2009

User Interface Design

Moritz Stefaner

Sebastien Ferre

Saverio Perugini
University of Dayton, sperugini1@udayton.edu

Jonathan Koren

Yi Zhang

Chapter 4
User Interface Design

Moritz Stefaner, Sébastien Ferré,
Saverio Perugini, Jonathan Koren, and Yi Zhang

"Design is not just what it looks like and feels like.
Design is how it works."

Steve Jobs, 1955–

As detailed in Chap. 1, system implementations for dynamic taxonomies and faceted search allow a wide range of query possibilities on the data. Only when these are made accessible by appropriate user interfaces, the resulting applications can support a variety of search, browsing and analysis tasks.

User interface design in this area is confronted with specific challenges. This chapter presents an overview of both established and novel principles and solutions. Based on a definition of core principles (see Sect. 4.1) and challenges (see Sect. 4.2), we define a taxonomy of navigation modes observed in existing applications (see Sect. 4.3). On that basis, design patterns for enabling these navigation modes in user interfaces (see Sect. 4.4) as well as extensions and related approaches (see Sect. 4.5) are discussed. The chapter closes with an approach to personalizing faceted search (see Sect. 4.6).

4.1 Principles

Extending traditional models of Information Retrieval, *search for digital resources* has lately been widely recognized as multi-step processes [32, 130, 179, 228]. To follow the terminology introduced in [130], a search usually involves an *initial constraint definition*, followed by an *orienteering and refinement* phase based on first inspections of the result, and finished with a closer examination of individual results in the so-called *endgame*.

In this context, the exploration of dynamic taxonomies [236] with *facet browsers* is often seen as a most promising candidates for "rich exploration of a domain across a variety of sources from a user-determined perspective" [155]. These make different aspects of the underlying data accessible in parallel. Selecting one of the values, and thus filtering the result set, restricts the available metadata values only to those occurring in the results. Consequently, the user is visually guided through an iterative process of query refinement and expansion, never encountering situations with zero results.

DOI 10.1007/978-3-642-02359-0_4, © Springer-Verlag Berlin Heidelberg 2009

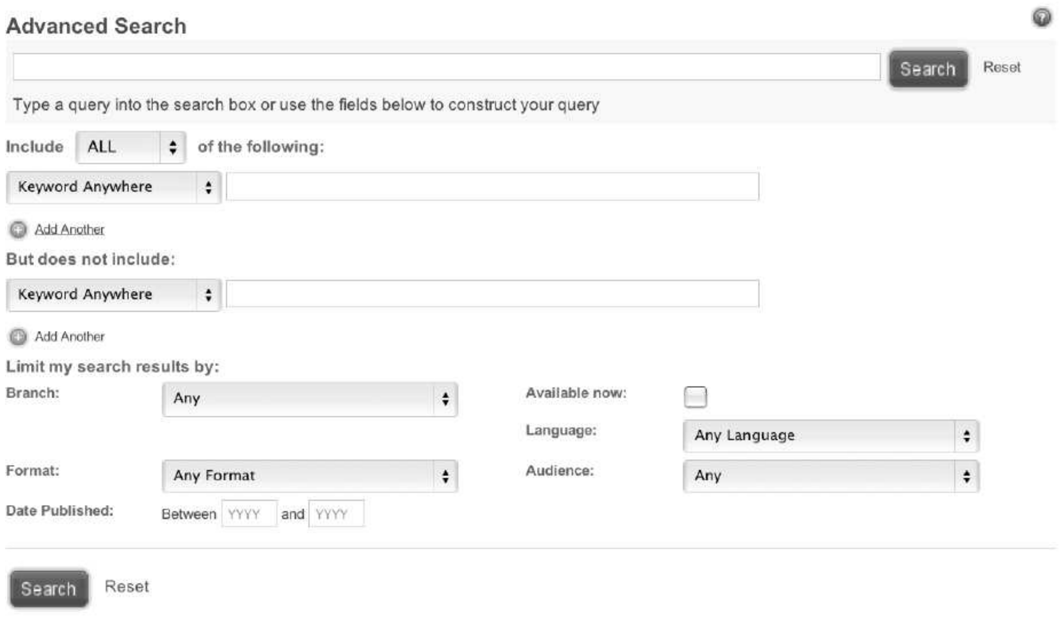


Fig. 4.1 The advanced search interface for the Oakville Public library at http://opl.bibliocommons.com/search

Applications for faceted search and dynamic taxonomies offer the following key features to support a wide range of search and browsing tasks:

- **Unrestricted query formulation over multi-dimensional classification**
  Facet browsing applications impose no restrictions, in which order, or in which granularity filters are applied on a result set. Filters stem from various, orthogonal dimensions that can be combined by Boolean operators. This allows the formulation of complex queries, such as "All documents created before date A, related to topic B, and of file type C or D". The equal treatment of multiple dimensions differs from, e.g. typical web site structures or file systems, where a single taxonomy is the pre-dominant organization principle, and other metadata are only supplements for sorting or filtering.
- **Poka yoke: no more empty result sets**
  One of the core principles of dynamic taxonomies is to restrict the available filtering options in the given focus to only those, which will lead to a non-empty result set. Hence, the user can never run into a situation with zero results. This is opposed to the process in a typical *advanced search* situation, where first a complex boolean query is constructed, which is then evaluated on demand (see e.g. Fig. 4.1). That, however, can result in empty result sets, often without further indication on which part of the query could be relaxed in order to retrieve some results. The exclusion of potentially frustrating situations by design is often referred to as poka-yoke principle.[1]

[1] See e.g. http://en.wikipedia.org/wiki/Poka-yoke.

- **Orienteering and domain understanding**

  It is a common pattern to visualize the number of occurrences of a concept in the given focus. The simplest option is to provide it after the concept label (e.g. "Europe (5)"). Advanced techniques include the application of visual indicators, such as bar height or small bar charts (see Sect. 4.4.7).

  This provides valuable *information scent* [212], i.e. "a user's (imperfect) perception of the value, cost, or access path of information sources obtained from proximal cues" [317]). Orienteering, or "directed situated navigation" [284] is the process of reaching a goal through a series of small actions, supported by continuous evaluation of the respective focus. In this context, knowing beforehand how many resources to expect after adding a concept as a filter, can be a valuable indicator of the utility of the filtering action. Additionally, this principle can be extended in order to foster domain understanding by learning about characteristic metadata distributions (see Sect. 4.4.7).

4.2 Challenges

The prototypical facet browsing application has at least two main interface areas: one for presenting facets and their values, one for displaying the result set. Additional components might include a detail view for selected resources and a breadcrumb strip for filter summary and selection history navigation (see Sect. 4.4.5).

Based on this basic setup, a number of dimensions can vary in the system and user interface design, and need to be carefully decided upon:

- Which is the data type of the different facets—nominal, hierarchical, ordinal, real valued?
- How are facet values presented to the user? Are all facets and values visible, or only a selection?
- Can the user select multiple values per facet? If so, does this result in conjunctive or disjunctive queries?

Based on these fundamental considerations in setting up a faceted navigation scheme, and designing an appropriate interface, the following recurring challenges in designing these systems will have to be tackled [131, 133, 163]:

- **Boolean query logic** A selection of single concepts from different facets is usually understood as conjunction (AND-query). If, however, multiple values within one facet are selectable (for instance, "red" and "green" from the "color" facet), depending on context and data set, either a conjunctive ("red" AND "green") or a disjunctive ("red" OR "green") interpretation are conceivable. If an application only uses one of these selection modes, this needs to be communicated to the user; if both are possible, separate controls for both modes will be needed (see Sect. 4.4.1).

- **Cluttered interfaces** The paradigm of making all filter options available in parallel naturally leads to the challenge of having to fit many controls and text fields on the user screen. Hence, clear visual structure and hierarchy as well as strategies to reduce visual clutter are vital. If a full exposure of all facets is not possible due to size constraints, strategies and user controls for showing and hiding, or expanding and collapsing facets will have to be integrated (see Sects. 4.4.2 and 4.4.3).
- **Incorporating keyword search** A free-form keyword field in order to search for arbitrary terms in addition to the pre-defined classification scheme is a "key component to successful faceted search interfaces" [131]. One source of confusion can be the question, if the search field will act as a plain text filter (e.g. searching over titles and descriptions of the resources) or if it will also match classification terms. A third conceivable option is a "search within the results", which just filters the result display, but does not act as a full-fledged facet. In either case, the relation of the free-form search to the rest of the filters has to be signalized clearly in order to avoid misconceptions (see Sect. 4.4.4).
- **Change blindness** Change blindness is a well-known psychological phenomenon [222]: a person viewing a visual scene apparently fails to detect large changes in the scene, if the change in the scene coincides with some visual disruption such as a saccade (eye movement) or a brief obscuration of the observed scene or image. This situation often occurs in web applications, where the web page briefly flashes after actions demanding a new server request. In this context, animated transitions can facilitate perception of changes in user interface design [135, 286, p. 84]. Perception of change is especially important for facet browsing, as the sudden disappearance of list items after a click can be a source for misconceptions and confusion. Besides animation, clear marking of the current focus and the resulting effects are recommended (see Sect. 4.4.6).

4.3 Navigation Modes

As a basis for comparing user interface design patterns in the next section, this section defines and illustrates different navigation modes, that enable the user to navigate the available information space by consecutively applying operators on the query.

Given an infobase over a taxonomy $(T, \leq)$ of concepts, a query is a Boolean combination of concepts. We recall the extension of such a query can be computed from the extensions of concepts by applying set operations: intersection for conjunction (*and*), union for disjunction (*or*), and complement for negation (*not*).

From this perspective, browsing an infobase consists in navigating from query to query. This is more general than defining browsing as navigating from sets of objects to sets of objects, because every query determines a set of objects, its extension, and not all navigation modes can be defined as a function from sets of objects to sets of objects. The queries are constructed by following navigation links or using interface controls. Most navigation links are provided by dynamic taxonomies, which also summarize the extension of the current query.

Based on an analysis of existing applications, we can distinguish the following navigation modes:

- *zoom-in* makes the query more specific,
- *zoom-out* makes it more general,
- *shift* replaces a part of the query by a related concept,
- *pivot* replaces the whole query by a related concept,
- *slice-and-dice* allows the disjunctive selection of multiple concepts within a facet,
- *range selection* offers the options to specify query intervals within ordinal or real value facets.

The change from a query to another query, and hence, from a focus to another focus, is defined as a *navigation link*. A navigation link is decomposed into a *selection* and a *navigation mode*. This means that a same selection can be used in different ways to reach different foci. In the simple case, a selection is a concept in the dynamic taxonomy of the current focus. In the general case, a selection is the disjunction of the concepts that are selected in the dynamic taxonomy (e.g., `France or Germany or Italy`). Controls in the interface can be activated to apply modifiers on such selections: adding negation (e.g., `not (Animal or Plant)` from the selection of `Animal` and `Plant`), replacing equalities by inequalities (e.g., `date >= 2002` from the selection of `date = 2002`). Given those selections, the above navigation modes can be reduced to only two primitive navigation modes, *zoom* and *pivot*:

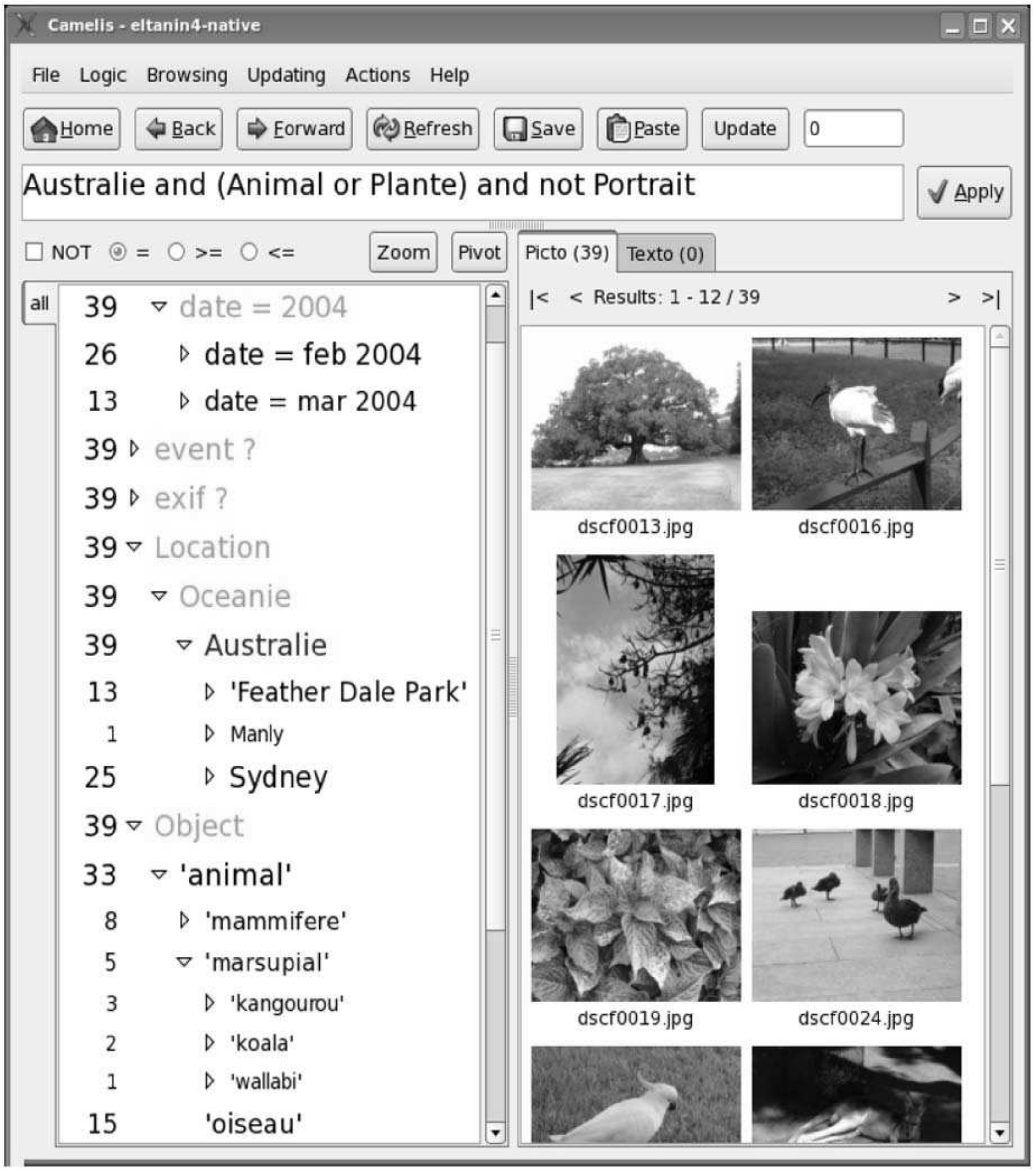
- *zoom-in* is a zoom on a selection whose extension contains some objects of the focus, but not all,
- *zoom-out* is a zoom on a selection whose extension contains all objects of the focus,
- *shift* is a combination of zoom-in and zoom-out,
- *pivot* is a basic mode,
- *slice-and-dice* is a zoom on a selection with disjunction,
- *range selection* is a zoom on a selection with inequalities.

