Browsing Digital Collections with Reconfigurable Faceted Thesauri

Joaquín Gayoso-Cabada, Daniel Rodríguez-Cerezo and José-Luis Sierra

1 Introduction

Faceted navigation is a common interaction technique in business, the cultural industry and many other domains [2, 18, 26, 29, 30, 32]. For this purpose, resources are classified in terms of suitable *faceted thesauri*. A faceted thesaurus groups classification terms into facets, which in turn can have associated sub-facets, yielding a hierarchical arrangement. This hierarchical organization can, in turn, guide navigation through the underlying collection of digital resources (regardless of whether these are records in a database, objects in a virtual museum, entries in a virtual shop catalog, or any other type of digital object).In mature digital collections, in which there are few or no changes in the underlying resources, and in which classification schemata are pre-established and stay immutable, faceted navigation can be accomplished in very efficient ways [7]. However, for live collections, such as those arising in social or other highly dynamic and changing environments, not only are changes in the underlying resources frequent, but these changes can also affect the classification schemata themselves. When faceted thesauri are used in these dynamic settings, reconfiguring the thesaurus can mean a profound rearrangement of the collection's internal structures, which can be costly

A prior version of this paper has been published in the ISD2016 Proceedings (http://aisel.aisnet.org/isd2014/proceedings2016).

J. Gayoso-Cabada · D. Rodríguez-Cerezo · J.-L. Sierra (✉)
Complutense University of Madrid, Madrid, Spain
e-mail: jlsierra@ucm.es

J. Gayoso-Cabada
e-mail: jgayoso@ucm.es

D. Rodríguez-Cerezo
e-mail: drcerezo@ucm.es

J. Gołuchowski et al. (eds.), *Complexity in Information Systems Development*,
Lecture Notes in Information Systems and Organisation 22,
DOI 10.1007/978-3-319-52593-8_5

in time (often, it must be carried out offline). In consequence, user experience can be seriously hindered. Indeed, when a user changes the thesaurus, what he/she probably expects is an almost instant response in navigation; in these cases, high response times and/or a temporarily outdated underlying information system are inadmissible. We have realized this fact during the compilation of research and education-oriented collections of digital objects in digital humanities scenarios [4, 23, 24]. In these scenarios faceted thesauri-like classification schemata were subjected to continuous change, refinement and evolution throughout the collections' life cycles. Many times those reconfigurations in the schemata were performed with experimental and/or exploratory purposes in mind, and domain experts (researchers and/or instructors in charge of compiling and maintaining the collections) were not willing to wait for long periods until the changes were reflected in their collections. On the contrary, they wanted to see the changes in the browsing system immediately after changing the classification schemata, in order to determine whether these changes in the schemata really met their expectations. Thus, in this paper we partially respond to these needs by firstly providing a model of digital collection with a reconfigurable faceted thesaurus, in which the facet hierarchy can be freely rearranged, thus accomplishing the exploratory needs of the potential users. Secondly, we also introduce indexing strategies that provide reasonable time-space tradeoffs concerning navigation reconfigurability, while preserving acceptable levels of user experience.

The rest of the paper is organized as follows. Section 2 describes the digital collection model. Section 3 addresses browsing in the presence of the kind of reconfigurable thesauri introduced by this model. Section 4 introduces some works related to our browsing approach. Finally, Sect. 5 outlines the final conclusions and some lines of future work.

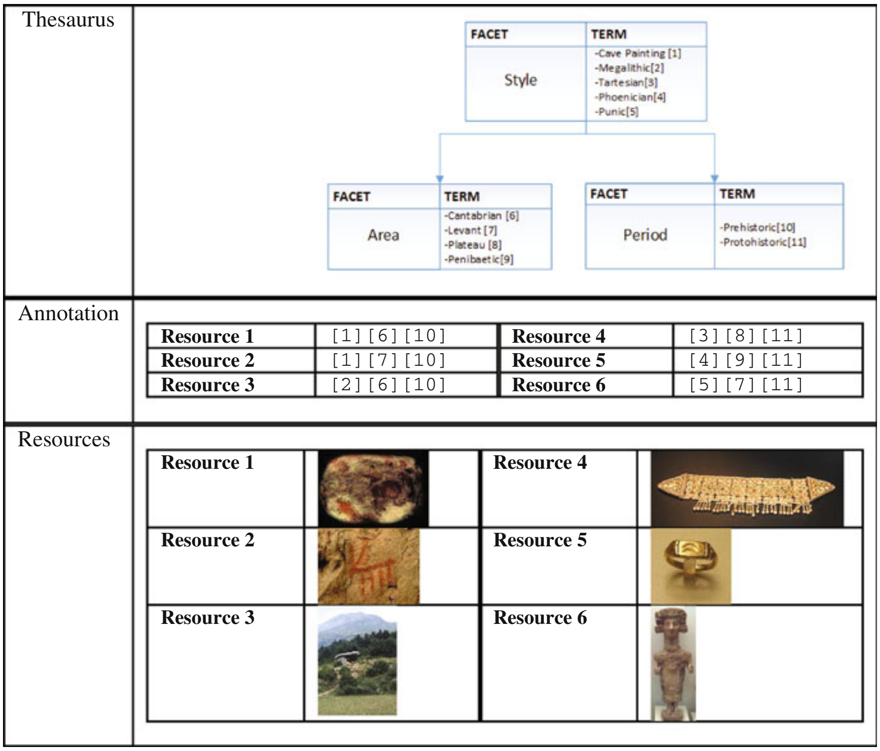
2 Digital Collections with Reconfigurable Faceted Thesauri

In this section we introduce our model of digital collection with reconfigurable thesauri. Section 2.1 describes the structure of these collections. Section 2.2 addresses thesauri reconfiguration.

