Single-Focus Broadening Navigation in Concept Lattices

Gillian J. Greene and Bernd Fischer

CAIR, CSIR Meraka
Computer Science Division
Stellenbosch University, South Africa
{ggreene, bfischer}@cs.sun.ac.za

Abstract. Formal concept analysis has been used to support information retrieval tasks in many domains, in particular the traditional "by keyword" document search with a conjunctive query interpretation. However, support for exploratory search or *browsing* needs new navigation algorithms that allow users (*i*) to continuously update the current query and (*ii*) to broaden as well as refine the result set. In this paper we investigate a step-wise navigation algorithm that supports both broadening and refinement operations. Our navigation operations maintain some useful algebraic properties. We motivate our approach on a dataset of wine reviews, which contains different facets of information.

Keywords: Information Retrieval, exploratory search, step-wise navigation, broadening navigation, Formal Concept Analysis

1 Introduction

Formal concept analysis has been used to support information retrieval (IR) [1, 2] tasks in many domains [3, 4] and to implement different IR algorithms. The traditional approach [5] to IR using formal concept analysis views the documents as objects and their associated meta-data and extracted terms as attributes. The concept lattice is then computed from these document contexts. Each concept's intent represents a possible query (interpreted as the conjunction of all corresponding terms), with the extent forming the set of retrieved documents [5].

One particular IR task, and the one that we are interested in, is exploratory search [6] or *browsing* [7, 8]. It is aimed at familiarizing the user with the underlying data through serendipitous navigation, and so complements the traditional, direct keyword lookup-based document retrieval. Browsing is supported in graph structures by moving from vertex to vertex where each vertex represents the current query [7]. Therefore, in order to implement browsing with concept lattices, we need a step-wise navigation algorithm that allows users (*i*) to incrementally update the current query and (*ii*) to restrict (i.e., move down in the lattice) as well as broaden (i.e., move up in the lattice) the result set. In this paper we focus on such step-wise navigation algorithms and in particular a broadening navigation approach.

Concept lattices can in principle be navigated directly, by following the subconcept relation to move from one concept to another concept in its direct neighborhood [7]. However, this only allows for small navigation steps and thus restricts the serendipitous nature of the browsing operation and becomes impractical for large lattices. Instead, we aim at large step navigation algorithms that allow users to select and deselect arbitrary attributes and rely on the meet and join operations to move between concepts [9].

Large-step navigation algorithms should ideally satisfy a number of properties that ensure that their behavior is transparent to users. First, they should be *Markovian*, i.e., rely only on the current query concept and the new selection (or de-selection) to determine the next concept as result of the navigation step. This means that users do not need to remember the navigation history in order to understand the results. Second, they should be *Abelian*, i.e., the order of the navigation steps should have no effect on the next navigation result. This allows users a certain degree of freedom in how they navigate through the underlying document collection. Finally, they should have the *single focus* property, i.e., each query result can be represented by a single concept in the lattice. If the concept lattices constructed from the document contexts were Boolean lattices then these properties would follow automatically; however, this is not the case for most document collections.

If we follow a purely conjunctive query interpretation (i.e., consider all query terms to be connected by the AND operator), we can use the lattice's meet operation as implementation of the AND operator [10] in a document-term concept lattice. Moreover, the navigation algorithm is then by construction Markovian and Abelian, and has the single focus property. However, this does not provide us with disjunctive queries, or, with any broadening navigation operations. We therefore investigate broadening navigation approaches that maintain only a single focus concept.

It remains unclear what exactly constitutes broadening navigation, and there are several different operations that extend the query result and can be considered as "broadening".