An additional navigation mode is *querying-by-examples*, which defines the query from the selection of a set of objects, the examples.

The definitions of navigation modes rely on the fact that queries can be put in conjunctive normal form, i.e. conjunctive sets of simpler queries. For instance, `France and not date <= 2000 and (Building or Landscape)` is equivalent to {`France`, `not date <= 2000`, `Building or Landscape`}.

In the following, these interaction modes are illustrated with an example scenario using Camelis,[2] a system for browsing a personal photo collection spanning the period 1999–2007. This collection contains 5,820 photos, which are described by date, location, event, type, visible persons and objects, and EXIF descriptors (e.g., time, flash, orientation).

[2]The version used here is 1.4, and can be downloaded at http://www.irisa.fr/LIS/ferre/camelis/.


Camelis - eltanin4-native
File Logic Browsing Updating Actions Help
Home Back Forward Refresh Save Paste Update
Apply
NOT = >= <=
Zoom Pivot
Picto (39) Texto (0)
Results: 1 - 12 / 39
all

Australie and (Animal or Plante) and not Portrait

39 date = 2004
26 date = feb 2004
13 date = mar 2004
39 event ?
39 exif ?
39 Location
39 Oceanie
39 Australie
13 'Feather Dale Park'
1 Manly
25 Sydney
39 Object
33 'animal'
8 'mammifere'
5 'marsupial'
3 'kangourou'
2 'koala'
1 'wallabi'
15 'oiseau'

dscf0013.jpg dscf0016.jpg
dscf0017.jpg dscf0018.jpg
dscf0019.jpg dscf0024.jpg

Fig. 4.2 The graphical interface of CAMELIS

Figure 4.2C shows a screenshot of Camelis. The current query is at the top. The extension of this query, i.e., the current focus, is at the right, where each object is represented by a thumbnail or a text snippet depending on its type. The dynamic taxonomy is at the left, in the form of concept trees whose nodes are expanded on demand. The number at the left of each concept represent its count in the current focus, and the font scale is logarithmically proportional to this count. At the top of the dynamic taxonomy, there are check and radio buttons to modify the current selection (insertion of negation and inequalities), and two buttons for applying the two primitive navigation modes (zoom and pivot).

4.3.1 Zoom-in

First, suppose some user, say Lisa, wants to find some photos from a trip in Australia at the conference ICFCA'04. She first expands the concept `Location`, and finds she has photos from Europe (5,346), Africa (162), and Australia (148). After selecting the concept `Australia`:[3]

- the query becomes `Australia`,
- the extension displays 12 photos (out of 148),
- the concept `Australia` has now maximal font scale because all photos in the extension belong to it, and it is automatically expanded to show sub-locations of Australia (Lisa finds that she has been mainly in Sydney (105), and in the Blue Mountains (18)),
- the concepts `Europe` and `Africa` are no longer visible, because they are no longer relevant, i.e., count = 0.

Now she expands the concept `Type` and sees there are different types of photos: buildings (29), animals (34), and plants (6). She becomes interested in Australian organisms, so she selects both `Animal` and `Plant`, which leads her to the refined query `Australia and (Animal or Plant)`, whose extension contains 40 photos. One of these photos is a portrait, which Lisa does not want, so she selects the negation of `Portrait`. This leads her to the new query `Australia and (Animal or Plant) and not Portrait` (39 photos). By expanding more concepts, she discovers that these photos were taken in February and March 2004, mainly in Sydney and at the Feather Dale Park, and that 5 photos of three different species of marsupials are present: kangaroo, koala, wallaby.

Figure 4.2C shows the interface obtained after the previous navigation operations. At this stage, Lisa can either browse the 39 photos in CAMELIS, or launch a slideshow in an external application.

These three navigation steps lead to local views with increasingly more precise queries, and hence increasingly smaller extensions. This is called *zoom-in* navigation, because it corresponds to moving towards smaller extensions. Its principle is to specialize the current query q by the selection x. A simple definition of the resulting query would be q `and` x, but this would entail redundancy in queries: e.g., `Australia and Sydney` which is equivalent to `Sydney` because `Sydney` is subsumed by `Australia` in the taxonomy. A better definition is to replace by x every part of the query that subsumes x:

$$q \rightsquigarrow (q \setminus \{y \in q \mid x \leq y\}) \cup \{x\} = Min_{\leq}(q \cup \{x\}).$$

The extension of the new query is $extension(q) \cap extension(x)$. Therefore, in case the extension of x contains the extension of the query, the new query has the same

[3]There are French words in the screenshot as it is a personal photo collection, but English translations are used in the text for better consistency.

extension as the query q. So, we restrict zoom-in to selections x such that

$$extension(q) \cap extension(x) \neq extension(q),$$

i.e., to selections whose extension contains some of the objects of the focus, but not all.

4.3.2 Zoom-out

During navigation, the user may want to remove or generalize concepts in the query so as to reach larger extensions: this is the *zoom-out* navigation mode. For instance, Lisa realizes she needs more photos of animals and plants. The back button can be used to retract the previous refinement. Hence if she wants to remove the first refinement `Australia`, she needs to move three steps backwards, and then re-select the last two refinements. She could also edit the query by hand, but users usually prefer to navigate rather than to edit queries [121].

Besides, selections whose extension contains all objects in the focus, i.e. $extension(q) \subseteq extension(x)$, cannot be used for zoom-in. This makes them available for zoom-out. When such a selection is part of the query, it is removed from the query:

$$q \rightsquigarrow q \setminus \{x\}.$$

For instance, if Lisa selects `Australia`, the new query is `(Animal or Plant) and not Portrait` (282 photos from many locations). When such a selection subsumes some query parts, it replaces such parts in the query:

$$q \rightsquigarrow (q \setminus \{y \in q \mid y \leq x\}) \cup \{x\} = Max_{\leq}(q \cup \{x\}).$$

For instance, if she selects `Pacific`, the new query is `(Animal or Plant) and not Portrait and Pacific`. It is now clear how zoom-in and zoom-out can be reduced to a single primitive navigation mode. When zooming on a selection, the relationship between this selection and the current query determines whether this is a zoom-in or a zoom-out.

Compared to existing approaches, i.e., a list of removable concepts, this approach has three advantages: (1) it is integrated into the dynamic taxonomy, (2) it allows the replacement of a concept by a more general one, and (3) it extends to selections with disjunction and negation by extending the ordering $\leq$ to such selections.

4.3.3 Shift

Zoom-in and zoom-out can be combined in two forms of *shift* navigation modes. From the previous query `Australia and (Animal or Plant) and not`

`Portrait`, Lisa first chooses to zoom-in on the concept `Plant`, resulting in the query `Australia and not Portrait and Plant` (6 photos). This is her starting point for shift navigation.

At this point, Lisa sees that 1 photo has also the type `Landscape`, which interests her. She selects this concept (zoom-in) and, since the result has only 1 photo, she generalizes it by removing the concept `Plant` from the query (zoom-out). Therefore, she has executed a shift from Australian plants (6 photos) to Australian landscapes (80 photos), replacing in the query the concept `Plant` by the concept `Landscape`. From there, she performs a new shift from the concept `Landscape` to the concept `Building`, resulting in 28 photos of Australian buildings. These navigation steps are suggested and supported by photos belonging to two concepts, i.e., by extensional relations [236]. This illustrates the relevance of assigning several types to photos, which is common in this photo infobase. The same would apply to persons visible on photos, as a photo can contain several people.

However, the same does not apply to locations, as a photo cannot be taken in two incomparable locations (e.g., in Australia and in Europe). Nonetheless, it is still possible to shift between locations through the taxonomy of locations. Suppose Lisa wants to find building photos from other locations. She first generalizes `Australia` by `Location` in the query (zoom-out), and then browses suggested locations before selecting `Spain` (zoom-in). Thus, she has performed a shift from Australian buildings to Spanish buildings, and find 48 photos (mainly churches taken in the north-west of Spain in 2003).

The former form of shift is a zoom-in/zoom-out combination, and can be qualified as *extensional* because it relies on extensional relations in the infobase. The latter form of shift is a zoom-out/zoom-in combination, and can be qualified as *conceptual* because it relies on conceptual relations in the taxonomy.

4.3.4 Pivot

The user may not remember a concept she wants to use to refine the query, but she can find it through another query. For instance, suppose Lisa wants to retrieve the photos of the building of some town. She does not remember which town it is, but she remembers that the ICFCA conference took place there in 2004. Therefore, she can first reach the query `event contains "ICFCA" and date = 2004` by zoom-in navigation. The resulting extension shows photos of ICFCA'04, and the dynamic taxonomy shows relevant information about these photos, such as precise dates, locations, and so on. By browsing the dynamic taxonomy, she discovers that Sydney, in Australia, is the location of ICFCA'04. Then, she can make the query become `Sydney`, and refine it to the desired query `Sydney and Building` by zoom-in. The concept `Sydney` plays the role of a *pivot* between the two queries.

Pivot navigation relies on the ability of DTs to answer queries not only by a set of objects (the extension), but also by a set of concepts (the dynamic taxonomy). In previous navigation modes, these concepts where added or removed from the query, whereas here they are used as new queries. Given a query q and a selection x, the query transformation is defined by

$$q \rightsquigarrow x.$$

Therefore, pivot navigation is a way to restart a search from the results of a first search. This kind of navigation has already been applied in collaborative websites [189, 335].

There is an interesting analogy with natural language. Indeed, the query above can be rephrased as "photos of buildings in the town, *where* the ICFCA conference took place in 2004". The idea of pivot is reflected by the fact that Sydney occurs in the main sentence as "town", and in the relative sentence as the relative pronoun "where". The relative pronoun indicates which facet to browse for a pivot: e.g., "where" indicates a location, "when" indicates a date, and "who" indicates a person. Iterated pivot navigation then corresponds to nested relative sentences, such as "photos of buildings in the town, *where* the ICFCA conference took place in the year, *when* I also visited Hinterzarten". The first pivot to be applied is the year 2004, and the second pivot is the town Sydney.

4.3.5 Slice and Dice

Section 4.3.1 shows how the query `(France or Italy) and Building` can be reached by performing a zoom-in successively on `France or Italy` and `Building`, thus selecting French and Italian buildings. The disjunction is introduced because both concepts `France` and `Italy` were selected when the zoom mode was activated. Now suppose Lisa wants to extend the selection to landscapes, while retaining the current selection of locations. She just has to extend the selection `Building` to the selection `Building or Landscape`, and activate the zoom mode. Because the new selection is more general than the old one, the zoom is interpreted as a zoom-out. According to the definition of zoom-out (Sect. 4.3.2), the new query is `(France or Italy) and (Building or Landscape)`. Now, Lisa wants to refine the selection to Italy only. To this end, Lisa unselects `France` in the selection of locations, and applies the zoom mode. Because the new selection is more specific than the old one, the zoom is here interpreted as a zoom-in. According to the definition of zoom-in (Sect. 4.3.1), the new query is `Italy and (Building or Landscape)`. This short navigation scenario demonstrates that in a query each facet can be refined and extended independently from other facets, simply by applying the zoom mode on the new selections. In fact, it is not necessary that the different concepts in a selection belong to the same facet, while this is the most common case.

4.3.6 Range Selection

Range selection is similar to slice-and-dice (Sect. 4.3.5), except disjunctions of concepts are replaced by inequalities as selections. This makes sense because an inequality `date >= 2002` is equivalent to the infinite disjunction `date = 2002 or date = 2003 or ...`. Then, every range is the composition of two inequalities. For instance, the date range `date in [2002, 2007]` is equivalent to `date >= 2002 and date <= 2007`. Therefore every range can be reached by two successive zoom-in steps on inequalities: one for the lower bound, and the other for the upper bound. It is assumed that the user interface allows the selection of inequalities even if the dynamic taxonomies contains only values (e.g., `date = 2003`).

