Faceted User Interfaces

Group 5 Mathias Blum, Matthias Frey, Daniel Lamprecht, and Nelson Silva

706.041 Information Architecture and Web Usability WS 2015
Graz University of Technology

9 Dec 2015

Abstract

Faceted search user interfaces support users of information systems in the process of searching and exploring information spaces. Thanks to their ease of use, they have become one of the most important strategies used to guide users through large item collections. Examples of faceted search interfaces can be found on a variety of Web sites, ranging from libraries, shopping sites to big data exploration and visualization projects. This survey presents an overview of the state of the art of faceted user interfaces that help users in the exploratory process of searching and finding the desired piece of information. After a brief discussion of the theoretical background of faceted user interfaces, this survey is made up of two parts: Part one introduces a survey of current Web sites making use of faceted interfaces and classifies them based on a range of properties. This classification aims at helping to better understand the topic of faceted user interfaces and categorizes the different factors that need be taken in consideration when implementing a faceted user interface. Part two gives an overview of the state of the art of the technologies that can be used to setup a powerful faceted user interface and surveys both open source and commercial solutions.

Contents

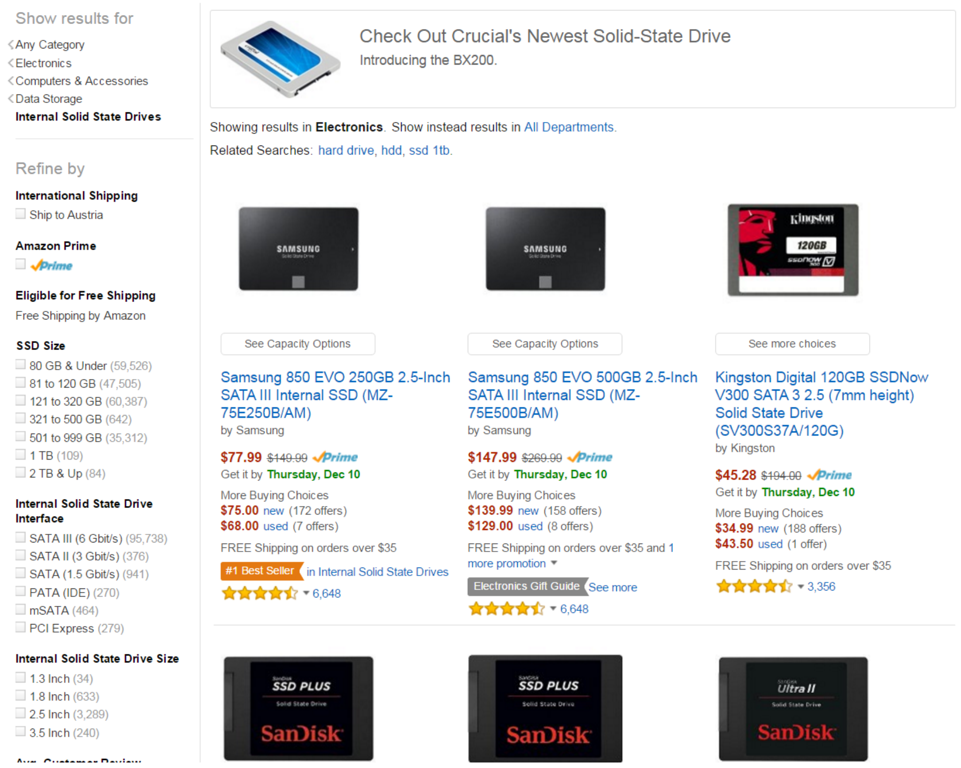
List of Figures

List of Tables

Chapter 1

Introduction

Faceted search interfaces support users in the search and exploration of a collection of items. By offering guidance, these interfaces help users to narrow down the search space intuitively and mitigate problems arising with earlier generations of search interfaces such as the complex syntax required in earlier parametric search interfaces [Tunkelang, 2009, Chapter 3.1]. Faceted search interfaces make use of *facets*: multiple independent taxonomies for different properties of items. As an example, solid state drives could be described with facets for the drive size, the price, the brand or the customer ratings. Figure 1.1 shows an example of the search results for "SSD" on amazon.com [Amazon, 2015b] and displays the faceted search interface. Note that facet values generally differ for different items (e.g., a search for shoes would return entirely different facets such as *color*).