- We can de-select a previously selected term; under a purely conjunctive query interpretation the new focus is then computed as the meet of the introducing concepts of the remaining terms (although Lindig [11] has described an optimized implementation). Note that the new focus has not necessarily been visited during the previous navigation steps (so de-selection is not always an undo operation), but it is a super-concept of the old focus, and conjunctive navigation with selection and de-selection is still Markovian and Abelian.
- We can use a separate concept to represent each argument of an OR operator; the result of such a disjunctive query is then the union of all corresponding extents [12]. However, this disjunctive navigation gives up the single focus property and is no longer Abelian, since the order of AND and OR operators matters.
- We can also retrieve or insert a *query concept* [13] into the lattice, where the *query concept's* intent contains the current search terms and retrieve the

	{country}		{review text}		{varietal}		
	USA	South Africa	Fruity	Berry	Cabernet Sauvignon	Merlot	Pinotage
wine-bottle1	X		X		X		
wine-bottle2		X	X	X	X		
wine-bottle3	X			X		X	
wine-bottle4		X		X			X

Fig. 1: Example context derived from wine review data, wine bottles are objects with review text, review year, vintage, location and winery as attributes. Attribute facets are indicated by the color of the column.

parents of the query concept (called the *query generator*), as a generalization [14]. Additionally, more children of the query generator can be included to broaden the results further. These are referred to as *cousin concepts* [14].

– We can use the lattice's join operation as a generalization operation; if the generalized concepts are determined by objects (rather than attributes) this is also known as *object-based navigation* [15]. This navigation has the single focus property by construction, and is still Markovian and Abelian (when it is not mixed with refinement operations, otherwise lattice distributivity is also required to ensure the Abelian property), but does not implement the Boolean OR operation: due to the closure operations in the lattice construction, the extent of the new focus typically contains additional objects. This could be seen as a feature [15] but in contexts where the attributes represent different categories or *facets* [16] this is prone to *overgeneralization*. Overgeneralization refers to the focus moving too high in the lattice (possibly to top) which would result in a decrease of precision for the query's results. In particular, if we have functional facets (where each object can have only a single attribute for a given category, such as year of birth), the join will effectively cancel the selected attributes from this category. The join operation is thus unsuited as an intuitive generalization operation. We therefore investigate an alternative generalization operation that makes use of only subsets of the extents of the attribute concepts of the selected items, in order to provide a more intuitive broadening navigation.

Our approach is motivated from navigation in a dataset of wine reviews extracted from [17]. The full dataset contains over 16000 objects. However, we use a small example of the dataset in order to make the drawing of the concept lattices feasible. For each wine bottle we have as attributes, the winery, the vintage, the reviewer, the review year as well as the location and keywords extracted from the reviews. We use individual wine bottles as objects in the context and assign all other fields as the attributes. Figure 1 provides an example of the constructed context for this dataset. This dataset contains functional facets, such as the country, where each wine bottle can originate from only one country.

In this paper we provide a brief overview of information retrieval tasks and navigation in concept lattices (Section 2). We then illustrate our refinement selection and de-selection approach (Section 3) where the de-selection operation reverses a refinement selection. We define a single-focus boolean OR navigation operator, followed in Section 5 by an intuitive generalization operation which prevents the full object set in the lattice from being returned. Additionally in Section 5 we discuss an approach for finding similar objects within the concept lattice.

2 Information Retrieval and Navigation using Concept Lattices

There have been many approaches to supporting information retrieval tasks using concept lattices, some of which extend to disjunctive queries and broadening approaches.

Codocedo et al. [14] propose an information retrieval approach using concept lattices where queries are answered using the *cousin concepts* of the *query concept*. The *query concept* is inserted into (or identified in) the concept lattice with a placeholder object and all the attributes that form a part of the current query [13]. The superconcept of the query concept is then referred to as the *query generator*. The *cousin concepts* of the query concept refer to the subconcepts of the *query generator*. The cousin concepts and the query generator are used to implement a broadening approach in the concept lattice [14]. The query's result is then returned as the union of the cousin concepts' extents.

Ferré [18] uses a navigation technique where a generalization is similar to our de-selection (it does not need to take place in any particular order) and de-selection refers only to removing the last selected item (e.g., an undo operation).

Godin et al. [8] described an iterative retrieval algorithm which maintains a *focus* concept whose extent is the retrieval result. Initially, the focus is the lattice's top element; in each iteration the user moves it to an adjacent concept, by adding (removing) an attribute (not) in the intent of a concept directly above (below) the current focus. However, this navigation style is too incremental, because the focus can move only one level at a time, and too constrained, because the user can only choose attributes from the intents of the directly adjacent concepts, and has no indication which choices are hidden behind paths not taken.