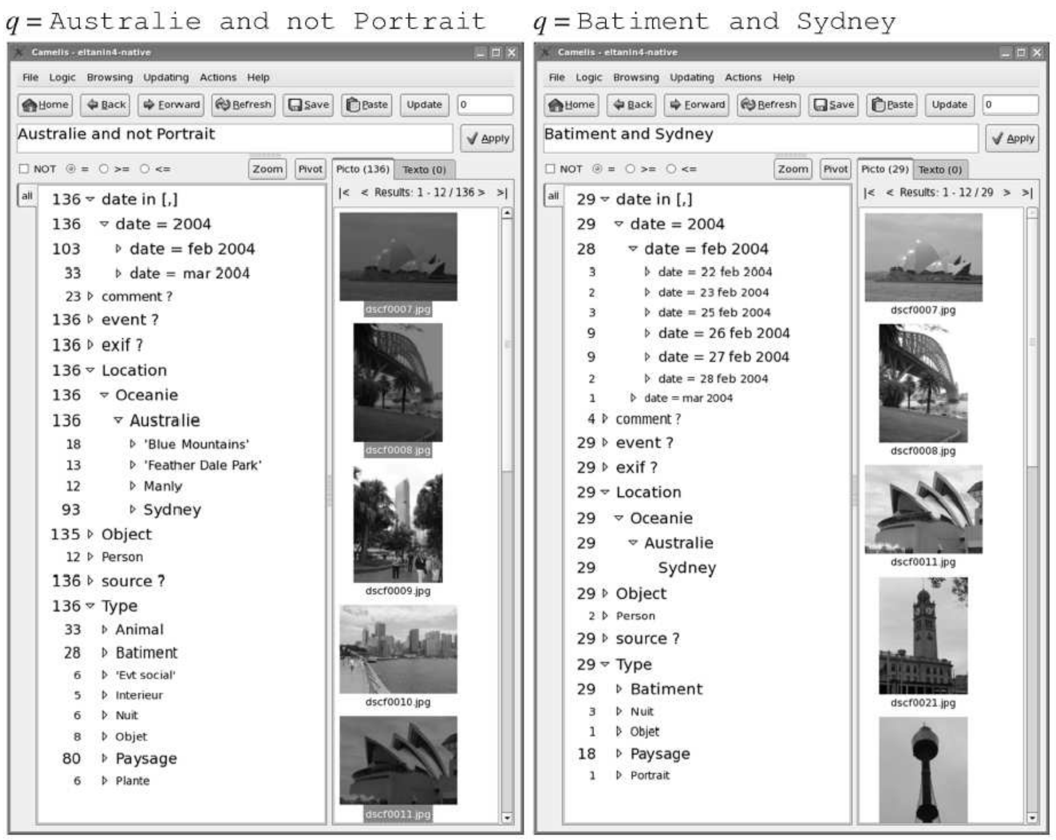
Starting from `date >= 2002 and date <= 2007 and France`, the date range can be refined by zooming on `date >= 2003` or `date <= 2006` (zoom-in), or extended by zooming on `date >= 2000` or `date <= 2008` (zoom-out). The upper bound of the range can also be removed altogether by zooming on `date <= 2007` (zoom-out); and similarly for the lower bound. The above formulas are used to give the logic of the navigation, and to reduce range selection to basic navigation modes; but a user interface may render it in a more graphical way, e.g., with the help of a double slider on a scale covering the relevant values.

4.3.7 Querying by Examples

A query can be determined by the selection of a subset of objects, thus supporting querying by examples. The idea is to construct the query as a conjunction of all most specific concepts which are shared by the selected objects O:

$$q \rightsquigarrow Min_{\leq}\{y \in T \mid O \subseteq extension(y)\}.$$

For instance, suppose Lisa starts with `Australia and not Portrait`. While browsing photos in the result, she sees interesting photos of buildings (e.g., 2 photos of the Opera, and 1 photo of the Harbour Bridge), and she would like to find more. By selecting them she moves to a new query that is the conjunction of the concepts shared by those 3 photos. As usual with this form of navigation, the resulting query is very specific and she receives no additional photos. At this stage, Lisa can use zoom-out navigation to generalize the query. Unlike approaches based on metrics, Lisa can choose which properties of the query should be generalized or removed [20]. By removing in the query concepts related to date and event, the query becomes `Sydney and Building`, and Lisa finds 29 photos. Figure 4.3 shows the three selected photos in the initial query (left side), and the resulting view of the final query (right side). From there, she can further zoom-out, zoom-in to find photos of modern buildings, or shift to find buildings in different countries. *Interactive query relaxation* [136] is similar, except that only one facet is retained in the


Camelis - eltanin4-native
File Logic Browsing Updating Actions Help
Home Back Forward Refresh Save Paste Update
NOT = >= <=
Zoom Pivot
Apply
all

$q =$ `Australie and not Portrait`

Australie and not Portrait

Picto (136) Texto (0)

|< < Results: 1 - 12 / 136 > >|

```
136 ▿ date in [,]
136   ▿ date = 2004
103     ▹ date = feb 2004
 33     ▹ date = mar 2004
 23 ▹ comment ?
136 ▹ event ?
136 ▹ exif ?
136 ▿ Location
136   ▿ Oceanie
136     ▿ Australie
 18       ▹ 'Blue Mountains'
 13       ▹ 'Feather Dale Park'
 12       ▹ Manly
 93       ▹ Sydney
135 ▹ Object
 12 ▹ Person
136 ▹ source ?
136 ▿ Type
 33   ▹ Animal
 28   ▹ Batiment
  6   ▹ 'Evt social'
  5   ▹ Interieur
  6   ▹ Nuit
  8   ▹ Objet
 80   ▹ Paysage
  6   ▹ Plante
```

dscf0007.jpg, dscf0008.jpg, dscf0009.jpg, dscf0010.jpg, dscf0011.jpg

$q =$ `Batiment and Sydney`

Batiment and Sydney

Picto (29) Texto (0)

|< < Results: 1 - 12 / 29 > >|

```
29 ▿ date in [,]
29   ▿ date = 2004
28     ▿ date = feb 2004
 3       ▹ date = 22 feb 2004
 2       ▹ date = 23 feb 2004
 3       ▹ date = 25 feb 2004
 9       ▹ date = 26 feb 2004
 9       ▹ date = 27 feb 2004
 2       ▹ date = 28 feb 2004
 1     ▹ date = mar 2004
 4 ▹ comment ?
29 ▹ event ?
29 ▹ exif ?
29 ▿ Location
29   ▿ Oceanie
29     ▿ Australie
29         Sydney
29 ▹ Object
 2 ▹ Person
29 ▹ source ?
29 ▿ Type
29   ▹ Batiment
 3   ▹ Nuit
 1   ▹ Objet
18   ▹ Paysage
 1   ▹ Portrait
```

dscf0007.jpg, dscf0008.jpg, dscf0011.jpg, dscf0021.jpg

Fig. 4.3 A screenshot of CAMELIS before and after querying by examples

generalized query. For instance, starting with the same photos, Lisa could reach the query `Sydney` or the query `Building`, but not `Sydney and Building`.

A special case of querying by examples is when selecting only one photo. Then there is only one object in the extent, because there are enough properties to uniquely characterize each photo, and the query contains all the object properties, which are more easily read in the dynamic taxonomy. So this is an easy way to access the full description of any object.

4.4 Design Patterns

This section gives an overview of solutions for solving the issues and challenges in the user interface design of applications for faceted browsing and dynamic taxonomies, with a special focus on how to enable the previously introduced navigation modes. Where applicable, these are referred to the respective user interface design patterns from established pattern libraries.

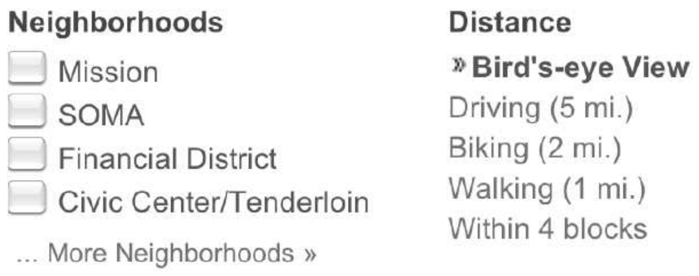


Fig. 4.4 Mixing multi-select and single-select facets in the yelp (http://yelp.com) application

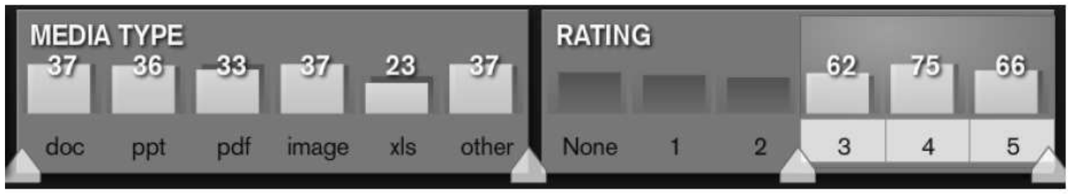


Fig. 4.5 The ContentLandscape application (see Sect. 4.5.3) combines bar chart representations with slider controls for range selection

4.4.1 Selection Management

Filter selection and de-selection is of central importance in faceted search. The basic navigation modes of *zoom-in* and *zoom-out* are present in all examined user interfaces.

If only one concept should be selectable at a time within a facet (thus avoiding the possible confusion if multiple values are to be connected by AND or OR), traditional single-select controls such as *radio buttons*, *dropdown list controls* or *simple links* (in web applications) are advisable. The standard multi-select elements, on the other hand, are *check boxes*.

Interfaces that allow only one concept selection per facet support *shift* navigation in the easiest manner, since only one click is necessary to replace a selection with another one from the same facet. If multiple selections are allowed per facet (*slice-and-dice* navigation mode), a distinction has to be made between *zooming-in*, adding the clicked value to the active filters, and *shift*, i.e. replacing the previously selected concepts from the same facet.

For instance, the yelp[4] web application provides check buttons for multi-select facets and simple links for facets with exclusive selection (see Fig. 4.4). Alternatives for allowing both modes in a facet would be dedicated controls (e.g. a "jump to" button), or modifier keys (such as pressing "shift" while clicking).

For *range selection* navigation mode, *slider controls* can allow the specification of upper and lower bounds on the result set (see Fig. 4.5C).

Additional UI functionality, however, is usually accompanied by additional complexity and visual clutter. Intelligently limiting users' options can help in allowing

[4]http://yelp.com.

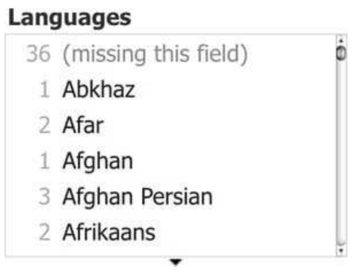


Fig. 4.6 The Exhibit (http://simile.mit.edu/exhibit) user interface signalizes missing concept assignments in a facet

the user to focus on his core tasks without additional burden of rarely used functionality. For example, for a web shop application, it might be sufficient to split the "price facet" into 3–5 discrete regions from low- over mid-priced to expensive goods, instead of giving the more fine-grained option to filter from 37 to 82.

Either way, concept de-selection should be as easy as concept selection. Additionally, if breadcrumbs or a similar filter summary indicator are present, these should include the option to clear individual filters as well. Also, buttons for resetting single facets or all filter options can help to *zoom-out* quickly.

Pivoting is usually supported not directly in the facet panels, but from the detail views for single contents. First established in Web 2.0 applications [189], it has become a common practice that a metadata value clicked in the content presentation leads to a new view with the respective value as the only selected concept. The same holds for *Querying by examples*, as this action is intrinsically related to resource instances, and not to individual facet concepts. Consequently, querying by example is usually realized with context menus or buttons adjacent to the result list presentations or detail view.

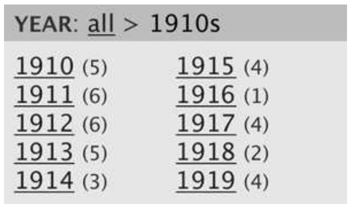
If the data is only partially tagged, it is advisable to include a "no value assigned" concept, as for instance, demonstrated in the Exhibit prototype [145] (see Fig. 4.6).

4.4.2 Revealing Hierarchy

For flat facets, i.e. not featuring a hierarchical relation between the concepts, simple list widgets are usually used. List sorting can either be alphabetical, or dynamically updated by the number of assigned items in the current result set. For navigating hierarchies, a number of different presentation and navigation options exist, which are discussed in the following.

4.4.2.1 Explorer Tree

The expandable explorer tree constitutes an established representation for hierarchical structures. This principle is, for example, used in the Camelis application (see Fig. 4.2C). Given the usually quite limited screen estate, however, the expanded lists often exceed the available facet widget space. This leads to the need for scrolling, which makes it difficult to orient in the hierarchical structure, especially if multiple levels are expanded.



YEAR: all > 1910s

1910 (5)	1915 (4)
1911 (6)	1916 (1)
1912 (6)	1917 (4)
1913 (5)	1918 (2)
1914 (3)	1919 (4)

Fig. 4.7 Zoom-and-replace and breadcrumbs in the Flamenco application

4.4.2.2 Zoom and Replace

The Flamenco application[5] [327] zooms into selected values, replacing the facet widget content with the level below the selected concept. The path from the root of the hierarchy to the current level is accessible via breadcrumbs in the header of each facet widget (see Fig. 4.7). This pattern works only for single-select facets, and optimizes for item presentation on one level. Consequently, navigation across the tree is facilitated.

4.4.2.3 Collapsible Panels

The ContentLandscape application [279] features compact, hierarchical widgets based on the accordion pattern,[6] where each hierarchy level is represented as an individual accordion level. On concept selection, the respective level is collapsed, and the subsequent level opened, to allow further drill-down in the hierarchy. Moreover, opening a level is possible by simply clicking the respective accordion header (see Fig. 4.8C).

4.4.2.4 Continuous Zooming

In the FacetZoom prototype[7] [78], hierarchical facets are displayed as space-filling widgets which allow a fast traversal across all levels while simultaneously maintaining context (see Fig. 4.9). It supports both horizontal panning for exploring a whole hierarchy level, as well as *tap-and-center* navigation, allowing to dynamically zoom-in on tree nodes. For selected concepts, the child nodes are displayed on top of the widget. Navigating one level up the tree is supported by a bottom row presentation of the parent node.

---

[5]Online demos available at http://flamenco.berkeley.edu/.

[6]See e.g. http://www.welie.com/patterns/showPattern.php?patternID=accordion.

[7]Open source version available at http://advancingusability.wordpress.com/2008/03/31/facetzoom-first-open-source-release/.

