, it

may happen that the same two concepts become more distant, or, viceversa, closer, as the set of documents in the collection changes. For instance, as a consequence of the introduction of a new document described by “NNS-Finance-Credit”, a new concept with intent “NNS-Finance-Credit” would be added to the lattice between the concepts Q and D7, thus increasing their distance. Thus, this kind of distance is a context-sensitive measure, similar to most distance measures based on some clustering method. By contrast, this important feature cannot be easily incorporated into more traditional statistical measures where the distance between two representations is based only on the characteristics of those representations (e.g., Euclidean distance), although there have been some attempts in this direction using the standard deviations or the covariance matrix of the variable values (Everitt, 1993).

Now let us see how the ranked list for the given query and documents is produced. The nodes that are closest to the query (ring 1) are “Finance”, “NNS”, “NNS-Finance-Credit-KBS”, and “NNS-Finance-Bank-Account”. The relevant documents are D1 and D7 (recall that, according to Definition 6, in order to reach a document concept it is not sufficient to simply generate a node whose extent contains that document, but it is necessary to generate the node with intent equal to the document description). Ring 2 consists of the nodes “NNS-Bank”, “Finance-Credit-KBS”, and “NNS-Bank-Account”, which yield the documents D3 and D4, and so on. The complete ranked list of documents is the following (in parenthesis we indicate their distance from the query node): D1 (1), D7 (1), D3 (2), D4 (2), D2 (3), D5 (3), and D6 (4). This result is correct, in the sense that it seems to reflect the actual document relevance to the given query.

It is useful to compare the above ranking with that produced by BMR methods. We would get the following ranked list: D1 and D7, with the same score, followed by D2 and D3, with the same score, by D4, and by D5 and D6, with the same score. This result presents several inversions with respect to the ideal ordering, involving not only pairs of documents that did not match the query (D5-D6), but also pairs where only one document matched the query (D2-D5), or both documents matched the query (D2-D3). The last case is interesting. Documents D2 and D3 have the same term in common with the query (“NNS”) and both contain the ambiguous word “bank”. Using the lattice, the concept “NNS-Bank” can be equivalently specialized with “River” and with “Account” (see Figure 3); however, the concept “NNS-Bank-Account” may be reached from the given query through a shorter alternative path, which reflects the higher overall structural similarity of the latter concept with the query. Incidentally, this observation may suggest that it is worth investigating the potentials of concept lattices for dealing with ambiguous query terms in query refinement based on the actual content of the collection (Cooper and Byrd, 1997).

4.3. Computational efficiency

Order-theoretical ranking can be implemented as a three-step procedure. The concept lattice associated with a document collection, along with its Hasse diagram, is built first; then a query is mapped onto the lattice derived in the earlier step, and finally a ranked document list for that query is computed from the augmented lattice. In order to build the lattice we used GALOIS. The system GALOIS, described in detail in (Carpineto and Romano, 1996a), is based on an incremental algorithm for lattice construction that works by examining one document at a time and updating the Hasse diagram of the lattice relative to the documents that have already been seen. Once the document lattice has been built, the query is simply added to it, as if the query were a document, by GALOIS itself. In fact, using an incremental algorithm has the advantage that every query can be processed by adding it to the document lattice, without recomputing the augmented lattice from scratch whenever the system is presented with a new query. The third step is also computationally simple. It is sufficient to perform a breadth-first search through the augmented lattice starting from the query node, without generating the nodes that have already been encountered, until all nodes have been reached. Then, if there are some disconnected nodes left,[4] they are added at the end of the ranked list that has been produced so far. Each step of the breadth-first search returns a set of nodes that are equally distant from the query, with an associated set of documents. Thus, the position of each document in the output ranked list is computed in parallel while visiting the augmented concept lattice.

The overall complexity of order-theoretical ranking is essentially determined by the complexity of lattice construction, because the other two operations require at most one pass over the lattice built in the first step. The space complexity of the concept lattice built from a document collection varies, in practice, from linear to quadratic with respect to the number of documents, and the time complexity of each update performed by GALOIS varies in a similar manner because it is proportional to the number of concepts in the lattice being updated (Carpineto and Romano, 1996a). Thus, the relative efficiency of the process of lattice construction easily allows small/medium size applications of our approach, like the one that will be considered here, but it also raises the question of its applicability to large scale applications, which will be addressed below (see Section 6).

[4] Since we usually remove the lattice top and the lattice bottom, it may happen that some documents cannot be derived from the query by transitive closure, according to Definition 6. In practice, however, such high disconnected documents should be rare, and they are likely to be of no interest at all for the given query.

5. Comparative Evaluation

5.1. Goal

The objective of the experimental test was to evaluate the comparative effectiveness of the three ranking methods discussed in this paper, namely BMR (Best-Match Ranking), HCR (Hierarchical Clustering-based Ranking), and CLR (Concept Lattice-based Ranking). We ran two experiments. In the first, standard experiment we evaluated the retrieval effectiveness of the three systems on the entire set of documents contained in a collection. In addition, as in this paper we focused on the word mismatch problem, we evaluated the specific ability of the three systems to produce a relevance ordering for documents contained in the collection that did not match a given query.