2.1 Structure of the Collections

Our collections comprise the following parts (see Fig. 1 for an example):

– On one hand, there are the *resources* in the collection. These resources are digital objects whose nature is no longer constrained by the model. Thus, these resources can be media files (images, sound, video, etc.), external resources



Resource 1	[1][6][10]	**Resource 4**	[3][8][11]
Resource 2	[1][7][10]	**Resource 5**	[4][9][11]
Resource 3	[2][6][10]	**Resource 6**	[5][7][11]

Fig. 1 A small digital collection

identified by their URIs, or entities of a more abstract nature (tuples of a table in a relational database, records in a bibliographical catalog, elements in an XML document, rows in a spreadsheet, etc.). For instance, the small collection depicted in Fig. 1 includes six image archives as resources, corresponding to photographs of artistic objects from the Prehistoric and Protohistoric artistic periods in Spain (Fig. 1 actually shows thumbnails of these images).

– On another hand, there is the *annotation* of the resources. This annotation consists of associating descriptive *terms* with resources. These terms are useful when cataloguing resources and, therefore, they enable future uses of the collection (navigation, search, etc.). Since each term has a unique identifier associated, annotating a resource consists of associating a set of term identifiers with such a resource. For instance, in Fig. 1 resource number 1 has the terms identified by **[1] [6] [10]** associated.

– Finally, there is a faceted *thesaurus* that organizes the terms into facets and which arranges these facets hierarchically. For instance, the faceted thesaurus in Fig. 1 includes a root facet *Style*, representing the artistic style used, and two sub-facets: *Area* (representing the geographical area), and *Period* (representing the artistic period). Each facet includes representative terms related to this facet.

Notice how each term consists of a descriptive name and the aforementioned unique identifier. In this way, the terms indicated below (**[1] [6] [10]**) actually refer to the terms *Cave Painting* in the facet *Style*, *Cantabrian* in the facet *Area*, and *Prehistoric* in the facet *Period*.

2.2 Thesauri Reconfiguration

Our model lets users reconfigure thesauri by rearranging the hierarchical organization of facets in order to accommodate their experimental and/or exploratory needs. For instance, Fig. 2 shows an example concerning the collection in Fig. 1. Indeed, the thesaurus in Fig. 2a (the original thesaurus in Fig. 1) reflects a structure primarily focused on the artistic style. Once this style has been set, it is possible to introduce either a geographical or an artistic period refinement. However, it may also be feasible to conceive of an alternative organization, with the artistic period as main focus, and with the geographical area and style as secondary features. This leads to the thesaurus in Fig. 2b, which has been obtained from the original one by altering the hierarchical facet arrangement.

Since the organization of a collection ultimately relies on its faceted thesaurus, by reconfiguring this thesaurus it is possible to implicitly reconfigure the structure of the entire collection, adapting it to different use scenarios as needed. This effect can be readily appreciated on the *navigation map* of a collection. Such a map is a directed graph in which:

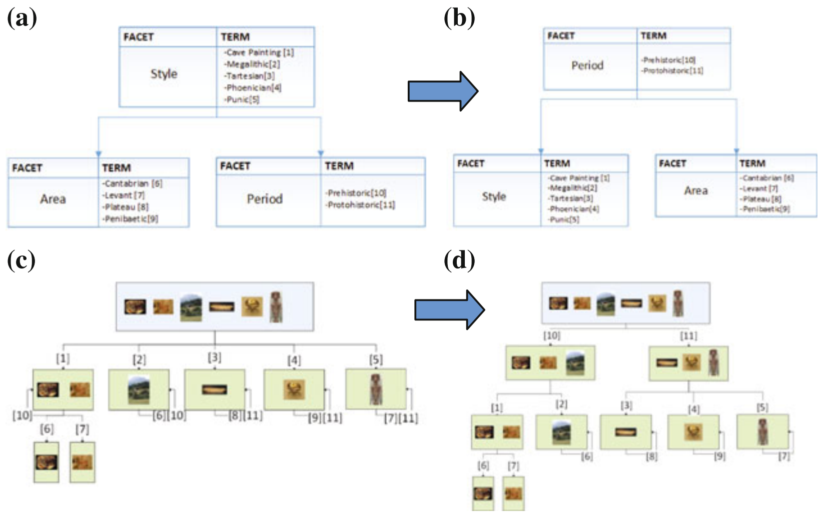


Fig. 2 **a** Original thesaurus in Fig. 1; **b** Reconfigured thesaurus; **c** Navigation map induced by (**a**); **d** Navigation map induced by (**b**)

- Nodes represent sets of resources, and arcs are labelled with terms used to narrow down the resources in the source nodes in order to yield the resources in the target ones (actually, all those resources in the source nodes annotated with the terms in the arcs).
- Structure is constrained by the facet hierarchy. In this way, root nodes can only be narrowed down with terms in root facets, and, if a node is produced by a term in a facet, it can only be narrowed down with terms in sub-facets of the mentioned facet.

Thus, reconfigurations in the thesaurus affect the entire navigation map. It is made apparent in Fig. 2c, d, which, respectively, outline navigation maps of the collection in Fig. 1 before (Fig. 2c) and after (Fig. 2d) thesaurus reconfiguration.

3 Browsing with Reconfigurable Faceted Thesauri

As the previous section makes apparent, reconfigurations in the hierarchical structure of a thesaurus profoundly impact the structure of the overall collection. In particular, after reconfiguration, the collection's navigation map can be completely altered. This hampers the use of efficient implementations of faceted browsing (e.g., [7]), which are basically driven by the navigation map structure and therefore require pre-established and unmodifiable faceted thesauri (otherwise, the navigation map would have to be regenerated after each thesaurus' reconfiguration, which would be a costly task even for collections of moderate size, and which could seriously impact user experience). Thus, by allowing reconfigurability, it is necessary to switch to alternative representations, enabling *all* the possible navigations induced by *all* the possible reconfigurations of the faceted thesaurus. Section 3.1 characterizes the expected behavior as a finite state machine. Section 3.2 addresses some complexity issues associated with a naïve representation directly based on such a machine. Finally, Sect. 3.3 discusses some indexing approaches that we have used to face these complexity issues.