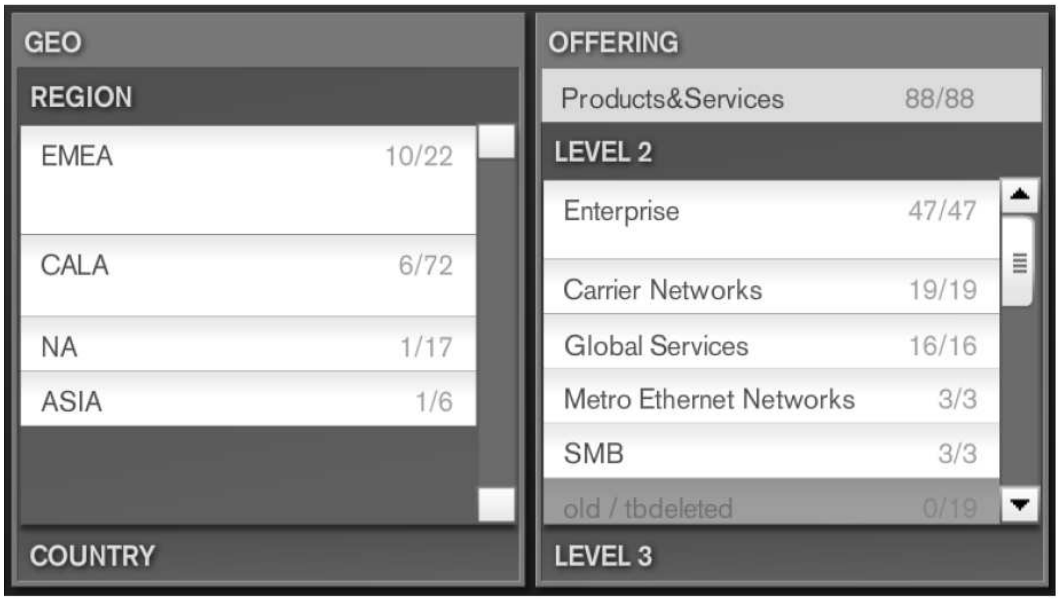


Fig. 4.8 The ContentLandscape application applies the collapsible panel pattern for zooming into concepts within a hierarchy

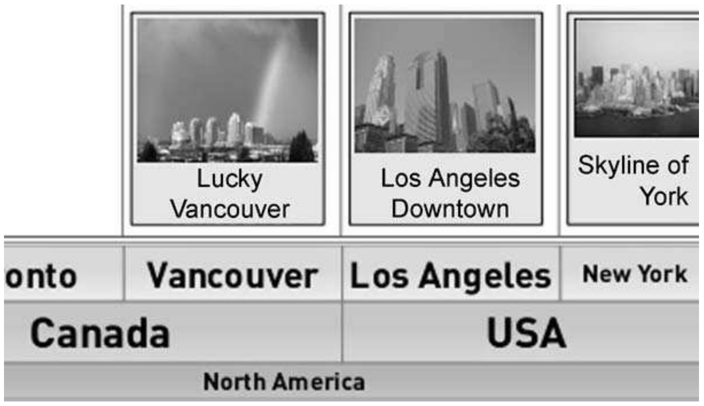
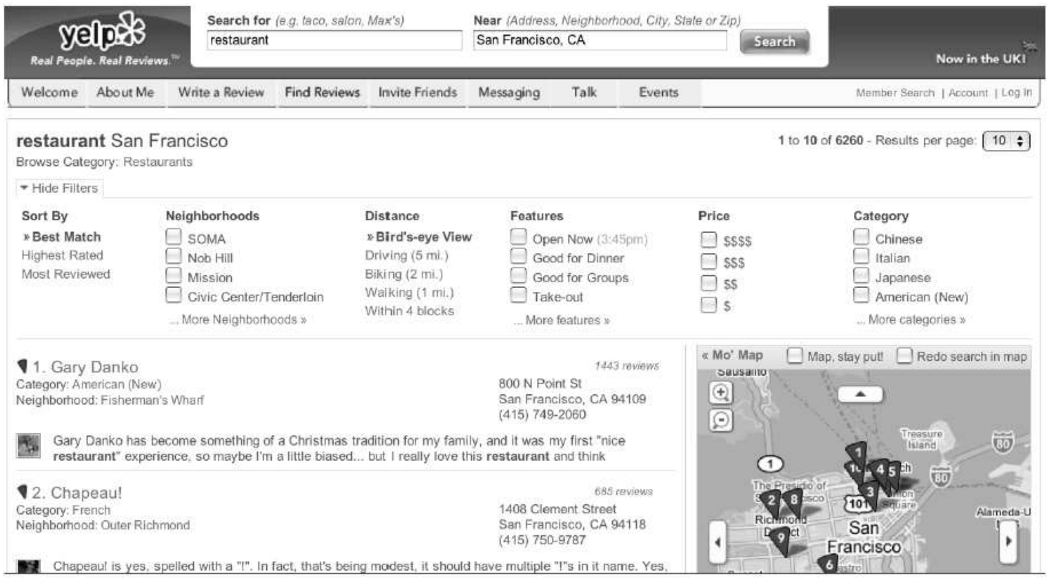


Fig. 4.9 The FacetZoom widget combines ideas from zoomable user interfaces (ZUIs) with faceted search

4.4.3 *Facet Management*

A variety of options to overcome the problem that, often, more facets are available than can be put on screen at the same time, are discussed in [131] and [133]. The options range from *collapsible facet widgets* (such as used by, for instance, Getty images' faceted navigation interface[8]) over expandable filter areas ("More ..." button) to dynamically selecting the shown facets based on the existing query (as demonstrated in the yelp application).

[8]http://gettyimages.com.



Search for (e.g. taco, salon, Max's): restaurant
Near (Address, Neighborhood, City, State or Zip): San Francisco, CA

restaurant San Francisco
Browse Category: Restaurants

1 to **10** of **6260** - Results per page: 10

Sort By	Neighborhoods	Distance	Features	Price	Category
» **Best Match**	SOMA	» **Bird's-eye View**	Open Now (3:45pm)	$$$$	Chinese
Highest Rated	Nob Hill	Driving (5 mi.)	Good for Dinner	$$$	Italian
Most Reviewed	Mission	Biking (2 mi.)	Good for Groups	$$	Japanese
	Civic Center/Tenderloin	Walking (1 mi.)	Take-out	$	American (New)
	... More Neighborhoods »	Within 4 blocks	... More features »		... More categories »

Map, stay put!
Redo search in map

1. Gary Danko — *1443 reviews*
Category: American (New)
Neighborhood: Fisherman's Wharf
800 N Point St
San Francisco, CA 94109
(415) 749-2060

Gary Danko has become something of a Christmas tradition for my family, and it was my first "nice **restaurant**" experience, so maybe I'm a little biased... but I really love this **restaurant** and think

2. Chapeau! — *685 reviews*
Category: French
Neighborhood: Outer Richmond
1408 Clement Street
San Francisco, CA 94118
(415) 750-9787

Chapeau! is yes, spelled with a "!". In fact, that's being modest, it should have multiple "!"s in it name. Yes,

Fig. 4.10 The yelp application automatically selects the presented facets based on the search term

4.4.4 Keyword Search

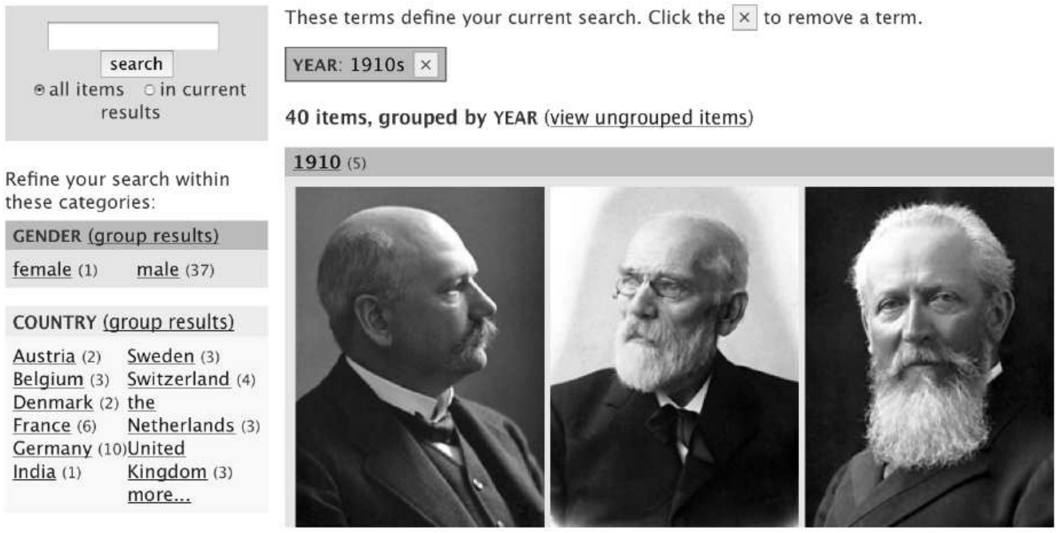
As noted above, a free-form keyword field in order to search for arbitrary terms in addition to the defined classification scheme is a "key component to successful faceted search interfaces" [131]. This task is especially challenging, since a search field can either act as a filter on the resources, e.g. searching over titles and descriptions, or if it can also match classification terms. The following sections discuss further variations within these two options.

4.4.4.1 Keyword Search as Additional Resource Filter

When search engines are enhanced with faceted search, often, a keyword search is used to define the initial result set, which can be further refined by concept selections from facets. For instance, the yelp application (see Fig. 4.10) asks for a topic (e.g. "auto repair") and a location (e.g. "San Francisco") to be entered, before entering the faceted search mode. In this application, displayed facets and concepts are selected dynamically depending on the type of query, i.e. a search for "auto repair" will yield different filtering options than one for "Chinese restaurant".

The Flamenco application [327] allows to choose between a full-text search over all results (overriding other filters) or within the current focus (see Fig. 4.11).

Obviously, when integrating keyword search with dynamic taxonomies, there might be zero hits, due to the unrestricted nature of the input. This violates the poka-yoke principle that we identified as one of the key features of applications in this domain. One solution could be to check for results after the user submits the keyword query, and leave the keyword filter in a "tentative" state if no results are


These terms define your current search. Click the × to remove a term.

YEAR: 1910s ×

40 items, grouped by YEAR (view ungrouped items)

Refine your search within these categories:

Fig. 4.11 The Flamenco application allows to choose between a full-text search over all results (overriding other filters) or within the current focus

found within the current focus. This would give the user the option to either zoom-out some other filters, or re-tract the query. In this case, it would be helpful to have a preview, of how many results could be achieved, if the respective concept would be removed from the query.

4.4.4.2 Keyword Search Within Facets

In order to avoid having to navigate large hierarchies, even though the target concept is already known by name, direct access to facet items can be achieved with a keyword search over the concept labels.

For instance, the /facet system [139], provides a keyword search box for each facet (see Fig. 4.12). This interface dynamically suggests matching concept labels after the user has typed a few characters; only keywords that produce actual results are suggested. This makes the interaction often faster than manually navigating the tree. The results are presented in a collapsible tree structure. Additionally, if the target concept is known, but it is unclear, in which facet it is located, a global search box executes the described operation over all search boxes in parallel.

A similar approach is described in [30], which is even extended to finding facets by label, and can thus be applied to very large and heterogeneous resource bases.

The ContentLandscape application [279] features a combo box component for quick access to concepts across all hierarchy levels (see Fig. 4.13C). It can be opened by clicking a search button in the facet box. In its initial state, the text field is empty, and a scrollable, alphabetical list presents all concept labels from that facet, regardless of depth level. When the user starts typing, this list is dynamically reduced to terms matching the query.

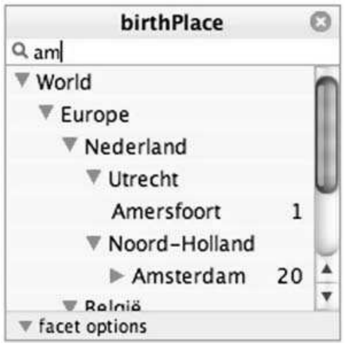


Fig. 4.12 The /facet system allows to quickly search within the concept labels of a facet

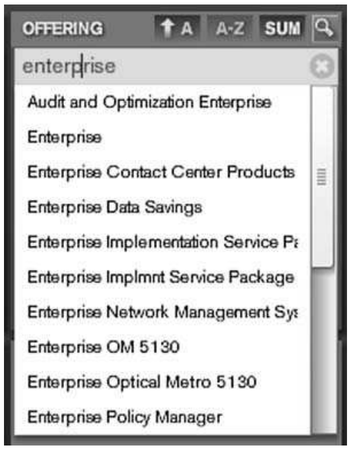

OFFERING
A
A-Z
SUM
enterprise
Audit and Optimization Enterprise
Enterprise
Enterprise Contact Center Products
Enterprise Data Savings
Enterprise Implementation Service P
Enterprise Implmnt Service Package
Enterprise Network Management Sy
Enterprise OM 5130
Enterprise Optical Metro 5130
Enterprise Policy Manager

Fig. 4.13 Quick access to concepts with a combo box in the ContentLandscape application

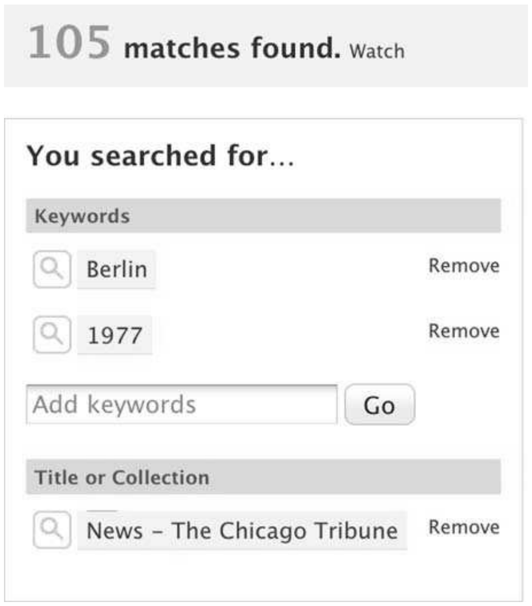
4.4.5 Filter Summary and History Navigation

Breadcrumbs can be used to summarize the current selection status in one central place in the user interface. These usually record the sequence of selection actions across all facets [131]. Breadcrumb entries should be clickable, leading to a zoom-out action on the respective concept. The footnote web site combines breadcrumbs with the option to refine with an additional keyword search (see Fig. 4.14).

4.4.6 Animated Transitions

Animated transitions can facilitate awareness of transformations and responses in user interface design [135, 291]. Perception of change is especially important for facet browsing, as the sudden disappearance of list items after click can be a source for misconceptions and confusion. In fact, studies have shown that so–called *change blindness* [222] is a common psychological phenomenon: changing details of visual scenes are often remain unnoticed, if the two states are separated by a short flash, as it is common, for example, in web applications. The Elastic Lists facet browser [278] demonstrates how smooth transitions can help in understanding filtering processes.[9]

[9]Available at http://moritz.stefaner.eu/projects/elastic-lists/.