Lindig [9] introduced a semi-constrained navigation algorithm where, the focus can be refined by selecting *any* attribute from *any* concept (except $\perp$) below the focus, provided the attribute is not already in the focus' intent. The focus is then updated by computing its meet with the attribute concept. A restriction on selectable attributes prevents navigation into dead ends, and ensures that each query refinement also refines the query results.

Fischer [15] exploited the duality of concept lattices and introduced object-based navigation; here, selection of an object not in the focus' extent is a widening step that is implemented via the join.

Lindig [11] uses object-based navigation to implement *relevance feedback*; selecting an object in the focus' extent selects all attributes in the intent of its object concept.

3 Refinement Selection and Deselection

Refinement operations in the lattice can be computed using the meet operation. For step-wise navigation, we maintain a current focus concept at each navigation step. The focus can be refined with a new selection by calculating the meet of the current focus and the attribute concept of the new selection.

Additionally, items available for selection can be restricted to those that have a non-bottom meet with the focus, ensuring that a selection never returns an empty extent.

Because of the duality in the lattice, we might expect that the de-selection (removing a previously selected item) can be implemented by the join (least upper bound) operation, however this is not the case. Intuitively the de-selection of the most recently selected item should return the focus concept to its previous position, undoing the selection. However, computing the join of the focus with the attribute concept of the new de-selection will cause all previous selections to be removed, except the attribute we are de-selecting, which is counterintuitive. Therefore, in order to reverse a single selection operation we need to recompute the focus as the meet in the lattice from all *still-selected* items.

Our de-selection performs essentially the same operation as illustrated by Lindig [11], although Lindig optimizes this operation by making use of the search path in order to compute the new focus concept. Note that de-selections do not always need to take place in the same order as the initial selections. The de-selection operation can return a focus which has not been visited during the previous navigation steps. De-selection is therefore not only an undo operation.

4 Boolean Disjunctive Selection

The meet operation in the lattice satisfies conjunction between selected items. The meet of the attribute concept of item a ($\mu(a)$) AND the attribute concept of item b ($\mu(b)$), results in a concept whose intent contains both items a AND b. However, boolean OR navigation, where an attribute must only apply to *at least one* object is not supported by either the meet or join operations.

Priss [12] makes use of a boolean disjunctive query operation which returns the union of the extents of the concepts that are retrieved for each of the items in the query when selected individually. This approach requires more than one focus to generate the query's result.

Codocedo et al.'s approach [14] (illustrated in Figure 4) does not implement a purely disjunctive query operation as the query generators are not necessarily the attribute concepts of the items selected for the disjunctive query.

In our approach (Figure 3) we alter the underlying context table in order to support the disjunctive navigation and maintain the single-focus property. By

updating the underlying context we are able to support further navigation steps (refinements or generalizations) and maintain the disjunctive queries using only a single focus concept.

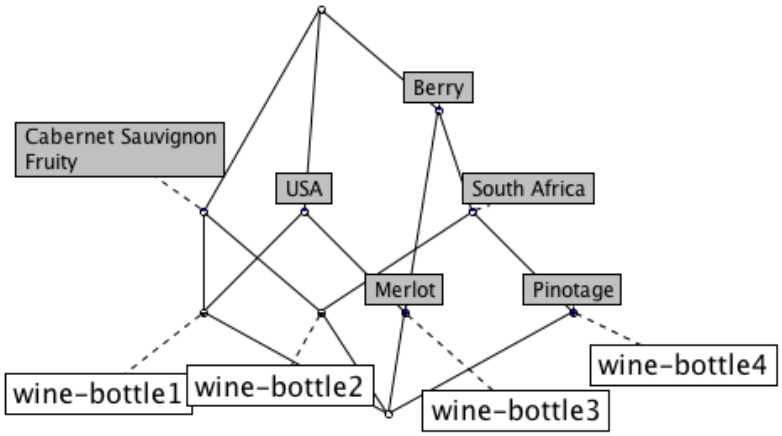

Fig. 2: Full concept lattice generated from the wine data context in Figure 1. The attributes are generated from different facets such as wine varietals and the objects are the individual wine bottles. Lattice generated with Concept Explorer [19]

In order to compute the boolean OR of two items, a and b, we merge the items in the underlying context table into a new attribute $a\ OR\ b$. We compute the introducing concept of the newly created merged item ($\mu(a\ or\ b)$) as the new focus. Our approach therefore returns the same query results as those that would be obtained in [12] for a *single* disjunctive query with no consequent navigation steps.