Show results for

‹ Any Category
‹ Electronics
‹ Computers & Accessories
‹ Data Storage
Internal Solid State Drives

Refine by

International Shipping
☐ Ship to Austria

Amazon Prime
☐ ✓Prime

Eligible for Free Shipping
Free Shipping by Amazon

SSD Size
☐ 80 GB & Under (59,526)
☐ 81 to 120 GB (47,505)
☐ 121 to 320 GB (60,387)
☐ 321 to 500 GB (642)
☐ 501 to 999 GB (35,312)
☐ 1 TB (109)
☐ 2 TB & Up (84)

Internal Solid State Drive Interface
☐ SATA III (6 Gbit/s) (95,738)
☐ SATA II (3 Gbit/s) (376)
☐ SATA (1.5 Gbit/s) (941)
☐ PATA (IDE) (270)
☐ mSATA (464)
☐ PCI Express (279)

Internal Solid State Drive Size
☐ 1.3 Inch (34)
☐ 1.8 Inch (633)
☐ 2.5 Inch (3,289)
☐ 3.5 Inch (240)

Check Out Crucial's Newest Solid-State Drive
Introducing the BX200.

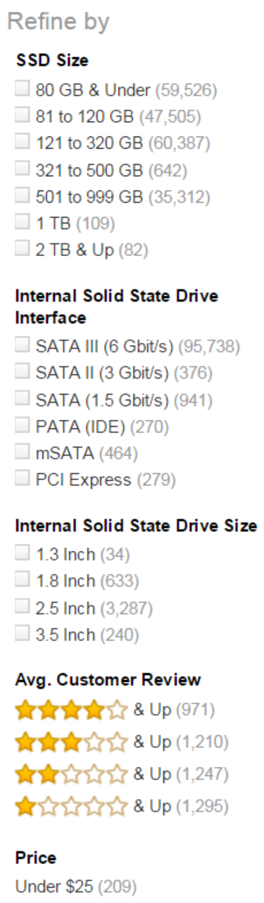
Showing results in **Electronics**. Show instead results in All Departments.
Related Searches: hard drive, hdd, ssd 1tb.

See Capacity Options
Samsung 850 EVO 250GB 2.5-Inch SATA III Internal SSD (MZ-75E250B/AM)
by Samsung
$77.99 ~~$149.99~~ ✓Prime
Get it by **Thursday, Dec 10**
More Buying Choices
$75.00 new (172 offers)
$68.00 used (7 offers)
FREE Shipping on orders over $35
#1 Best Seller in Internal Solid State Drives
★★★★☆ 6,648

See Capacity Options
Samsung 850 EVO 500GB 2.5-Inch SATA III Internal SSD (MZ-75E500B/AM)
by Samsung
$147.99 ~~$269.99~~ ✓Prime
Get it by **Thursday, Dec 10**
More Buying Choices
$139.99 new (158 offers)
$129.00 used (8 offers)
FREE Shipping on orders over $35 and 1 more promotion
Electronics Gift Guide See more
★★★★☆ 6,648

See more choices
Kingston Digital 120GB SSDNow V300 SATA 3 2.5 (7mm height) Solid State Drive (SV300S37A/120G)
by Kingston
$45.28 ~~$194.00~~ ✓Prime
Get it by **Thursday, Dec 10**
More Buying Choices
$34.99 new (188 offers)
$43.50 used (1 offer)
FREE Shipping on orders over $35
★★★★☆ 3,356

Figure 1.1: Example of a faceted user interface based on a search for "SSD" on amazon.com [Amazon, 2015b] (screenshot taken by the authors). The figure shows the *facets* on the left side (*SSD Size*, *Internal Solid State Drive Interface*, and so on) and corresponding *facet values* (such as *80 GB & Under*). For every possible facet value, the interface also shows a *query preview*, i.e., the number of remaining results if that facet value is selected.



Refine by

SSD Size

☐ 80 GB & Under (59,526)
☐ 81 to 120 GB (47,505)
☐ 121 to 320 GB (60,387)
☐ 321 to 500 GB (642)
☐ 501 to 999 GB (35,312)
☐ 1 TB (109)
☐ 2 TB & Up (82)

Internal Solid State Drive Interface

☐ SATA III (6 Gbit/s) (95,738)
☐ SATA II (3 Gbit/s) (376)
☐ SATA (1.5 Gbit/s) (941)
☐ PATA (IDE) (270)
☐ mSATA (464)
☐ PCI Express (279)

Internal Solid State Drive Size

☐ 1.3 Inch (34)
☐ 1.8 Inch (633)
☐ 2.5 Inch (3,287)
☐ 3.5 Inch (240)