3.1 Navigation Automata

In order to characterize all the possible navigations induced by all the possible reconfigurations of a faceted thesaurus it is necessary to free terms from the faceted structure. Therefore, a plain set of terms must be considered and, in each interaction state, all the meaningful selection of terms must be applied. The result can be represented as a finite state machine that we will call a *navigation automaton*. This automaton will consist of *states* labelled by sets of resources, and *transitions* labelled by terms. More precisely:

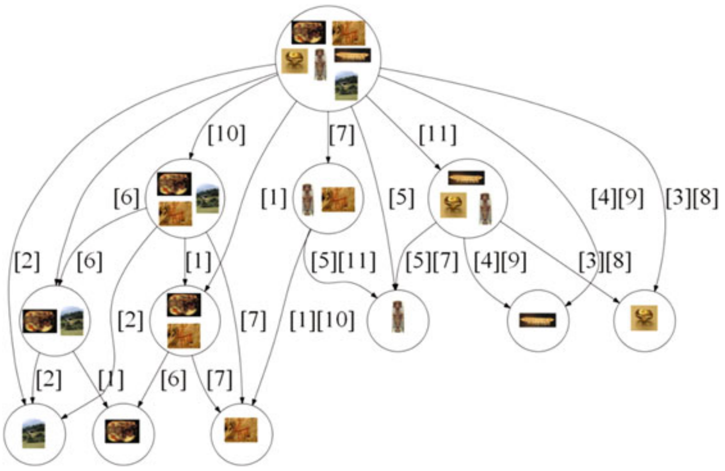


Fig. 3 Navigation automaton for the collection in Fig. 1

- There will be an initial state labelled by all the resources in the collection.
- Given a state S labelled by a set of resources R, for each term t annotating some resource in R there will be a state S' labelled by all the resources in R annotated by t, as well as a transition from S to S' labelled by t.[1]

Figure 3 shows the navigation automaton for collection in Fig. 1. Notice that the navigation automaton does not depend on the hierarchical organization of facets in the thesaurus, but only on the terms and on the resources in the collection. Therefore, it is not affected by reconfigurations in the thesaurus. Indeed, it can be thought of as the amalgam of all the possible navigation maps induced by all the possible reconfigurations of the collection thesaurus.

Since the navigation automaton embeds all the possible navigation maps, faceted browsing with respect to a particular thesaurus configuration can be formulated in a straightforward way, since there will be a direct correspondence among interaction states in the browsing process and states in the navigation automaton. In addition, in each interaction state will be a set of allowable facets to be explored. Indeed:

- The browsing process will start by considering the navigation automaton's initial state and the thesaurus' root faces as allowable ones.
- In each interaction state, the allowable facets will be used to constraint the possible terms to continue browsing. Once an allowed term is selected, the navigation automaton will be used to establish the new navigation state and the thesaurus to update the allowable facets.

[1]Notice that S and S' can be the same—when all the resources in R are annotated by t.

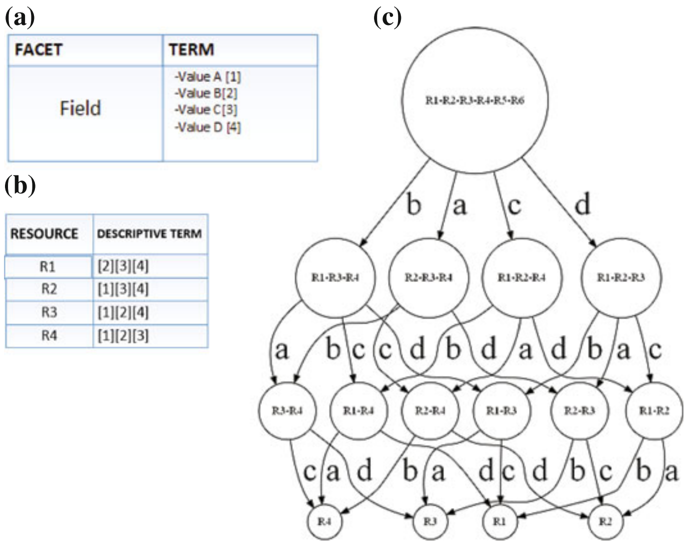
3.2 Complexity Issues

As indicated in the previous subsection, the explicit availability of the navigation automaton provides an elegant and efficient solution to faceted browsing in the presence of a reconfigurable thesaurus. Unfortunately, for collections with dense annotations there is the risk of facing unacceptably growing rates in the number of resulting states. It should not be surprising since we are attempting to represent all the possible ways of navigating in a single structure, regardless of the structure of the underlying thesaurus. In the worst case, the number of states can grow exponentially with respect to the number of resources. This extreme case, in which the number of states is 2^n-1 (with n the number of resources), arises, for instance, by distinguishing each pair of resource annotations in a single term (Fig. 4 shows an example with 4 resources and 4 terms).

While the extreme case presented can be somewhat artificial, it cannot be ignored if we hope to deal with arbitrary evolving collections. For this purpose, it can be desirable to look for alternative indexing approaches to enable the dynamic recreation of the relevant parts of the navigation automaton during browsing while preserving required levels of user experience.