Figures 2 and 3 illustrate our approach. Figure 2 shows the initial concept lattice generated from the unaltered context. However, if we select item "Cabernet Sauvignon", the focus will be the meet of $\top$ and the attribute concept of "Cabernet Sauvignon" (resulting in the attribute concept of "Cabernet Sauvignon"). Selecting "Merlot" for a boolean OR operation, the context will be updated to add the combination of the two attributes to the context and the updated lattice will appear as in Figure 3.

The join of the focus and the attribute concept of "Merlot" would return the top concept in the lattice and our result-set would contain wines of other varietals (such as "Pinotage") which is undesirable. By using the boolean OR operation we are able to retrieve all wines that are only of the "Merlot" OR "Cabernet Sauvignon" varietals and the attributes which these wines possess.

Note that this approach is similar to the use of conceptual scales in the concept lattice [20] for multi-valued attributes (such as prices). However, instead of using pre-defined scales, our scale is generated automatically when the user

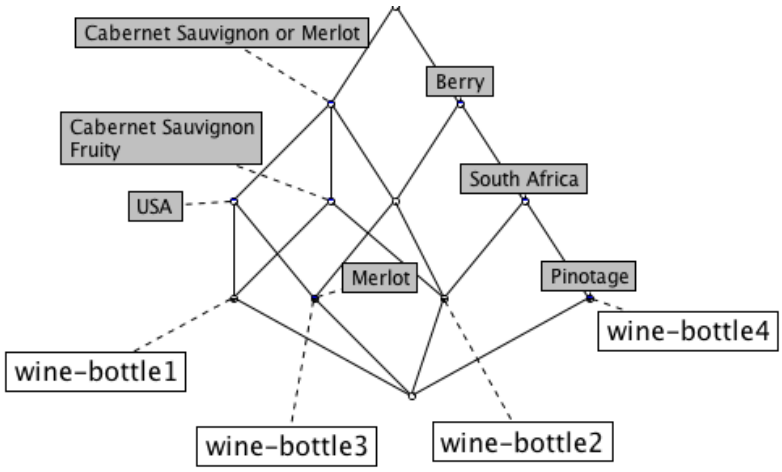

Fig. 3: Full concept lattice generated from the wine data context in Figure 1; where attributes "Cabernet Sauvignon" and "Merlot" have been selected for boolean OR navigation. A new concept with attribute "Cabernet Sauvignon or Merlot" has been inserted into the lattice. Lattice generated with Concept Explorer [19]

makes a boolean OR selection of an item in the dataset. The scale is therefore interactively created and we update the context on-the-fly.

5 Broadening Navigation Approach

The join operation supports broadening navigation, however, if the extents of both concepts are large then the join is likely to overgeneralize and can result in the top concept ($\top$) thereby losing all previous navigation steps and resulting in a low precision for the constructed query.

In order to support a broadening selection, that does not overgeneralize and return a concept with a large extent (and little or no common attributes) resulting in a low precision, we compute the join from only a *subset* of the objects in the full extents of the two concept selections. If the join of the two concepts is not $\top$ then we return their join, otherwise we recompute the join after removing one or more objects, until the join does not result in the top concept.

5.1 Generating Candidate Focus Concepts

Our broadening approach results in an updated focus that shows attributes which are common to *some* of the objects in the current focus *and some* of the objects in the attribute concept of the new item ($\mu(b)$) selected for broadening.