5.2. Test Collections

We used two well known, electronically-available test collections in our experiment, CACM and CISI. The CACM data set contains 3204 documents and 52 queries, the CISI data set contains 1460 documents and 35 queries. In order to determine, for each collection, the document-term relation used as input by the three ranking method we used the classical blueprint for automatic indexing suggested by Salton (1989), consisting of four steps: text segmentation, word stemming, stop wording, word weighting. Text segmentation amounted to extracting the individual words occurring in the documents, ignoring punctuation and case. Word stemming was done by using a very large *trie*-structured morphological lexicon for English (Karp et al, 1992), that contains the standard inflections for nouns, verbs, and adjectives. For stop wording we used a stop list included in the CACM test collection, containing 428 common function words. Word weighting was performed through the classical *tf·idf* approach.

The last step, word weighting, is the most important and deserves some explanation. Strictly speaking, word weighting is necessary only for BMR, while HCR and CLR do not need this kind of information. In practice, however, word weighting may be useful also for the latter systems. This is the case in our experiment, where all ranking systems take as input a weighted term vector representation of the document collection. HCR uses the term weights to compute the matrix of document-to-document distances and to order the documents in each cluster; CLR uses the term weights to reduce the set of terms describing each document, mainly for reasons of efficiency, and to order the documents in each ring, similar to HCR. It follows from these assumptions that the weighting method itself becomes an independent variable of our experimental setting, because it can influence not only the performance of BMR, but also that of HCR and CLR. We will return to this issue when we discuss the results of our experiments.

The automatic indexing of queries was the same as documents, except for word weighting. CLR did not use weighting of query terms at all, while for BMR and HCR the weight of each term was simply given by its frequency in the query, which seems to produce better results.

5.3. Implementation of the three ranking systems

The implementation of BMR was straightforward: we computed the inner product with cosine normalization between the document vectors and a query vector, determined as explained above.

The implementation of HCR was more elaborate because several choices can be made at each of the main steps of this method, in particular during the process of cluster formation. To determine the degree of similarity between documents we used the inner product with cosine normalization. For the cluster formation stage we used the single link method, because although it may be less effective than the other main methods (complete link, group average, Ward's method),[5] it can be implemented more efficiently: $O(n^2)$ time and $O(n)$ space for the clustering of n documents**.** The issue of efficiency does make an important difference because the storage requirement of most hierarchical clustering methods - $O(n^2)$ space - may make their computation exceedingly hard even for small collections, like those considered in our experiments. To rank the set of generated clusters according to their relevance to a query, we computed the inner product with cosine normalization between the query and each cluster in the hierarchy, using as a representation of a cluster the set of terms describing all the documents contained in the cluster, weighted with their frequency (Griffiths *et al*., 1986). Finally, we decided to individually rank the documents in each cluster against each query because this strategy produces better results (Voorhees, 1985). This inner ranking was done using again the inner product with cosine normalization.

The CLR method was implemented as described in section 4.3, with two important specifications. First of all, like other retrieval systems that do not use full-text indexing due to computational limitations (e.g., Deerwester et al., 1990; Maarek et al. 1991), CLR works with a restricted set of index terms. The goal is to choose the most informative terms while facilitating the subsequent process of lattice construction. For this purpose, we selected, for each document, the first k terms (at most), where k was automatically set using the mean of the number of terms per document produced by full-text indexing. This method pruned the representation of the documents that contained more than k terms, and left those containing k

[5] Most experimental studies agree that the single link method may have worse retrieval performance than the other hierarchical clustering methods (e.g., Voorhees (1985), Willett (1988), Rasmussen (1992), Burgin (1995)). These results, however, have been partially contradicted by other experiments, in which more comparable levels of retrieval effectiveness were found (Griffiths *et al*, 1986, El-Hamdouchi and Willett, 1989).

terms or fewer unchanged. For each data set, the actual input representation and the concept lattice built from it had the following characteristics. For the CACM data set, which after word stemming and stop wording contained 9608 distinct terms and was described on average by 23.48 terms per document, the number of distinct terms remained the same while the average number of terms per document shrunk to 14.48. The corresponding lattice contained 40185 nodes. For the CISI data set, the number of distinct terms remained again the same (8862), while the average number of terms per document passed from 45.41 to 39.50; the concept lattices contained 256826 nodes. The time needed to construct the two lattices on a SUN Ultra 2 workstation equipped with 512 Mbytes of RAM was about fifteen minutes and two hours for the CACM and CISI data sets, respectively.

Of course, while restricting the index set by some heuristic thresholds allows more efficient lattice construction, it may also affect the subsequent retrieval process. We did not systematically study how retrieval performance varied as a function of indexing exhaustivity, but some preliminary experiments suggest that the optimal size should be smaller than full-text indexing and greater than the one we used in the comparative evaluation. This observation is consistent with earlier findings by Burgin (1995), who showed that the representation that optimizes the retrieval performance of hierarchical clustering-based systems usually represents a compromise between indexing exhaustivity and specificity. In our case, this behavior can be explained by considering that term pruning may help improve CLR's performance by filtering out noisy terms in the document description and by making document length more comparable, as long as it does not remove informative terms. We will return to this in Section 6.

The second important point to note in the implementation of the CLR method used in the experiment is that we ordered the documents contained in each ring. Similar to the implementation of HCR, the inner ranking was obtained by computing the inner product with cosine normalization between the documents and each query, although we should emphasize that alternative, more sophisticated methods are possible. For instance, we could discriminate between two nodes that are connected with the query through paths of the same length by using the number of distinct paths of that length that link the query to either node. Such an approach would represent a more natural refinement of our main criterion, because it would be based on a measure of the degree of multiple structural connections between the lattice concepts; however, this would be computationally hard, while taking the inner product is much more efficient.