3.3 Indexing Strategies

In order to deal with the complexity issues raised in the previous subsection, we have explored two different indexing strategies: *inverted indexes* and *navigation*


(a)

FACET	TERM
Field	-Value A [1] -Value B[2] -Value C[3] -Value D [4]

(b)

RESOURCE	DESCRIPTIVE TERM
R1	[2][3][4]
R2	[1][3][4]
R3	[1][2][4]
R4	[1][2][3]

Fig. 4 Example of exponential explosion: **a** a simple thesaurus, **b** a simple associated collection, **c** the resulting navigation automaton

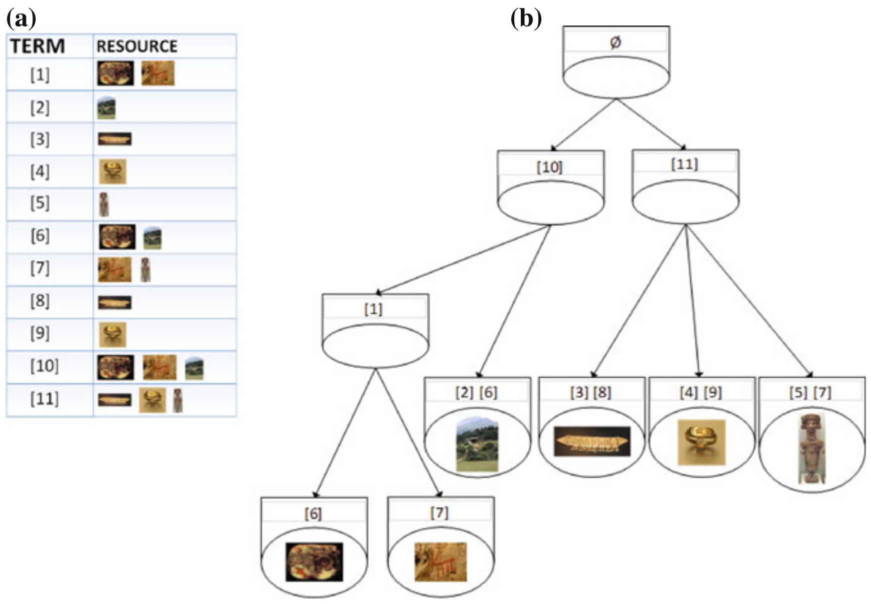


Fig. 5 **a** An inverted index for the collection in Fig. 1a; **b** a navigation dendrogram for such a collection

dendrograms. Following paragraphs analyze these strategies and provides some empirical comparison results.

Inverted Indexes

Inverted indexes are standard artifacts used for information retrieval [33]. Basically, for each description term, an inverted index associates the set of resources annotated with this term. Figure 5a shows an example of inverted index for the collection in Fig. 1.

Inverted indexes can be used to recreate the browsing behavior described in the previous section in a straightforward way. Basically, it suffices to maintain interaction states consisting of the set of terms chosen and the set of allowable facets to be explored.

- The initial state is constituted by the empty set of terms and by the root facets.
- Given an interaction state the selection of the next term to be applied obeys the same constraints as with the navigation automaton: this term must be a term *t* included in some facet *f* among the allowable facets. Then, the new interaction state will consist of: (i) all the terms in the previous one plus the new term selected, (ii) all the sub-facets of *f*.

Concerning the resources filtered in each interaction state, these resources can be determined by considering the set of terms $\{t_1, \ldots, t_n\}$ in such a state and by evaluating the conjunctive query $t_1 \wedge \ldots \wedge t_n$ using the inverted index.

The cost of evaluating the queries $t_1 \wedge \ldots \wedge t_n$ in each interaction state constitutes the main shortcoming of the approach. Indeed, it involves finding the intersections of the sets for $t_1 \ldots t_n$ in the inverted index. While there has been extensive research in performing these intersection operations efficiently [5], the cost is not negligible.[2] On the positive side is the availability of many mature implementations and frameworks that can be used in a straightforward way to support the technique. For instance, in our experiences, we used Lucene [17] for such a purpose.

Navigation Dendrograms

In order to avoid the proliferation of intersection operations, which is characteristic of inverted indexes representations, we have envisioned a tree-shaped indexing scheme inspired by *dendrograms* in hierarchical clustering [13]. The resulting representations are called *navigation dendrograms*. Following hierarchical clustering principles, nodes in the dendrogram represent subsets of the overall resource set. In this way:

– The dendrogram's root represents the whole resource set.
– If a node represents a particular resource set, then each child node represents a partition of this set (i.e., child nodes represent mutually disjoint subsets of the parent's set).

The resource set associated to a node is not explicitly stored in this node. Instead, each resource is only hosted in one node (the resource's *host node*). Resources placed in a node are called the mentioned node's *own resources*. The overall resource set of a node is given by its own resources and by all the own resources of its descendants. Finally, in order to partition the resource space, each node has a set of *filtering terms* associated, so that all the own resources in the node and in all their descendants' must be annotated with these filtering terms (the node is said to *filter* those resources). Figure 5b shows a navigation dendrogram for the collection in Fig. 1.

Initially, the dendrogram contains a single root node with an empty filtering set. Then, the dendrogram is incrementally constructed by sequentially adding resources, one resource at a time. Pseudocode in Fig. 6 details how a new resource is added to the dendrogram. Basically the process proceeds by looking for a host node for the resource, creating new nodes when needed. For this purpose the resource is firstly filtered through the dendrogram's nodes. When a node with no children filtering the resource is reached:

– If the resource has some term not included in any of the filtering term in the traversed nodes, and there is a child node containing some of these non-considered terms in its filtering set, a new intermediary *fork* node is created