105 matches found. Watch

You searched for...

Keywords

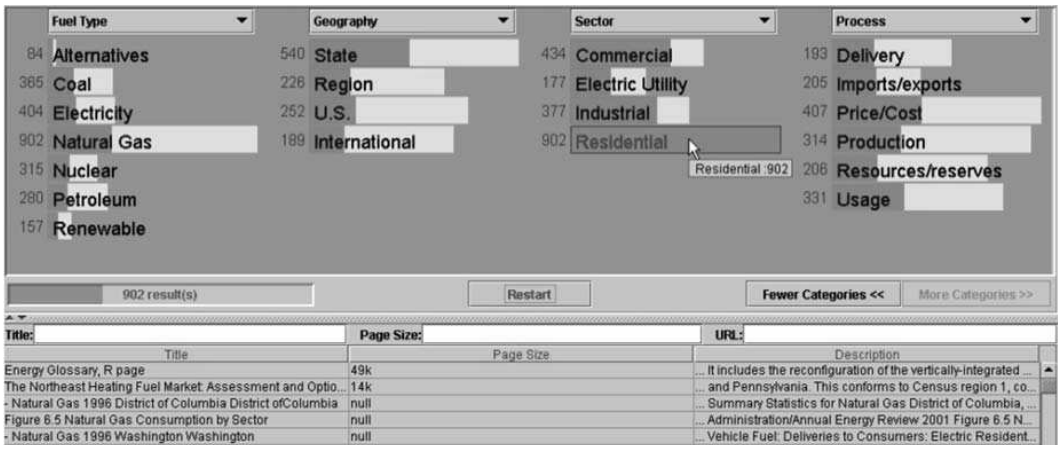
Berlin Remove

1977 Remove

Title or Collection

News – The Chicago Tribune Remove

Fig. 4.14 The footnote web site combines a filter summary with the option to refine with an additional keyword search



Title	Page Size	Description
Energy Glossary, R page	49k	... It includes the reconfiguration of the vertically-integrated ...
The Northeast Heating Fuel Market: Assessment and Optio...	14k	... and Pennsylvania. This conforms to Census region 1, co...
- Natural Gas 1996 District of Columbia District ofColumbia	null	... Summary Statistics for Natural Gas District of Columbia, ...
Figure 6.5 Natural Gas Consumption by Sector	null	... Administration/Annual Energy Review 2001 Figure 6.5 N...
- Natural Gas 1996 Washington Washington	null	... Vehicle Fuel: Deliveries to Consumers: Electric Resident...

Fig. 4.15 The RAVE system visualizes metadata value proportions in horizontal bar charts

4.4.7 Visualizing Proportions

As stated above, one common and useful technique is to exclude concepts with zero occurrences from the presented filter options, in order to avoid selections with zero results. For exploratory tasks, it can be useful to additionally see *how many* items match each of the respective concepts in the given focus.

Besides support for orienteering (see Sect. 4.1), analyzing metadata distribution can constitute a valuable information source by itself, e.g. in order to understand what makes a data set special compared to the whole collection and to generate hypothesis about the underlying reasons.

The simplest option is to provide this information in brackets after the concept label (e.g. "Europe (5)"). While this presents an economic and easy solution, the user is left with the task of processing and understanding these numbers. Visualiza-

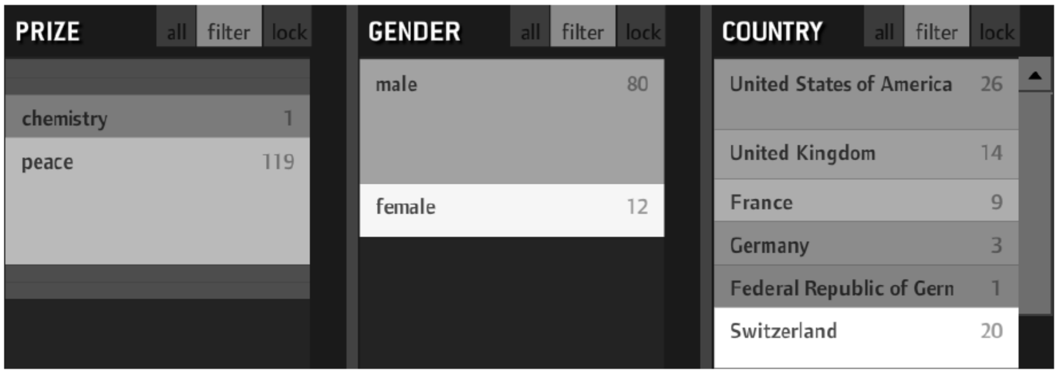


Fig. 4.16 Elastic lists indicate the number of matched resources in scaling list entry height. Additionally, unusually high proportions (compared to the global distribution) are indicated by brightness of the list entries

tion can help to make the relevant information available pre-attentively and in an intuitive manner.

For instance, the RAVE system [332] visualizes metadata proportions in horizontal bar charts, below the concept label (see Fig. 4.15). While the graphic layout could be improved in order to introduce less visual clutter, the prototype shows how additional information about local and global weights can be integrated without loss of screen estate.

In the Elastic Lists prototype [278], the height of a list item indicates the relative proportion of items associated with the respective metadata value in the given context (see Fig. 4.16). Additionally, a brighter color indicates that the current weight is significantly higher when compared to the global distribution. List entries with a weight of zero (i.e. not occurring in the current context) are collapsed to a minimal visible height.

The Visgets system [91] extends this principle by featuring a whole number of visualizations, with a weighted, coordinated brushing scheme (see Fig. 4.17C). The visualization elements include bar charts with range sliders, a map, and a tag cloud.[10] Visual representations for concepts and metadata values are scaled according to their global proportion. The coloring indicates, on the one hand, presence or absence of the respective value in the current result set. Additionally, on rollover on any concept or metadata value, more strongly associated items receive a higher opacity (*weighted brushing*).

4.5 Extensions and Related Approaches

Lately, the described principle of faceted search and dynamic taxonomies are being extended and translated to other types of search and browsing applications. This section provides some brief pointers to current research in this area, and introduces the novel principle of out-of-turn interaction.

[10]See http://www.welie.com/patterns/showPattern.php?patternID=tag-cloud.

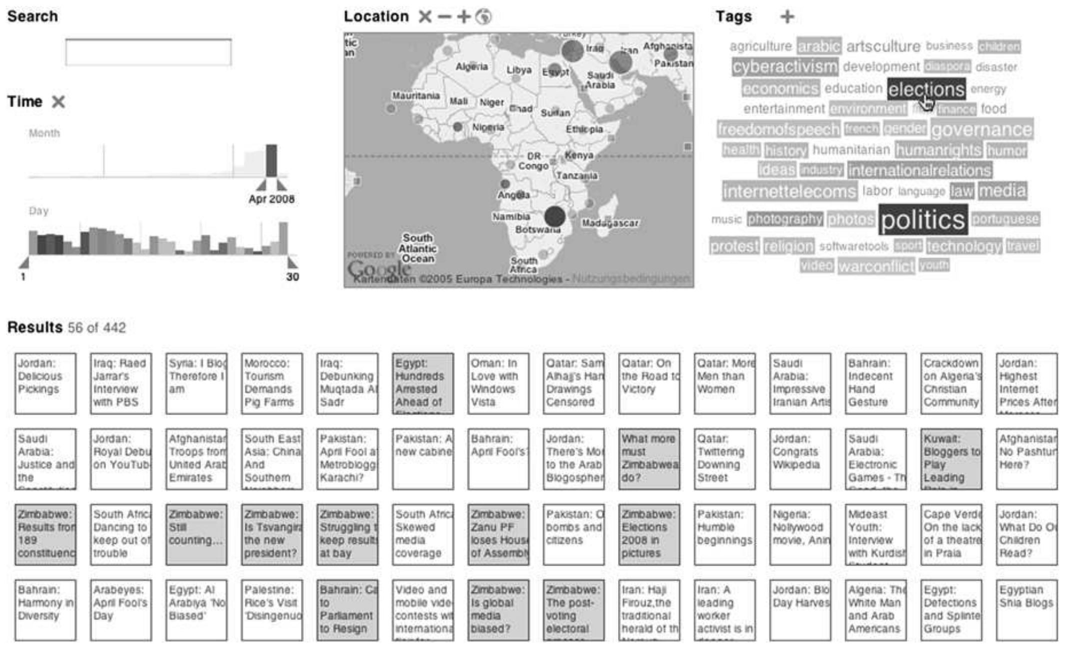

Search
Time
Month
Apr 2008
Day
1
30
Location
Tags
agriculture arabic artsculture business children
cyberactivism development diaspora disaster
economics education elections energy
entertainment environment finance food
freedomofspeech french gender governance
health history humanitarian humanrights humor
ideas industry internationalrelations
internettelecoms labor language law media
music photography photos politics portuguese
protest religion softwaretools sport technology travel
video warconflict youth
Results 56 of 442

Fig. 4.17 Weighted, coordinated brushing in the Visgets system

4.5.1 FaThumb

FaThumb [156] enables faceted search on mobile devices (see Fig. 4.18C). The filter area is grouped in nine zones, corresponding to the nine digit keys on mobile phones. The middle zone serves as a spatial overview during navigation. The surrounding eight zones allow the user to select hierarchy branches and repeatedly zoom in on subtrees. The left short shortcut key adds the currently selected concept to the query, the right one allows to quickly jump back to the top.

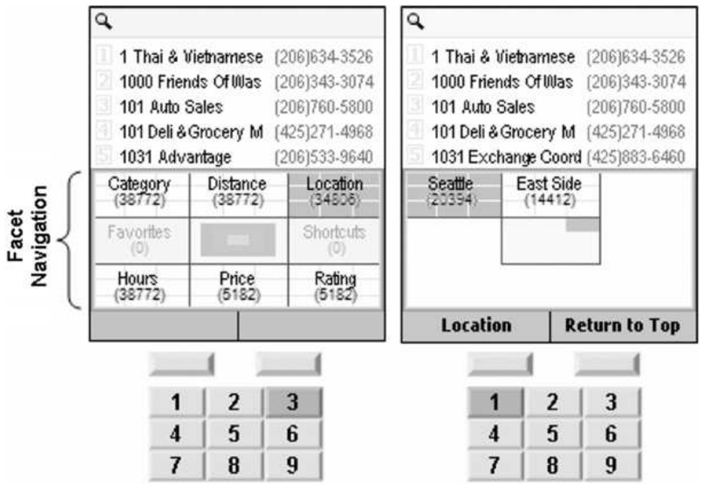
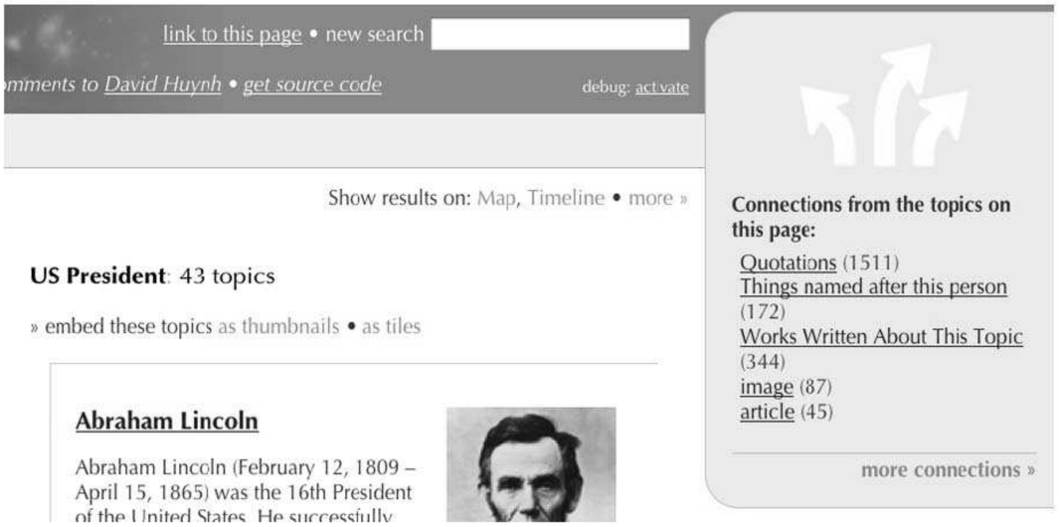


Fig. 4.18 Faceted search for small screens in the FaThumb prototype



link to this page • new search

mments to David Huynh • get source code

debug: activate

Show results on: Map, Timeline • more »

US President: 43 topics

» embed these topics as thumbnails • as tiles

Abraham Lincoln

Abraham Lincoln (February 12, 1809 – April 15, 1865) was the 16th President of the United States. He successfully

Connections from the topics on this page:

Quotations (1511)
Things named after this person (172)
Works Written About This Topic (344)
image (87)
article (45)

more connections »

Fig. 4.19 The parallax application allows to jump to related sets of items from a faceted browsing situation

4.5.2 Browsing Related Entities

Usually, the type of resource entity to be browsed (e.g. book, car, web page …) remains fixed in a faceted browsing application. In [159], a conceptual prototype of a browsing application named *Humboldt* is described, which allows to switch the type of displayed entities based on relations to the current result set. In principle, the application allows to treat any facet value space as a search result list, and arrange the interface accordingly. To cite an example given in [159], "a user who filters films on certain directors and then pivots[11] on actors will see all actors in the result list, who are related to any film in the previous result list".

This principle has also been demonstrated in the parallax application based on the freebase[12] public database [144] (see Fig. 4.19). It allows, for example, to query the data set for architects, then filter down to all modern architects: a classical zoom-in. The novel principle, however, is that the user can explore related collections, like the buildings they designed, their birth places etc. in the same facet browsing space. The jump to these new results is offered in a "connections" box on the top right of the interface. History navigation for these "related set" browsing steps is provided by a breadcrumb control.

[11]Note that the semantics of *pivoting* in this case differs from the definition introduced in Sect. 4.3.4.

[12]http://freebase.com.