5.4. Experiment 1: Ranking all documents

We ran each ranking system on each of the two collections for each query. To evaluate the retrieval effectiveness of the resulting ranked document lists, we used eight different evaluation measures. The results, averaged over the set of queries, are displayed in Table 3 and Table 4. If we look at the differences in performance between any two of the three methods, we get the following rough indications. HCR was clearly beaten by CLR as well as by BMR on both collections, while, of the two best methods, BMR was better than CLR on CACM and slightly worse on CISI.

	BMR	**HCR**	**CLR**
Average precision (non-interpolated)	0.320	0.231	0.253
11-point precision	0.340	0.257	0.281
Precision at 5 points	0.346	0.342	0.412
Precision at 10 points	0.304	0.298	0.240
Precision at 20 points	0.238	0.202	0.164
Recall at 5 points	0.227	0.136	0.228
Recall at 10 points	0.297	0.224	0.266
Recall at 20 points	0.428	0.323	0.319

Table 3. Comparison of retrieval performance for CACM.

	BMR	**HCR**	**CLR**
Average precision (non-interpolated)	0.164	0.127	0.162
11-point precision	0.183	0.153	0.185
Precision at 5 points	0.269	0.280	0.337
Precision at 10 points	0.266	0.254	0.286
Precision at 20 points	0.239	0.209	0.234
Recall at 5 points	0.027	0.042	0.043
Recall at 10 points	0.060	0.066	0.095
Recall at 20 points	0.107	0.103	0.139

Table 4. Comparison of retrieval performance for CISI.

A more precise picture can be obtained by examining the results relative to each of the eight selected performance measures, considering also how many of the differences were statistically significant according to a one-tailed paired *t* test with a confidence level in excess of 95%. This last datum is indicated in parenthesis below. BMR achieved better results than HCR on both CACM (eight wins (6)) and CISI (five wins (2), three losses (0)). Similarly, CLR fared better than HCR on CACM (five wins (2), three losses (0)), and on CISI (eight wins (2). Of the two best methods, BMR had better performance values than CLR on CACM (six wins (4), two losses (0)), and worse values on CISI (six losses (2), two wins (0)).

Even though the results relative to the comparison between CLR and BMR presented considerable variation across the two domains and the eight evaluation measures, a pattern seems to emerge from the data that is related to the type of performance variables measured in the experiment. If we consider the first two effectiveness measures, regarding the whole set of retrieved documents, CLR performed worse than BMR, which achieved much better results on CACM and comparable results on CISI. However, if we take the last six measures considered in the experiment, which focus on the system's retrieval effectiveness for the first retrieved documents, the results were quite different. Out of the overall 12 measurements of this kind, CLR was first 7 times, second 2 times, and last 3 times; by contrast, BMR was first 5 times, second 3 times, and last 4 times (HCR was never first, it was second 7 times, and third 5 times). In particular, CLR achieved better values of "precision at 5" and "recall at 5" than either opposing method on both data sets. The comparatively better results of CLR on the first retrieved documents can be explained by looking at the distribution of documents in the ordered set of rings returned by order-theoretical ranking.

We considered how the number of documents contained in a ring varied as the the ring's radius increased. It turned out that middle rings usually contained many more documents than nearest and farthest rings, somewhat similar to a normal distribution. In particular, the first rings typically contained a few documents, while the more numerous rings contained up to hundreds of documents. Thus, the system performance for the first retrieved documents was basically determined by the partially-ordered rank determined by the concept lattice, i.e., by the inclusion of documents in near or distant rings. As the system retrieved more documents, the rings grew much larger and the system performance became more and more influenced by the individual best-match ranking of the documents contained in each ring. Since CLR did not use full-text indexing, it is likely that CLR was less effective than BMR in ranking these distant documents, which resulted in an overall performance degradation.

An additional hidden parameter of our experimental design is the choice of the word weighting method, as mentioned in Section 5.2. In order to gain some insights into the effect of word weighting on performance, we performed some experiments with a different weighting scheme. We used the signal-noise ratio (*snr*), because it has a different theoretical basis than the

tf·idf approach and it is simple to implement (Salton and McGill, 83). Both efficiency and effectiveness were affected by the new weighting scheme. In particular, while the efficiency of HCR and BMR, which used full-text indexing, remained the same, we observed that the computation of CLR was adversely affected. Due to the specific distribution of *snr* weights (the same term has always the same weight in all documents), the pruning of the index set involved in CLR yielded a much smaller number of distinct terms and a smaller number of terms per document than the *tf·idf* approach, but, since there were more subsets of documents which had some terms in common, the final concept lattices computed using *snr* had a greater number of nodes.[6] As for retrieval effectiveness, the comparative results across ranking methods and domains were consistent with those obtained with *tf·idf*, while the absolute performance of *snr* was noticeably different from *tf·idf*. In particular, while the retrieval effectiveness of the two weighting functions was comparable on the CISI data set, the performance of *snr* was definitely inferior to that of *tf·idf* on the CACM data set, probably due to the different characteristics of the two collections (i.e., the CACM documents have more concise and better characterizing descriptions than the the CISI documents).