For example, in our wine review dataset if we select winery a for refinement and then broaden on winery b, our updated focus will show which wine characteristics (text from reviews etc.) are common to *some* bottles produced at

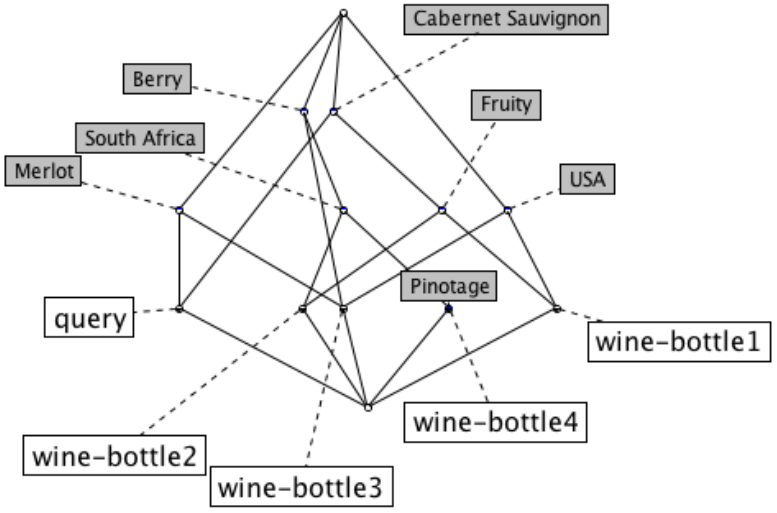

Fig. 4: Query concept with intent "Merlot" and "Cabernet Sauvignon" inserted into the concept lattice. Lattice generated with Concept Explorer [19]

winery b and *some* bottles produced at winery a. The join would reveal only characteristics that are common to *all* wine bottles from winery a and winery b, and since the set of all bottles from winery a and winery b it is likely to be large, the join risks navigating to top ($\top$).

If the join does not return $\top$, we return the join concept as the new focus, otherwise we traverse the lattice with a top-down depth-first approach using the focus as starting point. For each new concept in this traversal we then also perform a top-down depth-first traversal starting at $\mu(b)$. We compute the join of every concept derived from these iterations as candidate focus concepts for the next navigation step as shown in Algorithm 1.

Note that the amount of candidate focus concepts could be large and therefore we need to select a new focus from this pool in order to maintain only a single focus.

5.2 Selecting a new Focus Concept from the Candidate Focus Concepts

We choose a single focus from the generated set of candidate focus concepts. If a join for the previous focus (a = (A, B)) and the introducing concept of the selection (b = (C, D)) that is not the top concept in the lattice exists then we return the join concept, otherwise if the join results in the top concept then we want to return the highest concept (with the largest extent) such that at least one object from concept (a) is present and at least one object from concept (b) is contained in the extent.

```
Data: Current focus concept f, and attribute concept of selected item b, μ(b)
Result: Focus concept candidates for broadening navigation
iterator = top_down_traversal starting at f
while iterator.next is not null do
    f_subset = iterator.next
    inner_iterator = top_down_traversal starting at μ(b)
    while inner_iterator.next is not null do
        μ(b)_subset = iterator.next
        concept = join of μ(b)_subset and f_subset
        add concept to candidate_focus_concepts
    end
end
```

Algorithm 1: Computing candidate focus concepts for our broadening navigation operation. Join operations are computed using only a subset of the objects in the extents of the attribute concepts of the selections.

We therefore return the concept ($c = (E, F)$) which results in the highest score where the score is computed as

$score = |E| - \big||A| - |C|\big|$ where $|A| > 0$ and $|C| > 0$.

Our broadening operation therefore generalizes as much as possible without losing all previous selections and navigation steps and removing all previous navigation steps (navigating to $\top$).

6 Finding Similar Objects

Another method of generalizing from a single object in the dataset is to find a group of related or similar objects. To find objects that are similar to a selected object in the dataset we introduce a *more like this* operation. For example, if we want to find bottles of wine that are similar to a bottle that we have previously tried (i.e. generalize from a single wine bottle), we can apply the *more like this* operation to shift our focus to a concept that contains similar bottles, without the user needing to be aware of any of the attributes of the wine.