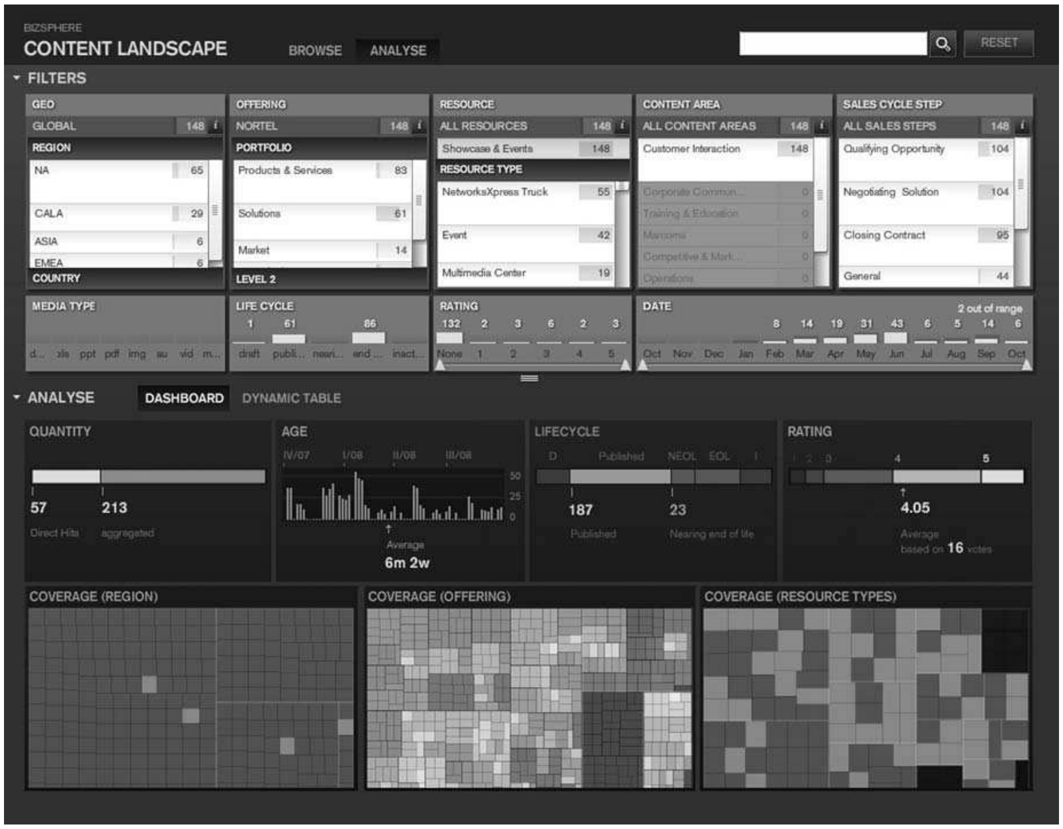


Fig. 4.20 The dashboard view of the ContentLandscape application

4.5.3 Resource Analytics

Understanding resource production, use and distribution across departments, regions, and product groups is one of the core challenges of knowledge management in the enterprise [230]. "What are the most downloaded contents?", "do the presentation materials for a given product cover all important sales regions?", "what parts of my resource collection are growing? and which are declining?" are typical questions in this area.

The ContentLandscape application[13] [279] is part of the BizSphere[14] application suite and uses faceted browsing and search in order to facilitate the understanding of resource distributions. In addition to traditional result set views, a dashboard view presents statistical measures for the resource set in the current selection (see Fig. 4.20C). It features visualizations of trend measures such as the quarter to quarter growth, a detailed age histogram, and the rating distribution. Moreover, the coverage of the selected resource set with respect to the three main taxonomies 'region', 'offering' and 'resource type' is presented in squarified treemaps [52, 270]. At first

[13]http://moritz.stefaner.eu/projects/content-landscape.

[14]http://bizsphere.com.

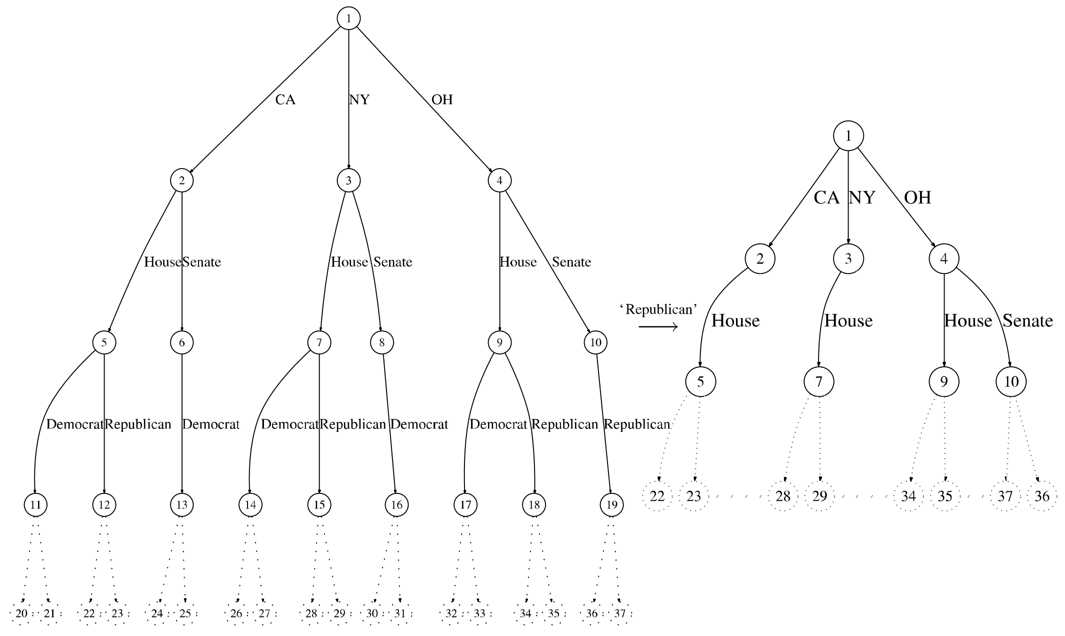


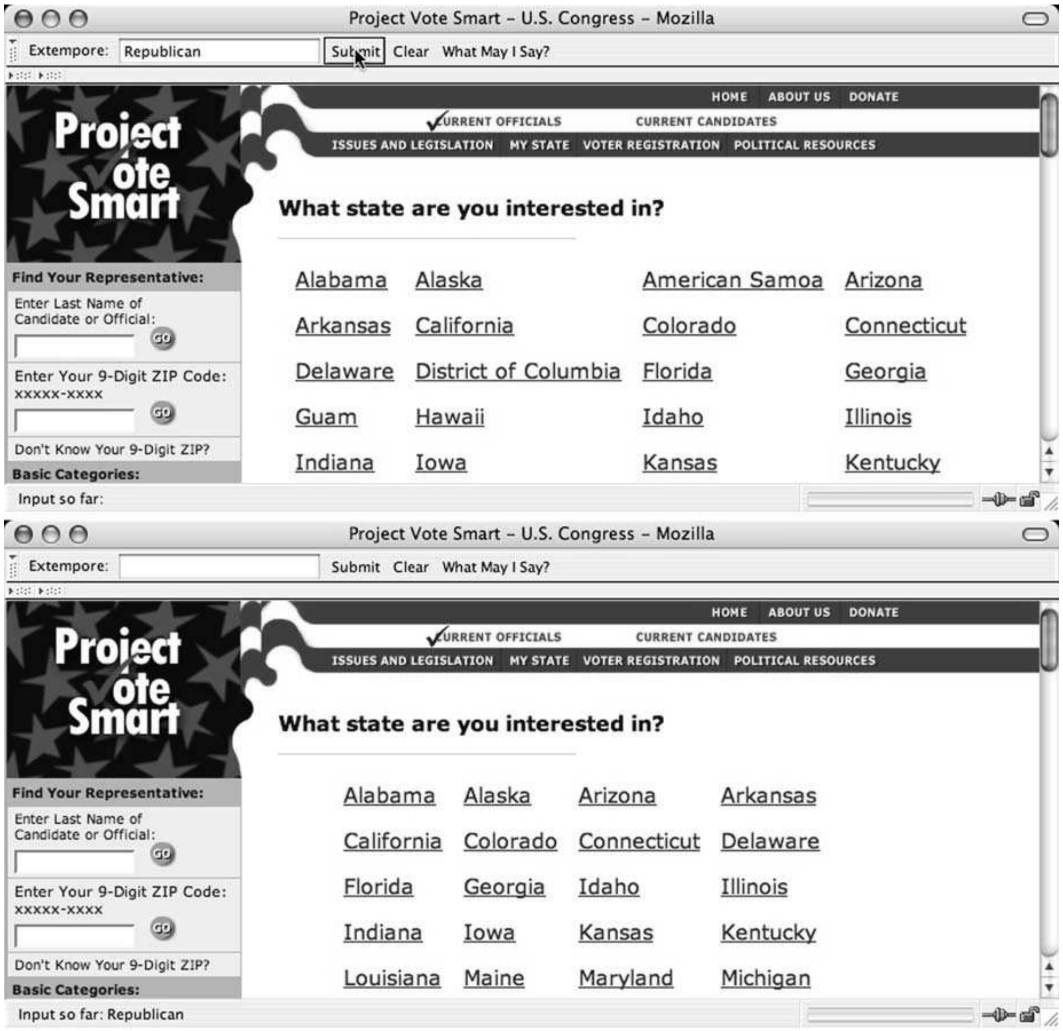
Fig. 4.21 Illustration of the pruning conducted as a result of out-of-turn interaction. (**left**) Sample hierarchy, simplified for purposes of presentation, with characteristics similar to those in PVS. (**right**) Site schema resulting from supplying 'Republican' out-of-turn

glance, this visual display allows to see, for instance, if all product groups are represented by resources in the current selection, and in which specificity. This statistical analysis can be further decomposed into in several matrix views. Inspired by OLAP approaches [70], these allow the user to split the result set according to up to three dimensions, and compare the statistical measures for the resulting sub-collections in parallel.

4.5.4 Out-of-turn Interaction

Out-of-turn interaction [195] is a technique for navigating hierarchical websites which augments traditional browsing by empowering the user to supply a hyperlink label which is presented beyond the current webpage (hence out-of-turn) to initiate a search over the site's hierarchical schema. When the system receives an out-of-turn input, it removes all paths through the site which do not contain a hyperlink labeled with the input and removes the hyperlink labeled with the input from the remaining paths. Figure 4.21 illustrates how a sample hierarchy with a structure similar to that of PVS would be pruned based on supplying 'Republican' out-of-turn. Notice that all paths leading to the webpages of Democratic politicians (nodes 20, 21, 24, 25, 26, 27, 30, 31, 32, and 33) have been removed. In addition, the hyperlinks labeled 'Republican' in the remaining paths (those leading to nodes 22, 23, 28, 29, 34, 35, 36, and 37) have been removed.

Figure 4.22 illustrates an out-of-turn interaction through a browser toolbar we call *Extempore* (as it permits the user to supply terms *extemporaneously*). Here the



Extempore: Republican

Input so far:

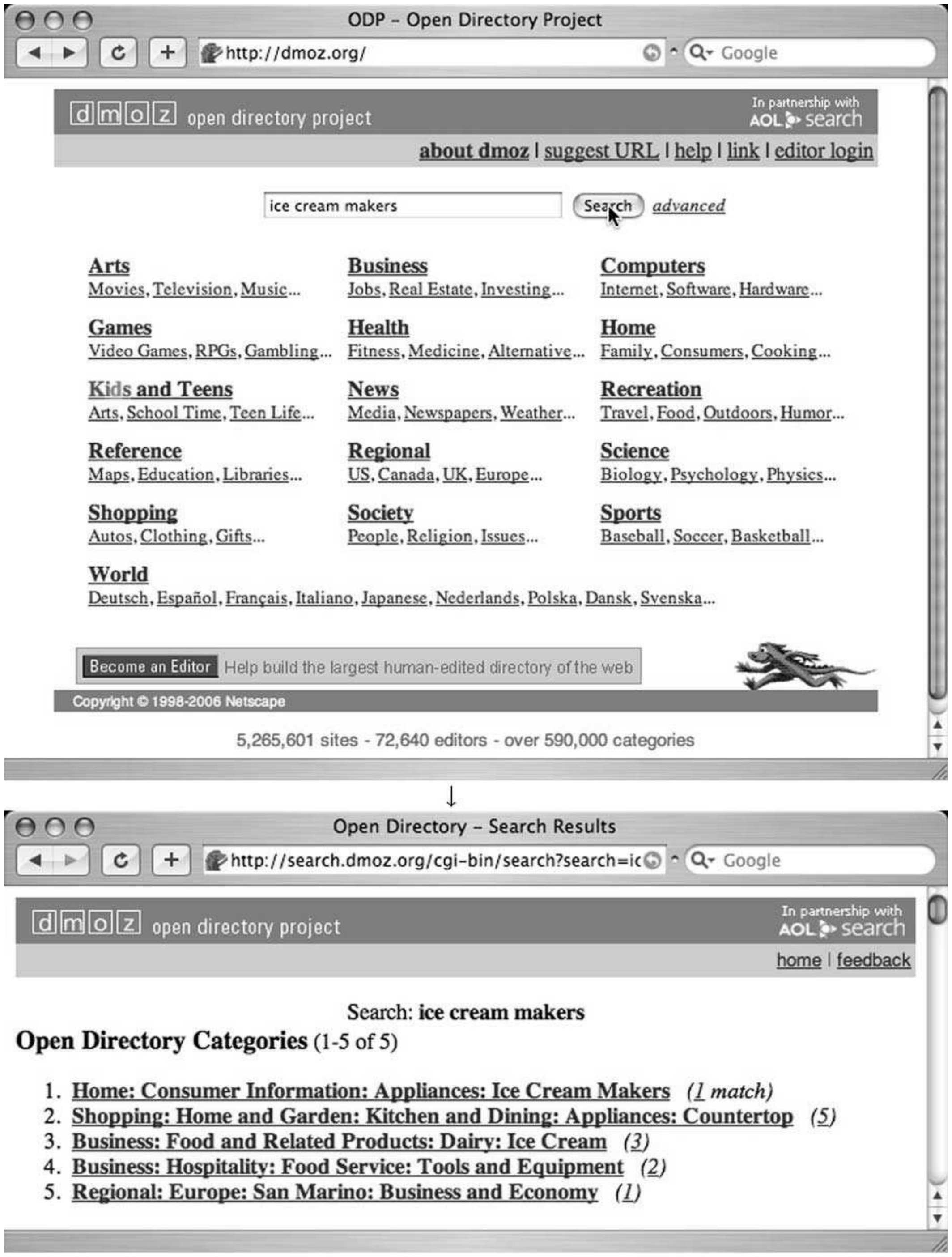
Extempore:

Input so far: Republican

Fig. 4.22 Illustration of an out-of-turn interaction with PVS through the *Extempore* toolbar

user supplies 'Republican' out-of-turn (see Fig. 4.22, top). This causes some of the hyperlinks presented on the root page (e.g., Hawaii), those which do not lead to the webpages of Republican congresspeople, to be pruned out (see Fig. 4.22, bottom). When used in conjunction with traditional browsing, the unsolicited reporting [19] involved in supplying an out-of-turn input supports a simple form of mixed-initiative interaction [211] and can be viewed as an approach to integrating querying and browsing in information hierarchies [45].