Finally, an interesting byproduct of our experiment concerns the comparative evaluation of HCR and BMR. Not only do these results confirm previous findings that HCR does not perform as well as BMR (Salton, 1971; Griffiths *et al*, 1986), but they also strongly suggest that the relatively good results of HCR reported by some authors (Jardine and van Rijsbergen, 1971; Croft, 1980) may be due to overly specific experimental choices, such as the use of a single small collection (i.e., Cranfield) and the emphasis on searches that retrieved only a few documents. A similar concern was also expressed by Willett (1988); we offered more evidence to support such a conjecture. In particular, we showed that HCR can be clearly outperformed by BMR when all documents are considered for evaluation, and that the performance of HCR, while remaining inferior to that of BMR, becomes more comparable to the latter as we consider for evaluation only the first retrieved documents.

5.5. Experiment 2: Ranking the non-matching documents

As one of our main claims is that order-theoretical ranking allows better treatment of documents that do not match the query than BMR and HCR, we are concerned with finding empirical evidence that supports this hypothesis. We are not aware of any earlier experiment of this kind, so we had to come up with our own test methodology, which is described in the following.

---

[6] Recall that the concept lattice contains, by definition, *all* nonempty and complete intersections between the documents of a context.

BMR, by its very nature, cannot rank documents that do not match a query, so we can assume that in this case the non-matching documents are randomly ordered. For HCR and CLR, we computed a ranking relative to the non-matching documents by removing the matching documents from the full document ranking. In this case, we get a strict partially-ordered output because the documents in a same ring cannot be further ranked by similarity with the query, as with matching documents. Then, in order to compare the performance of the three methods, we used Cooper's expected search length (ESL), first introduced in (Cooper, 1968), and then described in detail in (van Rijsbergen, 1979) and (Salton and McGill, 1983). We chose ESL because it very well suits partially-ordered retrieval output systems, like HCR and CLR, and, in addition, it allows sound comparison with random document ranking, which can be seen as the output of BMR methods for non-matching documents. In our experiment, ESL was defined as the average number of (non-matching) non-relevant documents that are retrieved by the system before all (non-matching) relevant documents are retrieved. To cope with the random element of this measure, it is convenient to assume that in a set of equal scored documents the relevant ones are located at equal intervals; it is then useful to compute a relative measure of ESL (ESL-reduction), which specifies the improvement obtained by the case in which the non-matching documents are partially ordered over the random case (Salton and McGill, 1983).[7] We computed ESL-reduction for each ranking method (HCR and CLR) and for each query on each collection, and then averaged the results over the set of queries. The results are shown in Table 5**.**

The CLR and HCR methods performed, on average, much better than BMR (random selection) in both domains, although we noted that for some queries HCR resulted in a small negative ESL-reduction over the random case. For the CACM data set, CLR performed better than BMR by a much greater margin than HCR did, while, on the CISI collection, CLR and BMR achieved more comparable improvements (over the random case). On the whole, the results of this experiment suggest that, for non-matching documents, both HCR and CLR may be much better than random ranking, and that CLR may be more effective than HCR. These indications received further support when, analogously to the earlier experiment involving all documents, we ran Experiment 2 using *snr* instead of *tf·idf* as weighting scheme. In particular, we found that CLR fared better than HCR not only on the CACM but also on the CISI data set, with the performance of HCR on the latter data set coming closer to random selection.

We should emphasize that the superior performance of HCR and CLR for non-matching documents had a limited effect on the performance relative to the whole set of documents, due

---

[7] *ESL-reduction* = [*random-ESL* - *ESL*] / *random-ESL* = 1 - {[PREVNONREL + (NONREL • REL) / REL+1]} / [(ALLREL • ALLNONREL) / (ALLREL + 1)], where PREVNONREL is the number of nonrelevant documents in all sets preceding the one where the search terminates, REL is the number of relevant documents in the final set, and NONREL is the number of nonrelevant documents in that set.

to the characteristics of the test collections. The CISI collection has, on average, 49.77 relevant documents per query. Of these, only 4.49 do not match the query (9.02%), while the total number of non-matching documents is 598.54. For the CACM collection, there are, on average, 15.31 relevant documents, of which 1.40 (8.86%) do not match the query; the total number of non-matching documents is 2104.25. The proportion of relevant documents that do not match the query is therefore low; the situation is similar for both collections, although the fact that the CACM documents contain, on average, a much smaller number of terms, should make this event more unlikely. In fact, many classical test collections seem biased towards producing relevant documents that do match the query. While this bias may be representative, in the sense that it reflects the way natural language is used, we regret that test collections containing a significant proportion of non-matching relevant documents are extremely rare, because this latter situation would probably better reflect the characteristics and the limitations of many real searches conducted with short user queries, as is the case for Web-based information retrieval. Furthermore, even for more typical information retrieval evaluation tasks involving longer queries, such as the TREC test collection, it is sometimes the case that a large proportion of relevant documents do not match the query terms (see for instance Berenci *et al*. (1999)).

	CACM	CISI
HCR	11%	16%
CLR	27%	13%

Table 5. Average ESL-reduction for non-matching documents.

6. Scaling issues

As in the experiments we used databases of limited size, it is useful to examine how well our approach could scale up. The two main aspects involved here are efficiency and effectiveness. We will consider both of them, in turn.