[2]Notice that, although as indicated earlier, frameworks like Solr [7] support faceted browsing in a straightforward and efficient manner by identifying paths in the thesaurus with terms, in our context these features are useless, since thesauri can be reconfigured anytime, thus invalidating this solution. So we are confined to explicitly evaluating conjunctive queries in each interaction state.

```
Add resource r to dendrogram d:

CurrentNode = d's root
$\Delta_{Res}$ = r's terms

while there is some child n of CurrentNode such as
      n's filtering terms $\subseteq \Delta_{Res}$ {
    CurrentNode = One of such child nodes
    $\Delta_{Res}$ = $\Delta_{Res}$ - CurrentNode's filtering terms
}

InsertionNode = CurrentNode
if ($\Delta_{Res} \neq \emptyset$) {
   if there is some child n of InsertionNode such as
      n's filtering terms $\cap \Delta_{Res} \neq \emptyset$ {
          ChildNode = One of such child nodes
          ForkNode = create new node
          $\Delta_{Fork}$ = $\Delta_{Res} \cap$ ChildNode's filtering terms
          $\Delta_{Child}$ = ChildNode's filtering terms - $\Delta_{Fork}$
          $\Delta_{Res}$ = $\Delta_{Res}$ - $\Delta_{Fork}$
          set ForkNode's filtering terms to $\Delta_{Fork}$
          set ChildNodes'filtering terms to $\Delta_{Child}$
          change the arc InsertionNode $\rightarrow$ ChildNode
                 to InsertionNode $\rightarrow$ ForkNode
         add an arc ForkNode $\rightarrow$ ChildNode
         InsertionNode = ForkNode
   }
  if ($\Delta_{Res} \neq \emptyset$)) {
         HostNode =  create new node
         set HostNode's filtering terms to $\Delta_{Res}$
         set HostNode's own resources to $\emptyset$
         add an arc InsertionNode $\rightarrow$ HostNode
         InsertionNode = HostNode
   }
 }
add r to InsertionNode's own resources
```

Fig. 6 Pseudocode of the process for adding a resource to a navigation dendrogram

with this child node as child. In addition, the resource node is placed in the fork node or in a new host node (which is also included as a child of the fork node) depending on whether it has some term not included in the reached node's filtering set. In other case, the resource is also placed in a new host node, which is added as a new child node of the reached one.
– Otherwise, the resource is hosted in the reached node.

Thus, in the worst case, insertion of a resource involves the creation of two new nodes. In consequence, the number of nodes in the dendrogram is bound by $2R$ (with R the number of resources).

Concerning browsing, it is possible to conceive interaction states formed by:

– A set of *dendrogram nodes*, which are active in the interaction state.
– A set of allowable facets.

```
Find next interaction state of s given t of facet f:

notation: Given a dendrogram's node n:
    •  $n^{\uparrow}$: all the ancestors of n (including n itself)
    •  $n^{\downarrow}$: all the descendants of n (exluding n)
let N the set of dendrogram nodes in s {
    N' = ∅
    foreach n in N {
       if there is n' in $n^{\uparrow}$ such as t is in the filtering terms of n' {
          N' = N' ∪ {n}
       }
      else let $n^{\downarrow}_t$ the nodes in $n^{\downarrow}$ with t in their filerting terms {
         N' = N' ∪ $n^{\downarrow}_t$
      }
    }

  let F the set of subfacets of f {
    The next interaction state has N' as the set of dendrogram nodes
    and F as the set of allowable subfacets
  }
}
```

Fig. 7 Pseudocode to find the next interaction state during browsing

As in the earlier proposals, navigation firstly proceeds by selecting a term from one of the allowable facets. Then, the next interaction state can be obtained by:

– Establishing the allowable facets to the subfacets of the facet used.
– Refining the set of nodes according to the term selected. Nodes having such a term in their filtering terms or in the filtering terms of some ancestor are directly preserved. Nodes having this term in the filtering terms of some descendant are replaced by said descendants. Other nodes are discarded (see Fig. 7). Notice that, in order to speed up this computation, it is convenient to have direct access to the filtering terms for each node and its ancestors, as well as to have the node's descendants classified by their filtering terms (for the sake of simplicity, details are not shown in the pseudocode in Fig. 7).

Once the interaction state is determined, resources selected in an interaction state can be lazily recovered by iterating the dendrogram nodes and their descendants' own resource sets.

Finally, it is worthwhile to notice how navigation dendrograms overcome the main shortcoming of inverted indexes: the need to explicitly carry out set intersections during browsing. On the negative side, the indexing process is substantially more complex than in the case of inverted index construction.

Experimental Results

In order to compare the two indexing strategies described, we implemented both on *Clavy*, an experimental system for managing digital collections with reconfigurable

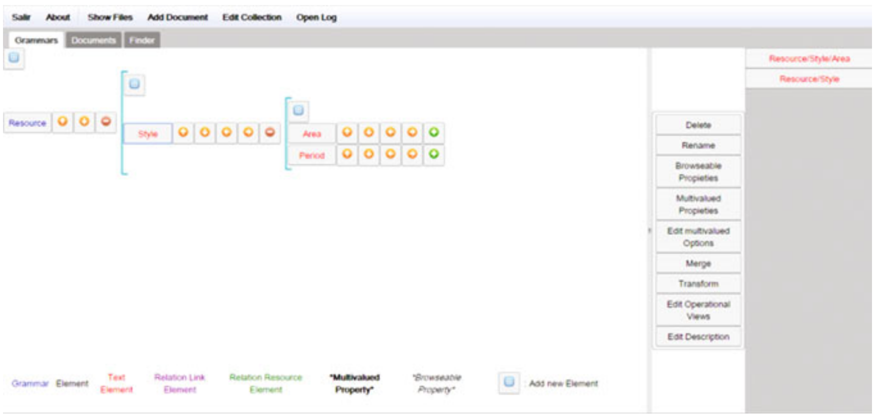


Fig. 8 Editing a reconfigurable thesaurus with Clavy

faceted thesauri-like schemata (see Fig. 8).[3] We also set up an experiment consisting of adding the resources in *Chasqui* [23],[4] a digital collection of 6283 digital resources on Precolombian American archeology, to *Clavy* and to simulate runs concerning browsing and schemata reconfiguration operations.