All concepts in the lattice in which the object of interest appears in the extent can be considered to present similar objects. However, in multi-faceted data, we are interested in finding a similarity between the objects in comparable facets (e.g, wine bottle 1's origin and wine bottle 2's origin). We also restrict the operation to returning results from only a single concept in the lattice so that all subsequent navigation steps can continue after a *more like this'* generalization operation.

Since not all facets can be used to compare objects, for example being reviewed by the same wine reviewer may not imply that two wine bottles are similar, we use only relevant facets (such as the wine review text and varietal) to compare objects. Various objects in the lattice will be similar across different dimensions. We look at descriptors from relevant facets of the object that are introduced lower in the lattice (are more specific) and include as many of these

as possible in the meet calculation to derive the new focus. However, if specific terms result in the meet returning only the original object of interest then we remove these terms in order to move the focus up and generate a larger extent.

Calculating the size of the extent of the attribute concept can provide a kind of TF/IDF [21] measure for the attribute in the entire corpus of objects. If the extent of an attribute concept is large, then the objects in that extent are unlikely to be very similar, since the attribute can be considered to be less specific as it applies to a large portion of the corpus.

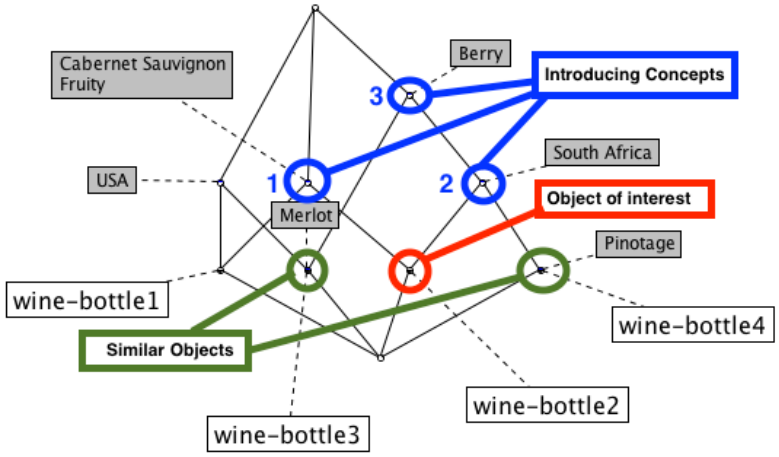

Fig. 5: Finding Similar Objects in a Concept Lattice. The object of interest is indicated in red, its attribute concepts are labeled 1,2 and 3. Similar objects (bottles 3 and 4) are indicated in green.

Figure 5 provides an example of our approach to finding similar objects. Wine bottle 2 (indicated in red) is the object of interest. The introducing concepts of the attributes of wine bottle 2 (berry, south africa, fruity, cabernet sauvignon) are indicated in blue. The meet of all three of these concepts would lead only to wine-bottle 2. Therefore the concept with the smallest extent is removed from the meet set first in order to move the focus up. Since concepts 1 and 2 both have an equal extent size, we use the size of the intent in order to decide which concept to remove from the meet calculation. Concept 1 provides two introducing attributes and so we remove concept 2 from the meet set. The meet of concepts 1 and 3 returns only wine bottle 2 (providing no similar wine bottles) and so concept 1 is subsequently removed from the meet set, leaving only concept 3. Therefore, wine bottles 3 and 4 are considered similar to wine bottle 2 in our approach because according to their reviews they all share flavors of “berry”. Although bottle 1 can also be considered similar to wine bottle 2 as they both share attribute (“Merlot”), our approach favors the more general concept (concept 3) so that the updated focus has a larger extent, including more similar objects.

7 Related Work

There have been many applications of concept lattices in the information retrieval domain [22], for example, the FaIR [12] and Credo systems [5].

The FaIR information retrieval system [12] combines a lattice-based thesaurus approach with boolean queries. The lattice-based thesaurus is used to generate the query language. Terms from each facet are separated into different lattices, unlike in our approach where the term and the facet name are used to represent a term in a single lattice. The thesaurus is used to add synonyms to the lattice so that queries with a wider vocabulary can be handled.