As mentioned in Section 4.3, the size of the lattice may grow quadratically with respect to the number of documents, but there is experimental and theoretical evidence that, when the number of index terms used to represent each document is small, both the size of the lattice and the time necessary to compute each update by Galois grow linearly with respect to the number of documents (Carpineto & Romano, 1996a). Reducing the set of index terms may be therefore an effective strategy to deal with larger databases than used in our experiments, although this may not be sufficient for large scale databases. For the case when it is exceedingly hard to build

the concept lattice associated with a database, we could use a method that finds an approximated solution.

The idea is to let the user formulate a query first, and then dynamically compute a limited portion of the lattice, centered around the concept that corresponds to the query. This approach has been recently explored by Carpineto and Romano (1998) to support query reformulation in Boolean retrieval. The core of this method is an algorithm that finds the neighbors (parents and children) of a given node, with a time complexity proportional to the number of documents and to the average number of neighbors in the lattice. If the collection contains a few thousands of documents, the neighbors of a given node can be generated in real time even with full-text indexing (Carpineto and Romano (1998)). Furthermore, the dynamic algorithm can be recursively applied to each generated node to find more distant nodes. This procedure may therefore be more efficient than full lattice construction to find the rings around the query, provided that it is not necessary to rank all documents but only the most relevant ones and that to find such documents it is sufficient to generate a limited number of rings.

We can make these considerations more concrete by providing a rough estimate of how much of the lattice would have to be built to achieve certain retrieval results. By analyzing the data collected in our experiments, we observed that the first two lattice rings generated for each query contained on average the 13% of the relevant documents associated with the query. Since the number of neighbors of each node is usually proportional to the size of the node's intent, irrespective of the number of documents in the collection, this implies that for a query with q terms, in order to retrieve a small but not negligible fraction of the relevant documents, it may be sufficient to generate only $1 + q + q^2$ nodes, which will require $1 + q$ invocations of the dynamic algorithm. For short queries and collections of medium size, such an approach would still be feasible in real time.

Whichever construction method we use, for large data sets it might be necessary to choose a set of index terms of limited size. At this point, it is important to consider whether working with a smaller set of index terms than used in our experimental evaluation might affect retrieval effectiveness. In order to gain some insights into this phenomenon, we performed some experiments varying the threshold used to select the set of index terms that describe each document. We observed that working with a smaller set of index terms did not always result in performance degradation, although this was typically the case, and, perhaps more importantly, we noticed that the retrieval effectiveness usually underwent small decreases in response to major reductions in the size of the index set. For the CACM database, for instance, when using a significantly more restrictive selection criterion than the average number of terms (i.e., the mean of weights in the document), the average number of terms per document reduced to 8.47 (- 42%), the number of nodes in the lattice to 13301 (- 67%), and the average precision decreased "only" to 0.23 (- 8%). This is encouraging, of course, but it is not enough to

guarantee that the retrieval performance of the system would be equally good when shifting to a larger environment. Working with a large number of documents and with a comparatively small lattice would probably increase the number of documents associated with each ring, thus reducing the utility of lattice-based ranking. Also, due to computational limitations, the system might be unable to correctly rank the documents that did not match the query, which may be located in the farthest rings. As a consequence, the overall retrieval performance might still be badly affected.

Admittedly, while the above considerations may be useful to expand the range of practical applications of our system, they do not solve the scalability problem all. For large domains, an alternative, and perhaps more promising, approach is to try to exploit the potentials of combined ranking methods. This will be discussed in the final section of the paper.

7. Related work

7.1. Hierarchical clustering-based ranking

Since a concept lattice can be seen as a particular structure for clustering documents and terms, it is important to further compare the relative merits and drawbacks of HCR and CLR. One distinguishing feature, as remarked in the first part of the paper, is that CLR, unlike HCR, has a clear semantics by which to characterize the ranked document list produced in response to a query in terms of the query and documents descriptions. A related important characteristic for a cluster-based ranking system is the ability to integrate cluster formation and cluster selection: HCR needs to map different representations and similarity functions, while CLR uses a unique framework for representing documents and query and comparing them. Also, the clustering structure used by HCR has a smaller number of free parameters (n clusters for n documents) than CLR, whose number of clusters is usually in the range between n and n^2. From the point of view of retrieval effectiveness, these seem to be clear advantages of CLR over HCR, and the results of our experiments support this claim.

For the worst case computational complexity, CLR compares less favorably, because there are some cases in which the size of the lattice may grow exponentially with respect to the number of documents. In practice, however, the efficiency of CLR is comparable to and sometimes better than that of most HCR methods, whose time and storage requirements are at least $O(n^2)$, except for the single link method, in which the latter complexity may reduce to $O(n)$.[8] Furthermore, the concept lattice method has the advantage that its complexity can be

[8] As mentioned in Section 4.3, the size of the lattice may not, and usually does not, grow quadratically with respect to the number of documents. In our experiments, for instance, the concept lattice associated with the

controlled by reducing the number of terms used to represent each document. This does not hold, in general, for the hierarchical clustering methods, although some inverted file algorithms for the calculation of interdocument similarity coefficients have been developed that can take advantage of short document descriptions (Croft, 1977; Willett, 1981). Other variables of interest in the comparison concern incrementality and use of background knowledge (e.g., thesauri). In short, both types of clustering structures can be built incrementally (Carpineto and Romano, 1996a; Can, 1993), while CLR can more easily accommodate background knowledge than HCR because the clusters in the lattice have an intensional description which makes it possible to use broader terms to index more general clusters (Carpineto and Romano, 1996a).