Each run was customized as follows. We interleaved resource insertion with browsing/reconfiguration rounds. Each insertion round consisted of 100 resource insertions (with the exception of the last one, in which the remaining resources where inserted). In turn, each browsing/reconfiguration round consisted of executing $0.1n$ browsing operations randomly interleaved with $0.01n$ reconfigurations (n being the number of resources inserted so far). Each browsing operation consisted, in turn, of selecting a feasible term, computing the next interaction state, and visiting all the resources filtered. Reconfiguration operations, then, consisted of feasible interchanges of two randomly selected facets,[5] followed by a browsing step.

Inverted indexes were managed using Lucene, while navigation dendrograms were managed using our own implementation (implemented in Java, as well as the Lucene framework). In both cases, in-memory indexes were used in order to avoid side effects of persistence, disturbing the experiment.

Figure 9 shows the results obtained from the two runs. The experiment was run on a PC with Windows 10, with a 3.4 GHz Intel microprocessor, and with 8 Gb of DDR3 RAM. The horizontal axis corresponds to the number of operations carried out so far. The vertical axis corresponds to cumulative time (in seconds). As is made apparent, the dendrogram-based approach clearly outperforms the inverted indexes (even though we are using a highly optimized framework, like Lucene, for inverted indexing vs. our own in-house experimental implementation for dendrograms).

[3]clavy.fdi.ucm.es/Clavy/.

[4]oda-fec.org/ucm-chasqui.

[5]By *feasible* we mean avoiding cycles in the resulting thesaurus.

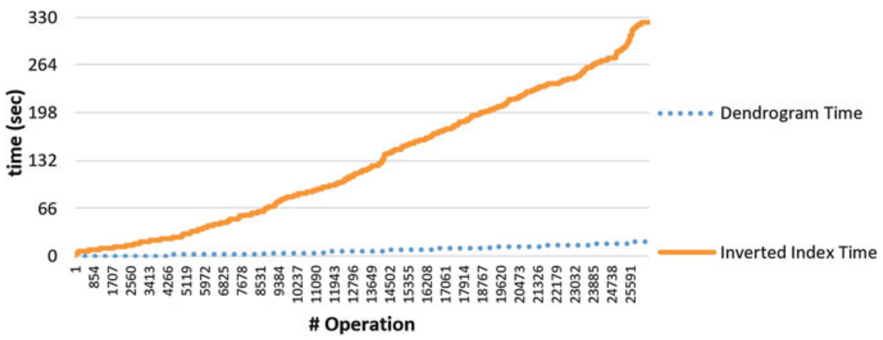


Fig. 9 Cumulative time of inverted indexes versus dendrograms

4 Related Work

There are several faceted browsing systems that, like ours, envision the possibility that the user reconfigures the underlying facet hierarchy. For instance, *mSpace* [22] makes it possible to organize information spaces in plain sets of facets, which can be interactively arranged in lists (*slices*), representing linear hierarchies between selected facets. Initially a slice may contain only a subset of the whole set of facets, and users can reorganize, add and delete the slice's facets according to their specific needs. In [21] an implementation based on semantic web (RDF) technologies is proposed. In [25] the advantages and disadvantages of this implementation are analyzed and more conventional solutions based on relational databases are proposed. In turn, */facet* [11] adopts a different approach as it is configured as a multi-faceted browser driven by RDFS schemata and RDF data. Facet hierarchies are inferred from RDF class hierarchies. In addition, being a multi-faceted browser, */facet* enables the user to jump from one facet hierarchy to another while maintaining filtering constraints. In consequence, actual facet hierarchies are not pre-established, but are dynamically configured according to the specific needs of each user. In addition, in [11] some efficiency issues associated with the extensive use of semantic web infrastructures are also reported. Contrary to *mSpace*, in which facets lack organization and in which users confer (linear) structures on these facets by arranging them in slices, or to */facet*, in which RDFS schemata pre-establish faceted hierarchies and in which the user is allowed to jump between hierarchies, in our approach, facets are doted with a hierarchical structure (the faceted thesaurus) from the beginning and users organize this hierarchy according to their needs. It provides users with more guidance during the reconfiguration process than *mSpace*'s approach and more freedom than */facet*'s (where reconfigurability consists of dynamically pasting different pre-established hierarchies). In addition, we propose efficient indexing approaches, specifically tailored to our model instead of piggybacking the implementation on general-purpose semantic web or relational database solutions.

Our navigation automaton model is actually equivalent to lattice-based proposals to browse information spaces, as described in the seminal work of [6]. In these proposals, resources are tagged with keywords. The lattice organization induced consists of nodes characterized by sets of resources and sets of keywords related by a *Galois connection* (i.e., the set of keywords is the intersection of the resources' keywords and the set of resources consists of all the resources filtered by the keyword set). This organization is actually the main subject of the fertile theory of *formal concept analysis* [20], where resources are called *objects*, keywords are called *attributes*, objects tagged with attributes are called *formal contexts*, and lattice nodes are called *formal concepts*. From this description, it should be apparent how states of navigation automata can actually be identified with formal concepts, and automata themselves with explicit representations of concept lattices (with an explicit representation of the whole order relation and an explicit labelling of the arcs with transition information). In [15] the intrinsic complexity of formal concept analysis is examined and the problem of determining the size of concept lattices is proved to be a #P-complete one (i.e., harder than NP-complete). In consequence, complexity results in concept lattice theory are directly translatable to navigation automata (in fact, construction of Sect. 3.2 was suggested by the proof of theorem 1 in [15]), In addition, there are several proposals on using concept lattices as the underlying indexing structures of digital collections. For instance, *ConceptCloud* uses concept lattices to support multilevel browsing of software repositories [10, 28] or academic publications [8]. A similar approach is described in [9], where concept lattices are used to organize the browsing of art-works from the University of Wollongong's Art Collection. All these proposals are also affected by the worst-case complexity of formal concept analysis.