In sites where each level of the hierarchy corresponds to a facet of information assessment, such as PVS, out-of-turn interaction permits the user to explore the facets in any order without the designer enumerating all possible paths of navigation. In hierarchies where each level does not correspond to a facet, such as Yahoo! and the Open Directory Project (ODP) at dmoz.org, out-of-turn interaction behaves more as a pruning operator and reveals to the user the portions of the taxonomy pertaining to their query. For example, ODP contains 16 top-level categories and a user starting from the homepage would be hard-pressed to know that only four (Home, Shopping,



ODP – Open Directory Project

http://dmoz.org/

dmoz open directory project — In partnership with AOL search

about dmoz | suggest URL | help | link | editor login

ice cream makers — Search — advanced

Arts
Movies, Television, Music...

Business
Jobs, Real Estate, Investing...

Computers
Internet, Software, Hardware...

Games
Video Games, RPGs, Gambling...

Health
Fitness, Medicine, Alternative...

Home
Family, Consumers, Cooking...

Kids and Teens
Arts, School Time, Teen Life...

News
Media, Newspapers, Weather...

Recreation
Travel, Food, Outdoors, Humor...

Reference
Maps, Education, Libraries...

Regional
US, Canada, UK, Europe...

Science
Biology, Psychology, Physics...

Shopping
Autos, Clothing, Gifts...

Society
People, Religion, Issues...

Sports
Baseball, Soccer, Basketball...

World
Deutsch, Español, Français, Italiano, Japanese, Nederlands, Polska, Dansk, Svenska...

Become an Editor — Help build the largest human-edited directory of the web

Copyright © 1998-2006 Netscape

5,265,601 sites - 72,640 editors - over 590,000 categories

↓

Open Directory – Search Results

http://search.dmoz.org/cgi-bin/search?search=ic

dmoz open directory project — In partnership with AOL search

home | feedback

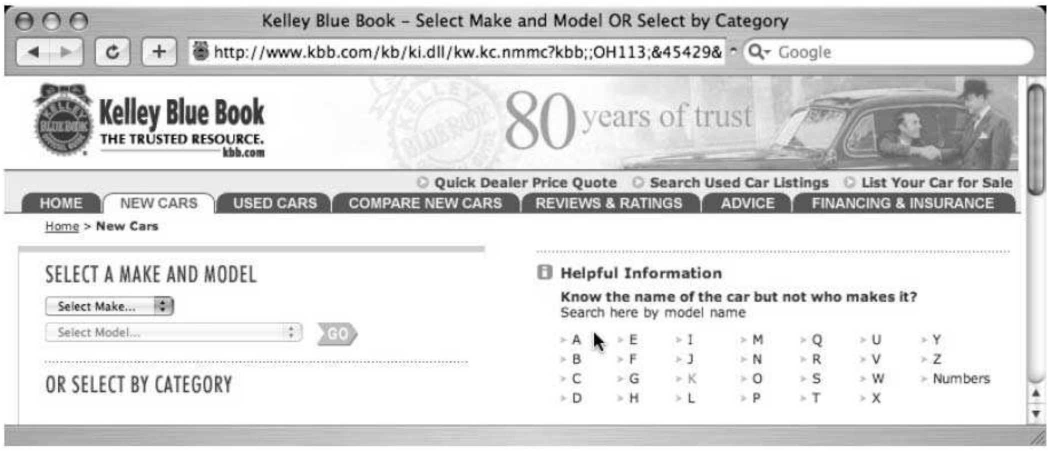
Search: **ice cream makers**

Open Directory Categories (1-5 of 5)

1. Home: Consumer Information: Appliances: Ice Cream Makers *(1 match)*
2. Shopping: Home and Garden: Kitchen and Dining: Appliances: Countertop *(5)*
3. Business: Food and Related Products: Dairy: Ice Cream *(3)*
4. Business: Hospitality: Food Service: Tools and Equipment *(2)*
5. Regional: Europe: San Marino: Business and Economy *(1)*

Fig. 4.23 (**top**) A query for 'ice cream makers' in the Open Directory Project and (**bottom**) its result as a flat list

Business, and Regional) contain links to information about 'ice cream makers'. An out-of-turn interaction reveals these categories. In fact, the search feature provided in ODP is similar to out-of-turn interaction with the exception that ODP flattens



Know the name of the car but not who makes it?
Search here by model name

Fig. 4.24 Facility for automobile-make lookup by model in the online Kelley Blue Book

the hierarchical structure in response to a query (see Fig. 4.23) whereas out-of-turn interaction preserves the hierarchical nature in order to retain context.

Notice that with the interpretation of out-of-turn interaction presented here, an unexpanded query will yield the same result as its expanded version and therefore query expansion here is simply a feedback mechanism to expose dependencies, unlike its use in traditional IR.

There are other means of exposing dependencies underlying information hierarchies during information-seeking. For example, the Kelley Blue Book (KBB) online at kbb.com provides a facility for automobile-make lookup by model (see under heading titled 'Helpful Information' in Fig. 4.24) since FDs of the form '*model* $\rightarrow$ *make*' are implicit in the domain of automobiles. When browsing new cars in KBB, users are first asked to make a selection for automobile make (see under heading titled 'SELECT A MAKE AND MODEL' in Fig. 4.24). The lookup facility allows the user to search for the make of an automobile based on the model so that they can proceed with the information-gathering dialog on the left side of the window in Fig. 4.24. A more sophisticated example of support for dependency exploration is *Sony's Advisor* facility available through sonystyle.com when browsing products such as digital cameras and camcorders.

4.6 Personalizing Faceted Search

4.6.1 Introduction

One of the primary ways users manipulate a faceted search interface is by refining their current query by clicking a facet–value pair from a list of possible system suggestions. How effective users are at finding their documents of interest is related to the quality of the query refinements suggestions. Traditionally, ad hoc approaches have been used to determine which values for a facet should be presented to the user during interaction. One common approach is to simply display all available values for a facet. While this may be effective when the number of available values is small, this approach may overwhelm users when the number too large [271]. Another approach is to display only the first few alphabetically ordered values [134]. While this approach avoids overwhelming the user with many values, it arbitrarily biases the interface towards values earlier in the alphabet. A better method is to display the most frequent values for a facet. However this method is not user centric since the most frequent values are endemic to the corpus instead of the users.

This chapter focuses on a user centric approach to determine which values are most useful to users: personalizing faceted search. Personalization allows the system to present the facet–value pairs that can help the user quickly find the document(s) that the current user is most interested. In order to determine which facet–values are most useful to a particular user, we analyze the distribution of values in corpus, and user's feedback on documents while using the system. With this knowledge, we can tailor the faceted search interfaces to individual users.

4.6.2 Related Work: Personalized Search and Filtering

The idea of personalizing search is not new in the information retrieval community [16, 41, 53, 74, 108, 140, 153, 268, 285, 289, 320]. Dumais, Cutrell, Sarin and Horvitz automatically generate queries based on keywords within an email being read or composed by a user [94]. To improve retrieval results, Bharat treats the previous information requests from the user as the context of the current query [41], whereas Shen, Tan and Zhai use the preceding queries and clicked-document summaries as the context of the current query [268]. On the other hand, researchers have developed personal information integration environments that store a particular individual's heterogeneous information and the context of the information, providing content and context-based retrieval [14, 89, 117]. Rui et al. [170] explored biasing cosine similarity methods based on user feedback in order to retrieve more documents that were similar to a user's interests. Abrol et al. personalized semi-structured search interfaces by using a user's transactional feedback from his/her queries [12]. Shen et al. used implicit user feedback, such as query refinement and click logs, to customize a KL-divergence model for document retrieval [268]. Personal WebWatcher passively observed a user's browsing behavior in order to highlight links that matched the inferred task [192].

On the other hand, personalization is a heavily studied problem in the information filtering research community and the research can be traced back to 1970s. For example, *content-based adaptive filtering* studies the scenario in which a recommendation system monitors a document stream and pushes documents that match a user profile to the corresponding user. The user may read the delivered documents and provide explicit relevance feedback, which the filtering system then uses to update the user's profile using relevance feedback retrieval models or machine learning algorithms (e.g. Boolean models, vector space models, traditional probabilistic models [229], inference networks [55], language models [75], Support Vector Machines, K-nearest neighbors clustering, neural networks, logistic regression, or Winnow [167, 324]). *Collaborative filtering* goes beyond merely using document content to recommend items to a user by leveraging information from other users with similar tastes and preferences. Memory-based heuristics and model-based approaches have been used [15, 36, 49, 85, 138, 141, 154, 160].

4.6.3 Personalization Based on Collaborative Filtering

Faceted search interfaces share three characteristics. The interfaces present a number of facets along with a selection of their associated values, any previous search results, and the current query.[15] By choosing from the suggested values of these facets, a user can interactively refine the query. The interface also provides a mechanism to remove previously chosen facets, thus widening the current search space.

In personalized faceted search, the key problem is to rank facet–value pairs according to how helpful they are for a particular user to use for query refinement. Complicating this task is the fact that rarely, if ever, does the system have access to user ratings of individual facet–value pairs. Instead, most of the existing faceted search systems, such as Amazon.com and Netflix.com, are designed so that users rate each individual document. This design trade-off is reasonable since the relevance/rating of a facet–value pair is not a well defined problem and thus hard for the user to provide. Besides, a user usually has seen a overwhelming number of individual facet–value pairs, and users rarely experience any particular facet–value pair in isolation. By rating the whole document, a user express a preference over many facet–value pairs simultaneously, especially for facet–value pairs that they may only be tangentially aware of.

Personalized faceted search is a comparatively new field that has not been well studied. Fortunately, we can develop personalized faceted search interface based on the earlier work in personalized search. For example, we can first use traditional collaborative filtering techniques to predict a user's ratings on unseen documents, and then propagate information from ratings on whole documents to individual facet–value pairs.

The basic assumption of collaborative filtering is that users that have similar preferences on some documents may also have similar preferences on other documents. Therefore the algorithm provides recommendations for a user based on the opinions of other like-minded users.

In collaborative filtering, users ratings over items are represented as a matrix A, where $A_{u,i}$ is user u's rating on item i. Many collaborative filtering techniques have

[15]For an interactive faceted search interface, the current query is the facet–value pairs the user has selected so far.

been proposed to predict the missing cells in the matrix [15, 36, 49, 85, 138, 141, 154, 160].

In this chapter, we first introduce two basic collaborative filtering techniques: K nearest neighbor and singular value decomposition. We choose the two techniques because they are commonly used, very different from each other and are complementary to each other. Variations of these two techniques have been successfully used in the Netflix movie recommendation competition [36]. Then we discuss how to go from document ratings to facet–value pair ranking.

4.6.4 K-Nearest Neighbors Based on Item–Item Similarity or User–User Similarity

There are two very commonly used collaborative filtering approaches: the first one compares each user to the other users, while the second one compares the items to each other. These are called *user–user similarity* and *item–item similarity* respectively. We describe Konstan et al.'s Pearson's correlation based user–user algorithm [160] below.

For each user, we calculate the average rating assigned by that user $\bar{A}_u$ to all rated items. Each unknown rating is then estimated as the user's average rating, perturbed by sum of the difference between every other user's assigned rating and his/her average rating, weighted by the correlation among the commonly rated items of the current user to every other user.

Formally, this is stated:

$$A_{u,i} = \bar{A}_u + \sum_{v=1}^{k} \frac{w_{u,v}(A_{v,i} - \bar{A}_v)}{|w_{u,v}|} \tag{4.1}$$

where k is the number of users that have at least one rated item in common with user u, and $w_{u,v}$ is the Pearson's correlation between user u's ratings and user v's ratings. Recall that Pearson's correlation is:

$$\begin{aligned} w_{u,v} &= \frac{\sum_{j=1}^{m}((A_{v,j} - \bar{A}_v)(A_{u,j} - \bar{A}_u))}{\sigma_v \sigma_u} \\ &= \frac{\sum_{j=1}^{m}((A_{v,j} - \bar{A}_v)(A_{u,j} - \bar{A}_u))}{\sqrt{(\sum_{j=1}^{m}(A_{v,j} - \bar{A}_v)^2)(\sum_{j=1}^{m}(A_{u,j} - \bar{A}_u)^2)}} \end{aligned} \tag{4.2}$$

where σ_u and σ_v are the standard deviations in user u's and user v's ratings, and m is the number items that user u and user v have both rated.

Herlocker et al. [138] examined using other similarity methods such as Spearman's correlation, information entropy, mean-squared difference, and found they performed similar to Pearson's correlation.

In its simplest form, the item-item algorithm is similar to the user-user algorithm, only with the rows and columns exchanged. Item-item similarity can be extended to take into account the content of the items being rated. For example, Sawar et al. [263] estimated ratings by summing the ratings of the other rated items, weighted by the cosine similarity of the rated and unrated plain text documents.

4.6.5 Singular Value Decomposition

One problem with the collaborative filtering techniques discussed in Sect. 4.6.4 is that the user–item matrix can become very sparse as the number of items and users increase. This sparseness can sometimes lead to poor predictions. By creating a low dimensional approximation of the rating matrix, it is possibly to improve the accuracy of the predicted ratings. One such technique that has been used successfully in the personalization and collaborative filtering domains is *singular value decomposition* (SVD) [36, 113, 209, 262, 264].