7.2. Using concept lattices for information retrieval

The application of concept lattices to information retrieval tasks is not new, but it has usually focused on interactive searches. The typical scenario (Godin *et al*., 1989 and 1993) starts with a user query, followed by some form of visualization of the lattice region on which the query zoomed in; the user may then see the documents contained in each of the nodes displayed on the screen or may jump to more distant regions by formulating new queries. This approach has been subsequently endowed with the possibility for the user to dynamically restrict the lattice being searched using a set of terms that should, or should not, be contained in the documents of interest (Carpineto and Romano, 1996b). More recently, the basic navigational framework has been further extended to accommodate for Boolean queries and full-text indexing (Carpineto and Romano, 1998). In all these approaches, the concept lattice associated with a collection of documents is seen as a search space that can be explored by a user using various and integrated retrieval strategies. To our knowledge, this paper is the first attempt to explore the potentials of concept lattices for automatic document ranking.

7.3. Other approaches based on interdocument similarity

The notion of query and document space has also been used by non-clustering approaches to document retrieval. Everett and Cater (1992) and Egghe and Rousseau (1998) suggested that a similarity function between a document and a query, along with a threshold that specifies the set of retrieved documents, can be seen as a topology on the document space, which implicitly determines neighborhoods around every document even without the formulation of a specific query. This resembles our approach, but the nature and the scope of topological retrieval systems are very different from CLR. Since a topological structure on the document space is

---

CISI data set (1460 documents) contained 256826 nodes, while the concept lattice associated with the the CACM data set (3204 documents) contained only 40185 nodes.

defined with respect to a particular (best-match) retrieval method, the topological approach may be useful to discover general properties and to investigate the behavior of the given retrieval method but it does not produce *per se* a set of documents in response to a user query. In fact, topological retrieval cannot be implemented as a truly document ranking system.

A proper ranking system that uses an alternative method to discover and exploit hidden similarities between query and documents was presented by Deerwester et al. (1990). Instead of computing some form of neighborhood in a structured document space, it changes the representation of documents and query first, and then it matches the query against individual documents using the newly-created representations. These representations are based on the singular value decomposition (related to factor analysis) of a term-document matrix from the entire document collection, which was termed latent semantic indexing (LSI). Besides the ranking strategy and the mathematical underlying tool, another main difference between LSI and CLR is the comprehensibility of the intermediate text structures generated by the two approaches. Unlike the lattice concepts, the factors extracted by LSI are difficult to interpret and to relate to the actual document description, although there has been some attempt in this direction (Story, 1996).

Although motivated by a different goal - i.e., extending the basic vector space model with term-term correlations - the work by Wong *et al*. (1987) can be seen as an earlier attempt at transforming document representation prior to ranking, based on interdocument similarity. In Wong *et al*.'s Generalized Vector Space Model (GVSM), the terms describing the documents are in turn described as a combination of atomic concepts (or minterms) associated with a Boolean algebra defined over the set of terms. In practice, the idea is to broaden the document representation by adding terms that are correlated with the original terms which described the documents. For instance, using the same example given by Wong *et al*. (1987), suppose that document D is described by terms T1 and T3, and that there are many documents in our collection that contain exactly T1 and T2. In this case, D's description is extended to include both the minterms T1-T3 and T1-T2, thus favoring the retrieval of the document D in response to the query T2. This kind of behavior has a natural counter-part in the CLR method, with the two concepts T1-T3 and T1-T2 being placed close to one another and the document described by T1-T3 being retrieved right after those described by T1-T2. The GSVM has not been widely used since its publication probably due to its computational limitations, although Wong *et al*. (1987) suggested also model approximations.

8. Conclusions and future work

We took the view that the ranked relevance of a set of documents to a query can be computed in terms of the length of the sequence of minimal refinements/enlargements that transform the query into each document, as determined by the concept lattice associated with the documents and the query. Concept lattice-based ranking (CLR) represents a major departure from other approaches based on document similarity such as hierarchical clustering-based ranking (HCR), because it does not rely on a similarity measure to build the cluster hierarchy and it does not involve scoring a query against individual document clusters to rank the documents in the cluster hierarchy. We have seen that CLR's firmer theoretical basis help overcome some inherent limitations of HCR due to fundamental assumptions and operational implementations.

We compared the retrieval effectiveness of CLR with that of HCR and best-match ranking (BMR). The results clearly showed that HCR was outperformed by both CLR and BMR. Of the two best methods, BMR achieved better performance than CLR on the whole document set, while CLR compared more favorably than BMR on the first retrieved documents, due to a finer granularity of their grouping by CLR, as well as on the documents that did not match the query, where the superiority of CLR's partially-ordered rank over BMR's random selection was apparent.

This research can be extended in several directions. Using concept lattices, the distinction between textual information and information expressed as attribute-value pairs is blurred. With some precautions and devices (Wille, 1992; Carpineto and Romano, 1996a), it is possible to build a concept lattice from a set of structured documents, where each document may be characterized by a set of attributes that may contain free-text descriptions or take on nominal or numeric values. One direction for future work is to explore the potentials of concept lattices for the retrieval and management of this class of enriched documents, with a view to XML documents.