Inverted indexes have been extensively used to support faceted browsing. In [31] the basic technique, as well as subsequent enhancements, are illustrated with the use of Lucene. In [27] an alternative approach, based on relational databases, is presented. However, all these approaches are based on the assumption of pre-established and immutable faceted thesauri. As noticed in [1], if this assumption is left out, and therefore arbitrary multilevel exploratory search is allowed, inverted indexes can become costly due to the set operations involved. For small amounts of terms, multidimensional structures (as used in data-warehouse and data-mining scenarios) can be advantageous [14]. However, the performance of these multidimensional approaches can dramatically decrease when dimensionality increases. For this purpose, in [1] a technique called *tree striping* is described, which proposes subdividing the overall information space in k disjoint sub-spaces, to apply standard inverted indexes or multidimensional indexing techniques to each resulting subspace and to use an efficient merging approach to aggregate the results of each subspace. Nevertheless, and contrarily to our navigation dendrograms, both multi-dimensional and tree striping techniques basically work with pre-established partitions of the information space, while in our approach thesauri are dynamic and evolving in nature.

Finally, it is worthwhile to notice that clustering techniques has been extensively used in social tagging systems (e.g., [12, 16, 19]) to enable the discovering of useful semantic relationships among tags in order to provide better guidance to users (e.g., by automatically discovering hierarchical structures of tags). Thus, clustering in these approaches is oriented to enhance users' browsing efficiency, while our navigation dendrograms are oriented to enhance the internal efficiency of the supporting software.

5 Conclusions and Future Work

Live digital collections, which involve active communities of specialized users (e.g., researchers or educators in a particular field), also require live organization schemata, which can be incrementally defined, refined and enhanced as collections evolve. In addition, in these scenarios users usually want systems to quickly respond to changes in the schemata, without waiting for costly and/or batch reorganization processes. In this paper we have addressed this problem of dynamic reconfigurability in the case of reconfigurable faceted thesauri, in which users can re-order facets in order to explore different and alternative ways of organizing the collections. Since facet hierarchy can be rearranged in unexpected ways, it is necessary to resort to a more free and exploratory browsing system. It has led us to model this system as a finite state machine, the *navigation automaton*, taking into account all the possible ways of navigating the collection, using terms selected from the facets. Unfortunately, we have also showed how, in some cases, the number of states in this automaton can increase exponentially with respect to the collection's size. In order to deal with this potential exponential factor we have explored two different indexing approaches: one based on standard inverted indexes (implemented in a robust and well-proven search framework: Lucene), and one inspired by hierarchical clustering techniques (the so-called *navigation dendrograms*). Some experiments with a real collection gave evidence of how the hierarchical clustering technique can outperform the inverted indexing one.

We are currently working on further optimizing our navigation dendrogram representation to leverage space requirements. Indeed, in order to provide efficient navigation we need to associate each node with the intersection and the union of all the terms annotating the resources under this node. In addition, for each union term we also need to store the descendant nodes that include such a term in their filtering sets. Fortunately, these sets present much regularity among nodes, which allows us to compress them by using tries and common node-set stores. Since the resulting structures provide time and space efficient representations for the nodes' intersection and union sets, all these optimizations enhance system performance, in addition to saving space. We are also looking for efficient ways to make all this information persistent, either by using standard relational databases or alternative NoSQL approaches (e.g., [3]), while causing again a minimum impact on system performance. Once efficient persistence mechanisms are established we want to run more empirical evaluations also taking persistence into account. We also hope to enhance our model with support for arbitrary Boolean queries and for different ways of exploring the resources selected. These mechanisms will be based on the navigation automaton model (supported by our indexing proposals, and, in particular, by navigation dendrograms) in order to get as much efficiency as possible. Finally, we plan to perform more comprehensive tests of our model in the context of different Digital Humanities efforts carried out by some of the Humanities research groups with whom we cooperate. Among these efforts we can highlight, in addition to the aforementioned *Chasqui* collection, different digital collections maintained by LEETHI, the UCM research group on European and Spanish Literatures, from Texts to Hypermedia (Mnemosine, a digital collection concerning rare texts from the Spanish Silver literature age,[6] Ciberia, a digital collection concerning Spanish digital literature[7] and Tropos, a digital collection concerning creative digital writing for literature education[8]), as well as those concerning the Panamanian "El Caño" archeological site,[9] created and curated by "El Caño" Foundation at Panama.

Acknowledgements This work has been supported by the BBVA Foundation (research grant HUM14_251) and by the Spanish Ministry of Economy and Competitiveness (research grant TIN2014-52010-R). The *Chasqui* repository was created and is maintained by Prof. Mercedes Guinea (currently a researcher at "El Caño" Foundation). The thesaurus used as an example in this work is adapted from the *Chasqui*'s cataloguing schema. *Chasqui*'s original software infrastructure was developed by Alfredo Fernández-Valmayor (currently also at "El Caño" Foundation).

References

1. Berchtold, S., Böhm, C., Keim, D.-A., Kriegel, H.-P., Xiaowei, X.: Optimal multidimensional query processing using tree striping. In: Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery, pp. 244–257. Springer, London, UK (2000)
2. Chengkai, L., Ning, Y., Senjuti, B-R., Lekhendro, L., Gautam, D.: Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In: Proceedings of the 19th International World Wide Web Conference, pp. 651–660. ACM, Raleigh, NC, USA (2010)
3. Chodorow, K.: MongoDB: the definitive guide. O'Reilly (2013)
4. Cigarrán-Recuero, J., Gayoso-Cabada, J., Rodríguez-Artacho, M., Romero-López, D., Sarasa-Cabezuelo, A., Sierra, J.-L.: Assessing semantic annotation activities with formal concept analysis. Expert Syst. Appl. **44**(11), 5495–5508 (2014)
5. Culpepper, J.-S., Moffat, A.: Efficient set intersection for inverted indexing. ACM Trans. Inf. Syst. 29(1), article 1 (2010)
6. Godin, R., Saunders, G.: Lattice model of browsable data space. Inf. Sci. **40**(2), 89–116 (1986)
7. Grainger, T., Potter, T.: Solr in Action. Manning Publications (2014)
8. Greene, G.-J., Dunaiski, M., Fischer, B.: Browsing publication data using tag clouds over concept lattices constructed by key-phrase extraction. In: Proceedings of Russian and South

[6]repositorios.fdi.ucm.es/mnemosine/.