Credo [5] facilitates the exploration of web search results. Index terms from each retrieved search result are extracted from the documents. Credo supports refinement of the search results by selecting additional terms. Initially the presented information is derived from the lattice's top element and possible refinements are presented to the user. These refinement terms can then be selected to display a more specific set of search results and refine the initial query. Credo only includes support for refinement navigation.

8 Conclusions

In this paper we have discussed step-wise refinement and broadening navigation approaches in concept lattices that maintain the single focus property. We have developed a broadening navigation algorithm that makes use of subsets of the extents of two concepts in order to prevent the join from resulting in the top concept in the lattice. We have modified the disjunctive navigation technique to allow only a single focus concept to be stored and used to generate the results of the disjunctive query, allowing consequent broadening and refinement navigation steps to take place. We have discussed refinement navigation in concept lattices and our de-selection operation which is able to reverse refinement selections and does not restrict the order of the de-selection operation. Our navigation approaches can be used to facilitate exploratory search in large concept lattices and allow subsequent refinement, broadening and boolean OR navigation operations to take place.

Acknowledgments

This research is funded in part by a STIAS Doctoral Scholarship, CAIR-SU and NRF Grant 93582.

References

1. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA (1986)
2. Rijsbergen, C.J.V.: Information Retrieval. 2nd edn. Butterworth-Heinemann, Newton, MA, USA (1979)

3. Poelmans, J., Ignatov, D.I., Viaene, S., Dedene, G., Kuznetsov, S.O., et al.: Text mining scientific papers: A survey on fca-based information retrieval research. In: ICDM. Volume 7377., Springer (2012) 273–287
4. Poelmans, J., Ignatov, D.I., Kuznetsov, S.O., Dedene, G.: Formal concept analysis in knowledge processing: A survey on applications. Expert Syst. Appl. **40**(16) (2013) 6538–6560
5. Carpineto, C., Romano, G., Bordoni, F.U.: Exploiting the potential of concept lattices for information retrieval with credo. J. UCS **10**(8) (2004) 985–1013
6. Marchionini, G.: Exploratory search: from finding to understanding. Communications of the ACM **49**(4) (2006) 41–46
7. Godin, R., Pichet, C., Gecsei, J.: Design of a browsing interface for information retrieval. SIGIR Forum **23**(SI) (May 1989) 32–39
8. Godin, R., Saunders, E., Gecsei, J.: Lattice model of browsable data spaces. Information Sciences **40**(2) (1986) 89–116
9. Lindig, C.: Concept-based component retrieval. In: Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs. (1995) 21–25
10. Carpineto, C., Romano, G.: Effective reformulation of boolean queries with concept lattices. In: Flexible Query Answering Systems. Springer (1998) 83–94
11. Lindig, C.: Algorithmen zur Begriffsanalyse und ihre Anwendung bei Softwarebibliotheken. PhD thesis, TU Braunschweig (1999)
12. Priss, U.: Lattice-based information retrieval. Knowledge Organization **27**(3) (2000) 132–142
13. Carpineto, C., Romano, G.: Order-theoretical ranking. Journal of the American Society for Information Science **51**(7) (2000) 587–601
14. Codocedo, V., Lykourentzou, I., Napoli, A.: A semantic approach to concept lattice-based information retrieval. Annals of Mathematics and Artificial Intelligence **72**(1-2) (2014) 169–195
15. Fischer, B.: Specification-based browsing of software component libraries. Autom. Softw. Eng. **7**(2) (2000) 179–200
16. Prieto-Diaz, R.: Implementing faceted classification for software reuse. Communications of the ACM **34**(5) (1991) 88–97
17. : Wine review online. `http://www.winereviewonline.com/`
18. Ferré, S.: Camelis: a logical information system to organise and browse a collection of documents. International Journal of General Systems **38**(4) (2009) 379–403
19. : Concept explorer. `http://conexp.sourceforge.net/`
20. Ganter, B., Wille, R.: Conceptual Scaling. In: Applications of Combinatorics and Graph Theory to the Biological and Social Sciences. Springer US, New York, NY (1989) 139–167
21. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation **28**(1) (1972) 11–21
22. Codocedo, V., Napoli, A.: Formal concept analysis and information retrieval–a survey. In: Formal Concept Analysis. Springer (2015) 61–77