Another important issue is a more fully utilization of term weights in the process that leads to the final ranked document list. At the moment, term weights are used to restrict the set of index terms used to describe the documents and to order the documents contained in each ring, but not to assign rings to documents. The use of term weights in the determination of the partial ordering might better reflect the relative importance of terms in each document and it might also explicitly account for document length when documents are not the same length. In order to augment order-theoretical ranking with term weights, we plan to investigate the use of an extension of the concept lattice theory, recently presented by Wille (1995), that combines objects, attributes, and conditions under which objects may have certain attributes.

A third avenue for further research concerns the combination of CLR and BMR, which seems to complement each other very well. BMR is more efficient and may have better effectiveness when considering the whole set of documents, CLR may be more effective for the first retrieved documents and it can discriminate between non-matching documents. A combined strategy might keep the strengths of the two methods while avoiding their main weaknesses. One simple integration strategy is to have BMR rank an entire collection first, and then to use CLR to refine the ranking of the best-matching documents returned by BMR. The anticipated advantages of such an integrated approach are that it might be applied to large scale collections and it might feature a better retrieval performance than BMR on the first retrieved documents, which are of greater interest in many practical applications. That this is indeed a promising research direction is indirectly confirmed by some recent work on the application of clustering techniques to ranked documents, which showed that a combined strategy may produce good results (Hearst and Pedersen, 1996), even when the clusters are created from short document descriptions such as the snippets returned by Web search engines (Zamir and Etzioni, 1998).

Acknowledgements

We would like to thank two anonymous reviewers for many useful comments and suggestions. This work was carried out under the framework of the agreement between Telecom Italia and the Fondazione Ugo Bordoni.

References

Berenci, E., Carpineto, R., Giannini, V., & Mizzaro, S. (1999). Effectiveness of keyword-based display and selection of retrieval results for interactive searches. To appear in Proceedings of the Third European Conference on Digital Libraries (ECDL'99).

Burgin, R. (1995). The retrieval effectiveness of five clustering algorithms as a function of indexing exhaustivity. Journal of the American Society for Information Science, 46(8), 562-572.

Callan, J. P., Croft, W.B., & Harding, S.M. (1992). The INQUERY retrieval system. Proceedings of the Third International Conference on Databases and Expert Systems Applications (pp. 78-83), Springer-Verlag.

Can, F. (1993). Incremental clustering for dynamic information processing. ACM Transaction on Information Systems, 11(2), 143-164.

Carpineto, C., & Romano, G. (1996a). A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval. Machine Learning, 24, 1-28.

Carpineto, C., & Romano, G. (1996b). Information retrieval through hybrid navigation of lattice representations. International Journal of Human-Computer Studies, 45, 553-578.

Carpineto, C., & Romano, G. (1998). Effective reformulation of Boolean queries with concept lattices. Proceedings of the Third International Conference on Flexible Query-Answering Systems (FQAS'98) (pp. 83-94), Springer Verlag.

Carpineto, C, De Mori, R., & Romano, G. (1999). Informative term selection for automatic query expansion. Proceedings of the Seventh Text Retrieval Conference (TREC-7) (pp. 363-369), NIST Special Publication 500-242.

Cooper, W. S. (1968). Expected search length: a single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. American Documentation, 19(1), 30-41.

Cooper, J., & Byrd, R. (1997). Lexical navigation: visually prompted query expansion and refinement. Proceedings of the Second ACM Digital Library Conference (pp. 237-246), Philadelphia, ACM.

Croft, W. B. (1977). Clustering large files of documents using the single-link method. Journal of the American Society for Information Sciences, 28, 341-344.

Croft, W. B., & Harper, D. J. (1979). Using probabilistic models of document retrieval without relevance information. Documentation, 35(4), 285-295.

Croft, W. B. (1980). A model of cluster searching based on classification. Information Systems, 5, 189-195.

Davey, B., & Priestley, H. (1990). Introduction to lattices and order. Cambridge, Great Britain: Cambridge University Press.

Deerwester, S., Dumais, S. T., Furnas, W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Sciences, 41(6), 391-407.

Defays, D. (1977). An efficient algorithm for a complete link method. Computer Journal, 20, 364-366.

Egghe, L. & Rousseau, R. (1998). Topological aspects of information retrieval. Journal of the American Society for Information Science, 49(13), 1144-1160.

El-Hamdouchi, A., & Willett, P. (1989). Comparison of hierarchic agglomerative clustering methods for document retrieval. The Computer Journal, 32, 220-227.

Everett, D. M., & Cater, S. C. (1992). Topology of document retrieval systems. Journal of the American Society for Information Science, 43, 658-673.

Everitt, B. S. (1993). Cluster analysis (third edition). London: Arnold.

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1987). The vocabulary problem in human-system communication. Communications of the ACM, 30(11), 964-971.

Godin, R., Gecsei, J., & Pichet, C. (1989). Design of a browsing interface for information retrieval. Proceedings of the 12th International Conference on Research and Development in Information Retrieval (ACM SIGIR'89), (pp. 32-39). Cambridge, MA, ACM.

Godin, R., Missaoui, R., April, A. (1993). Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. International Journal of Man-machine Studies, 38, 747-767.