[7]repositorios.fdi.ucm.es/CIBERIA.

[8]repositorios.fdi.ucm.es/Tropos/.

[9]oda-fec.org/nata.

African Workshop on Knowledge Discovery Techniques Based on Formal Concept Analysis, pp. 10–22. CEUR, Stellenbosch, South Africa (2015)
9. Greene, G.-J., Fischer, B.: Interactive tag cloud visualization of software version control repositories. In: Proceedings of the 3rd IEEE Working Conference on Software Visualization, pp. 56–65. IEEE, Raleight, NC, USA (2015)
10. Greene, G.-J.: A Generic framework for concept-based exploration of semi-structured software engineering data. In: Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering, pp. 894–897. ACM, Lincoln, Nebraska, USA (2015)
11. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: a browser for heterogeneous semantic web repositories. In: Proceedings of the 5th International Semantic Web Conference, pp. 272–285. Springer, Athens, GA, USA (2006)
12. Huang, J.-W., Chen, K.-Y., Chen, Y.-C., Yang, K.-N., Hwang, S., Huang, W.-C.: A novel spatial tag cloud using multi-level clustering. J. Inf. Sci. Eng. **30**, 687–700 (2014)
13. Jain, A.-K., Murty, M.-N., Flynn, P.-J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
14. Kriegel H.-P.: Performance comparison of index structures for multi-key retrieval. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 186–196. ACM, Boston, MA (1984)
15. Kuznetsov, S.: On computing the size of a lattice and related decision problems. Order **18**(4), 313–321 (2001)
16. Li, R., Shenghua, B., Fei, B., Su, Z., Yu, Y.: Towards effective browsing of large scale social annotations. In: Proceedings of 16th International World Wide Web Conference, pp. 943–952. ACM, Banff, Alberta, Canada (2007)
17. McCandless, M., Hatcher, E., Gospodnetic, O.: Lucene in Action, 2nd edn. Manning Publications (2010)
18. Perugini, S.: Supporting multiple paths to objects in information hierachies: faceted classification, faceted search, and symbolic links. Inf. Proc. Manag. **46**(1), 22–43 (2010)
19. Radelaar, J., Boor, A.-J., Vandic, D., van Dam, J.-W., Fasinca, F.: Improving search and exploration in tag spaces using automated tag clustering. J. Web Eng. **13**(3–4), 277–301 (2014)
20. Sarmah, A.-K., Hazarika, S.-M., Sinha, S.-K.: Formal concept analysis: current trends and directions. Artif. Intell. Rev. **44**(1), 47–86 (2015)
21. Schraefel, M.-C., Smith, D-A., Owens, A., Russell, A., Harris, C., Wilson, M.: The evolving mSpace platform: leveraging the semantic web on the trail of the memex. In: Proceedings of the 16th Conference on Hypertext, pp. 174–183. ACM, Salzburg, Austria (2005)
22. Schraefel, M.-C., Wilson, M., Russell, A., Smith, D.-A.: mSpace: improving information access to multimedia domains with multimodal exploratory search. Commun. ACM **49**(4), 47–49 (2006)
23. Sierra, J.-L., Fernández-Valmayor, A., Guinea, M., Hernanz, H.: From research resources to learning objects: process model and virtualization experiences. Education. Tech. Soc. **9**(3), 56–68 (2006)
24. Sierra, J.-L., Fernández-Valmayor, A.: Tagging learning objects with evolving metadata schemas. In: Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies, pp. 829–833. IEEE. Santander, Spain (2008)
25. Smith, D.-A., Owens, A., Schraefel, M-C., Sinclair, P., Max, P-A., Wilson, A., Rusell, A., Martinez, K., Lewis, P.: Challenges in supporting faceted semantic browsing of multimedia collections. In: Proceedings of the 2nd International Conference on Semantics and Digitial Media Technologies, pp. 280–283. Springer, Genoa, Italy (2007)
26. Tunkelang, D.: Faceted Search. Morgan & Claypool Publishers (2009)
27. Uddin, M.-N., Janecek, P.: The implementation of faceted classification in web site searching and browsing. Online Inf. Rev. **31**(2), 218–233 (2007)
28. Way, T., Eklund, P.: Social Tagging for digital libraries using formal concept analysis. In: Proceedings of the 17th International Conference on Concept Lattices and their Applications, pp. 139–150. Sevilla, Spain (2010)

29. Wei, B., Liu, J., Zheng, Q.: A survey of faceted search. J. Web Eng. **12**(1–2), 41–64 (2013)
30. Yee, K.-P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 401–408. ACM, Fort Lauderdale, Florida, USA (2003)
31. Yitzhak, O-B., Golbandj, N., Har'El N. et al.: Beyond basic faceted search. In: Proceedings of the 2008 International Conference on Web Search and Data Minining, pp. 33–44. ACM, Stanford, CA, USA (2008)
32. Zhang, Z., Li, W., Gurrin, C., Smeaton. A.-F.: Faceted navigation for browsing large video collection. In: Proceedings of the 22nd International Conference on Multimedia Modelling, pp. 412–417. Springer, Miami, USA (2016)
33. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. **33**(2) (2006) article 6