SVD, also known as *latent semantic indexing* (LSI) in the information retrieval community [83], works by combining rows and columns that are found to be strong correspondence. These correspondences are called *latent factors*.

Applying SVD to collaborative filtering task, we factor the $m \times n$ user–item matrix A into three smaller matrices U, Σ, and V. U is $m \times h$, Σ is an $h \times k$ diagonal matrix, and V is $n \times h$ matrix.

$$A \simeq U \Sigma V^T \tag{4.3}$$

The values along the main diagonal of Σ are the biggest h *singular values* of A in decreasing order. Each row of U and V contain orthogonal *singular vectors*. The vectors in U are known as the *left singular vectors*, while the vectors in V are the *right singular vectors*. Since we want a low dimensional approximation of the rating matrix, only the first $h < rank(A)$ singular values and singular vectors are used.

Typically, A is approximated as the product of two matrices:

$$A \approx \left(U\left(\sqrt{\Sigma}\right)^T\right)\left(\left(\sqrt{\Sigma}\right)^T V\right) \tag{4.4}$$

We can view the first matrix as the hidden representation of the users, and the second matrix as the hidden representation of the items. With this approximation, the predicted rating for a user u on item i can be calculated as $A_{u,i} = \bar{A}_u + (U(\sqrt{\Sigma})^T)_u^T((\sqrt{\Sigma})^T V)_i$.

4.6.5.1 Recommending Facet–Value Pairs

There are many methods to propagate information from ratings on whole documents to individual facet–value pairs. One method is to assign facet–value pair x_i a score based on the expected rating a user u gives to a document d containing that facet–value pair:

$$\begin{aligned} f(u, x_i) &= E[R_u(d)|x_i \in d] \\ &= \sum_{d \in D} R_u(d) P(x_i|d) \\ &\simeq \frac{\sum_{d \in D} R_u(d) \mathrm{I}_{x_i \in d}}{\sum_{d \in D} \mathrm{I}_{x_i \in d}} \end{aligned} \tag{4.5}$$

where D is the set of documents selected by the current query, $R_u(d)$ is the rating of user u on document d, and I is the indicator function.

One simple way to suggest facet–value pairs for query refinement is presenting the top scoring facet–value pairs that are contained in the documents selected by the current query. While this is an attractive option, this approach can suffer when the top scoring values are redundant, where an extreme case is that if they co-occur in the same documents. One elegant solution proposed by Chen and Karger [63] is to condition each suggestion on the assumption that none of the previous suggestions are relevant to the current query. For our purposes, this means that the $(k + 1)$th suggestion is the top scoring value that is not contained in a document that contains any of the previous k suggested values.

To determine the order that the facets are presented to each user, a simple approach is using the average score of the suggested values for each facet for the null query and then fix order of the facets throughout the lifetime of the user interaction session(s).

4.6.6 Personalization Using Content Based Filtering

In this section we provide a complementary approach to personalization based on the content of the documents. Motivated by relevance feedback retrieval models and content based adaptive filtering techniques, we focus on two statistical models: a model of the documents being searched, and a model of a user.

4.6.6.1 Document Model

While every faceted document has a set of facets associated with it, the number of values that each facet has in a particular document can vary a lot. We model this by expressing the number of values each facet has in a random document as a draw from multivariate Normal distribution:

$$\langle n_1, \ldots, n_K \rangle \sim MVN(\vec{\mu}, \Sigma) \tag{4.6}$$

where n_k contains the number of values for facet k in the document, $\vec{\mu}$ is a K-dimensional vector containing the mean number values for each facet, and Σ is the corresponding $K \times K$ covariance matrix.

Each facet in a document has a certain semantic meaning that dictates the type and thus the probability distribution of the values that can be associated with it. Five common types of facets are: *nominal*, *ordinal*, *interval*, *ratio*, and *free-text*. Nominal facets take discrete and orderless values. The values to this type of facet can be modeled as draws from a multivariate Bernoulli distribution. Ordinal facets also take discrete values, but there is an implicit ordering to these values. An example of this would be field that identified the sensitivity of a document as being for "full release", "limited release", or "secret". Interval facets can take any value on a defined range, as long as the range excludes an explicit zero point. Ratio typed facets on the other hand can take zero as a value. Values for ordinal, interval, and ratio facets can be modeled as draws from a normal distribution. Free-text facets allow arbitrary text to be associated with documents. Traditional statistical information retrieval techniques represent each word in unstructured text as a draw from a multinomial distribution.

After identifying the type and the proper probability distribution over the values associated with facet, the probability of a document existing is simply the product of the probability of each desired value occurring for the appropriate facet.

4.6.6.2 User Model

Instead of estimating a user rating for a document, in this approach we estimate the probability of a document being relevant to a particular user. Similarly, instead of calculating a suitability score for a facet–value pair to a user, we estimate the probability of a particular facet–value pair appearing in a document relevant to a user. For simplicity and without lose of generality, we assume that a document is either *relevant* or *nonrelevant* to a user. From this we can estimate the probability that any document will be relevant to a particular user, and the probability that a particular facet–value pair x_k will be contained in a relevant or a nonrelevant document. This tuple is the user relevance model and is represented as:

$$\theta_u = \{\mathrm{P}(rel \mid u), \mathrm{P}(x_k \mid rel, u)\mathrm{P}(x_k \mid non, u)\} \tag{4.7}$$

where $k = 1, \ldots, K$.

These individual probabilities can be estimated from training data. For example, assume for a particular user u, there exists a set of training data $D_u = \langle D_{u,rel}, D_{u,non}\rangle$, where $D_{u,rel}$ is the set of documents marked by u as being relevant, and $D_{u,non}$ are the set of documents marked as nonrelevant. If the facet type is free text, the maximum likelihood estimation of θ_u is:

$$\mathrm{P}(rel \mid u) = \frac{|D_{u,rel}|}{|D_u|} \tag{4.8}$$

Table 4.1 Facet types and distributions

Facet type	Values	Example	Distribution	Prior
Nominal	Unique tokens	Director	Multivariate Bernoulli	Multivariate Gamma
Ordinal	Repeatable ordered tokens	Critic's rating (e.g. A, B, C, ...)	Normal	Normal
Interval	Repeatable numbers excluding zero	Year of release	Normal	Normal
Ratio	Repeatable numbers including zero	Running time	Normal	Normal
Free-text	Repeatable tokens	Synopsis	Multinomial	Dirichlet

$$\mathrm{P}(x_k \mid rel, u) = \frac{1}{|D_{u,rel}|} \sum_{d \in D_{u,rel}} \mathrm{I}_{x_k \in d} \tag{4.9}$$

$$\mathrm{P}(x_k \mid non, u) = \frac{1}{|D_{u,non}|} \sum_{d \in D_{u,non}} \mathrm{I}_{x_k \in d} \tag{4.10}$$

This setup is very similar to the commonly used relevance language model in information retrieval [331]. We leave it as an exercise for the readers to derive the estimation for other facet types.

As stated in Sect. 4.6.1, it may take a while before enough user specific feedback information is gathered from a particular user, thus user could suffer from the so called "cold start" problem. To handle this problem and get a level of acceptable performance from the very beginning, a hierarchical Bayesian model is used and found success in user modeling experiments [328, 334, 336]. In this model, each individual user model is considered as a draw from a prior distribution that is common to all users. By using common prior, gaps in a particular user's model can be filled in by using information from the community of users. To applying the hierarchical Bayesian modeling approach for personalized faceted search, each distribution for each facet needs a separate prior that is estimated from the training data of all users in the system. Table 4.1 suggests different priors for different facet types.

Based on document models and user models described above, one can generate faceted search interface in various ways. For example, we can rank the facet–value pair based on (4.9).

4.6.7 An Ontological Approach

An alternative approach to personalize faceted search is using ontology created manually or automatically. Tvarožek and Bieliková [290] use the distance between values in a hierarchical ontology to measure similarity, and thus relevance to users.

In cases where an explicit ontology does not exist, one can be automatically constructed [42, 282, 288, 323].

The technique works as follows: Let $L_u(x)$ be the relevance of a facet x to user $u \in \mathcal{U}$ as computed by the ontological similarity of the facet and the user model [22]. In this case, the ontological similarity is the reciprocal of the maximum number of links needed to traverse from each value being compared, to a value common to both.

Each user model is then compared to the other users in the system, in order to calculate the *cross relevance* of a facet x to a user. This value, $C_u(x)$, is the average of the relevance of the facet to each user, weighted by the similarity of the each user v to the current user u.

$$C_u(x) = \frac{\sum_{v \in \mathcal{U}} \mathit{similarity}(u, v) L_v(x)}{|\mathcal{U}|} \tag{4.11}$$

The *global relevance* of a facet, $G(x)$, is calculated as the average relevance of the facet to every user. The *static relevance* of facet to a user is a weighted combination of the cross relevance and the global relevance.

In order to make recommendation to a user regarding a specific query, the temporary in-session relevance of each facet x to the current user u is introduced. This value, $T_u(x)$, is simply the fraction of query refinements (i.e. clicks) that utilize a facet x of all refinements in the current search session. This number is combined with the static relevance of the facet to determine the current *dynamic relevance* of a facet to the user. Values from facets with the highest dynamic relevance are then suggested to the user for query refinement. Evaluated with the Factic system, the ontology based approach reduce the number actions required for users to find their documents of interest when compared to an un-personalized baseline [290].

4.6.8 Evaluation Regime

Considering various personalized faceted search techniques, which one works better on a particular task? To compare personalization methods, an evaluation metric is needed. Traditionally user studies have been used to determine satisfaction with different user interfaces. While undeniably useful, user studies have some drawbacks. First, they are expensive to hold. A number of users must be gathered and then tested on the proposed system. This takes a nontrivial amount of time for even user studies with a moderate number of subjects. User studies also problematic when being used to evaluate personalized systems, as the test subjects may not interact with the system long enough for a sufficient user profile to be learned. This can lead to inconclusive, or possibly even contradictory results.

Koren et al. [162] proposed a complementary inexpensive evaluation metric based on calculating the expected utility to a user of a faceted search interface, through the use of simulated user interactions. This method allows designers to quickly compare various algorithms and determine which algorithms are the most promising. By using this method, or a similar one, designers can conduct fewer user studies by only submitting the top performing algorithms for an in depth user study. A similar approach has seen success when evaluating spoken dialog systems [118, 166].

The evaluation system works as follows: Assume that the goal of the search interface is to enable users to find their documents of interest with the least amount of effort. In order to measure this effort, the actions that a user can perform when interacting with the system are identified and the system is rewarded or penalized depending on what action is performed. A series of user interactions are simulated using a combination of real-user feedback and heuristics. The interface is then scored according to the expected total reward for an interaction session.

Given S user interaction sessions, the empirical utility of the interface can be estimated easily:

$$U = \sum_{s}^{S} \sum_{t}^{T_s} R(q_{s,t+1}, a_{s,t}, q_{s,t}) \tag{4.12}$$

where $R(q_{s,t+1}, a_{s,t}, q_{s,t})$ is the reward the system receives if the user s takes an action $a_{s,t}$ at time t, which changes the query from $q_{s,t}$ to $q_{s,t+1}$.

Let us now define the reward function. As stated earlier, we assume that the goal of the interface is to allow users with the least amount of effort on their part. In order to measure this effort, the designer must identify which actions a user can perform at each step of the interaction. Koren et al. identified eight actions common to many personalized faceted search interfaces, along with the rewards the system receives for each. These actions are listed in Table 4.2.

Instead of employing real users through a user study, actions are simulated based on certain assumptions about how real users interact with faceted search systems. Without loss of generality, it is assumed that each simulated user is searching for exactly one *target document* and that the simulated user can recognize the document and the facet–value pairs that are indicative of that document. At each step of the search session, the simulated user scans the top ranked documents that match current query looking for the target document. If it is found, then it is selected and the

Table 4.2 User actions and rewards

Action	Reward
Select facet–value pair	negative
De-select user selected facet–value pair	zero
De-select system selected facet–value pair	negative
View more facet–value pairs	negative
Mark document as relevant	positive
Mark document as non-relevant	negative
View more documents	negative
End session	zero

session ends. If the target document is not found, the simulated user removes any facet–value pairs that are contained in the current query that do not match the target document. Once the query is cleaned of incorrect terms, the simulated user scans the list of presented facet–value pairs. A pair is selected by some method for inclusion in the query from the set of pairs that match the target document. If none of the suggested facet–value pairs match the target document, then a facet is chosen at random and all of its values are examined until a matching facet–value pair is found for inclusion. If no matching value can be found for any facet, then the simulated user scans through the complete list of returned documents until the target document is found.

When there are multiple matching facet–value pairs, deciding which one to include in the query can greatly impact how much additional searching is required to find the target document. Koren et al. suggested four possible selection methods. *Stochastic users* simply select one of the matching facet–value pairs randomly from a uniform distribution. *First-match users* scan the list from top to bottom, and select the first matching facet–value pair found. This heuristic is modeled after how users select matching documents from a ranked list of results. *Myopic users* select the matching facet–value pair that is contained in the least number of documents. With this method assumes that users are trying to reduce the search space as quickly as possible. *Optimal users* that perform actions that directly optimize the utility of interface were also identified, but not examined in detail due to the complexity in searching for the optimal policy for the user to execute.

Although the simulated users differ from real users, the evaluation methodology does provide insight into understanding how various faceted interface design algorithms perform [162]. This evaluation method is neither better or worse than user studies. Instead, the approach serves to complement user studies by being cheap, repeatable, and controllable.

4.6.9 Conclusions

This section presented the problem of determining which facet–value pairs the system interface should provide to a user for query refinement. In particular, we focus on personalized faceted search techniques that try to find facet–value pairs most useful to individual users. We introduced three major approaches, collaborative filtering based faceted search personalization, content based personalization, and ontology based personalization. We present a utility based evaluation framework for various faceted search interfaces, and the general idea is that the best interface should minimize the number/cost of interactions needed to find a document of interest. This general evaluation framework can applied to all kinds of facets, including facets whose values are organized as taxonomies.