Griffiths, A., Luckhurst, H. C., & Willett, P. (1986). Using interdocument similarity information in document retrieval systems. Journal of the American Society for Information Sciences, 37, 3-11, 1986. Also in K. Sparck Jones, & P. Willett (Eds.), Readings in Information Retrieval (pp.365-373), Morgan Kaufmann, 1997.

Harman, D. (1992). Relevance feedback and other query modification techniques. In W.B. Frakes & R. Baeza-Yates (Eds.), Information Retrieval - Data Structures and Algorithms, pp. 241-263, Englewood Cliffs: Prentice Hall.

Hearst, M. A., & Pedersen, J. O. (1996). Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. Proceedings of the 19th International Conference on Research and Development in Information Retrieval (ACM SIGIR'96) (pp. 76-84), Zurich, ACM.

Jardine, N., & van Rijsbergen, C. J. (1971). The use of hierarchical clustering in information retrieval. Information Storage and Retrieval, 7, 217-240.

Karp, D., Schabes, Y., Zaidel, M., & Egedi, D. (1992). A Freely Available Wide Coverage Morphological Analyzer for English. Proceedings of the 14th International Conference on Computational Linguistics (COLING '92), Nantes, France.

Rasmussen, E. (1992). Clustering algorithms. In W.B. Frakes & R. Baeza-Yates (Eds.), Information Retrieval - Data Structures and Algorithms (pp. 419-442), Prentice Hall.

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to 2-Poisson method for probabilistic weighted retrieval. Proceedings of the 17th International Conference on Research and Development in Information Retrieval (ACM SIGIR'94) (pp. 311-317), Dublin, ACM.

Salton, G. (Ed.) (1971). The SMART retrieval system - Experiments in automatic document retrieval. Englewood Cliff: Prentice Hall.

Salton, G., & McGill, M. (1983). Introduction to Modern Information Retrieval. New York: McGraw Hill.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513-523.

Salton, G. (1989). Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer. Addison Wesley.

Sarkar, M, & Brown, M. (1994). Graphical Fisheye Views. Communications of the ACM, 37, 12, 73-84.

Shaw Jr, W., Burgin, R., & Howell, P. (1997). Performance standards and evaluations in IR test collections: cluster-based retrieval models. Information Processing & Management, 33(1), 1-14.

Singhal, A., Salton, G., Mitra, M., & Buckley, C. (1995). Document length normalization. Technical report TR95-1529, Cornell University.

Spink, A., & Saracevic, T. (1997). Interaction in information retrieval: selection and effectiveness of search terms. Journal of the American Society for Information Sciences, 48(8), 741-761.

Story, R. E. (1996). An explanation of the effectiveness of latent semantic indexing by means of a bayesian regression model. Information Processing & Management, 32(3), 329-344.

Turtle, H.,& Croft, B. (1991). Evaluation of an Inference Network-Based Retrieval Model. ACM Transactions on Information Systems, 9(3), 187-222.

van Rijsbergen, C. J. (1971). An algorithm for information structuring and retrieval. Computer Journal, 14, 407-412.

van Rijsbergen, C. J., & Croft, B. (1975). Document clustering: an evaluation of some experiments with the Cranfield 1400 collection. Information Processing & Management, 11, 171-182.

van Rijsbergen (1979). Information retrieval. London: Butterworths.

Voorhees, E. M. (1993). Using WordNet to disambiguate word senses for text retrieval. Proceedings of the 16th International Conference on Research and Development in Information Retrieval (ACM SIGIR'93) (pp. 171-180), Pittsburgh, ACM.

Voorhees, E. M. (1985). The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval. PhD thesis, Cornell University, Ithaca, NY.

Voorhees, E. M., & Harman, D. (1998). Overview of the sixth text retrieval conference (TREC-6), Proceedings of the the Sixth Text REtrieval conference (TREC-6), NIST Special Publication.

Walker, S., Robertson, S. E., Boughanem, M., Jones, G.J.F., & Sparck Jones, K. (1997) OKAPI at TREC-6. Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication.

Wille, R. (1984). Line diagrams of hierarchical concept systems. International Classification, 2, 77-86.

Wille, R. (1992). Concept lattice and conceptual knowledge systems. Computer & Mathematics with Applications, 23(6-9), 493-515.

Wille, R. (1995). The basic theorem of triadic concept analysis. Order 12, 149-158.

Willett, P. (1981). A fast procedure for the calculation of similarity coefficients in automatic classification. Information Processing & Management, 17, 53-60.

Willett, P. (1988). Recent Trends in Hierarchic Document Clustering: a Critical Review. Information Processing & Management, 24(5), 577-597.

Xu, J., & Croft, B. (1996). Query expansion using local and global document analysis. Proceedings of the 19th International Conference on Research and Development in Information Retrieval (ACM SIGIR'96) (pp. 4-11), Zurich, ACM.

Wong, S. K. M., Ziarko, W., Raghavan, V. V., & Wong, P. C. N. (1987). On modeling of information retrieval concepts in vector spaces. ACM Transactions on Database Systems, 12(2), 299-321.

Zamir, O., & Etzioni, O. (1998). Web document clustering: a feasibility demonstration. Proceedings of the 21st International Conference on Research and Development in Information Retrieval (ACM SIGIR'98) (pp. 46-54), Melbourne, ACM.