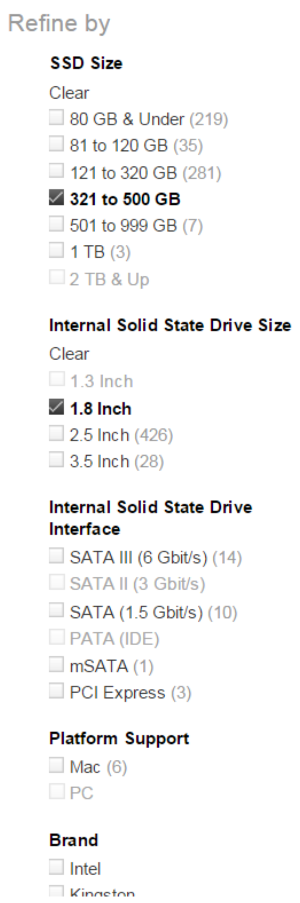
Avg. Customer Review

★★★★☆ & Up (971)
★★★☆☆ & Up (1,210)
★★☆☆☆ & Up (1,247)
★☆☆☆☆ & Up (1,295)

Price

Under $25 (209)

(a) Initial set of facets and values.



Refine by

SSD Size

Clear
☐ 80 GB & Under (219)
☐ 81 to 120 GB (35)
☐ 121 to 320 GB (281)
☑ **321 to 500 GB**
☐ 501 to 999 GB (7)
☐ 1 TB (3)
☐ 2 TB & Up

Internal Solid State Drive Size

Clear
☐ 1.3 Inch
☑ **1.8 Inch**
☐ 2.5 Inch (426)
☐ 3.5 Inch (28)

Internal Solid State Drive Interface

☐ SATA III (6 Gbit/s) (14)
☐ SATA II (3 Gbit/s)
☐ SATA (1.5 Gbit/s) (10)
☐ PATA (IDE)
☐ mSATA (1)
☐ PCI Express (3)

Platform Support

☐ Mac (6)
☐ PC

Brand

☐ Intel
☐ Kingston

(b) Facets with selected values.

Figure 1.2: Example of value selection in a faceted user interface based on a search for "SSD" on amazon.com [Amazon, 2015b] (screenshot taken by the authors). (a) shows the initial set of facets and values, which are then narrowed down after the selection of two facets values in (b). The interface aids the user by showing the number of remaining results, should an additional value be selected. Furthermore, options that would lead to empty results sets are disabled (shown in gray).

Frequently, these interfaces aid users by offering guidance through the search space. Interfaces often feature *query previews* by showing the number of remaining results before selecting an option. Another type of guidance is the avoidance of empty results sets by disabling selections that would lead to conditions impossible to match for any item in the database. Figure 1.2 shows an example of these guidance techniques.

In what follows, this survey presents i) a description of the fundamentals of faceted user interfaces, ii) a classification of information systems making use of faceted search and iii) a survey on the state-of-the-art tools and implementations.

Chapter 2

Theory

This chapter introduces the basic concepts of *information retrieval* and lays out *faceted navigation* and *faceted search*. Generally, search user interfaces allow users to interact with a system to enter a query and receive results to satisfy information needs [Marti A. Hearst, 2009, Chapters 0-1]. Initially, search interfaces were mostly used by specialists and offered rich features and customization options (such as elaborate search masks for libraries), but were also complex. Since the advent of the Web, search interfaces have mostly been kept simplistic and adopted only the most generally accepted features as an attempt to include all of the Web's diverse user base.

2.1 Information Retrieval

Information in a system needs to be retrievable in order to be useful to the system's users. The process of obtaining *relevant* documents is known as *information retrieval* [Tunkelang, 2009, Chapter 2.1]. Information retrieval systems frequently make use of *set retrieval* [Raghavan and Schatze, 2008, Chapter 1][Tunkelang, 2009, Chapter 2.2]. This approach divides the information space into into relevant and not relevant documents with respect to a query. From its use of logic operators such as *AND*, *OR* or *NOT*, set retrieval is also known as *Boolean Retrieval*. Notably, set retrieval does not provide any ranking of the resulting set.

The accuracy of information retrieval systems can be measured in terms of *precision* and *recall* [Tunkelang, 2009, Chapter 2.3]: *Precision* is the fraction of retrieved items that are relevant for the query. *Recall* is the fraction of relevant items that the system returned out of all relevant items. Precision and recall are usually a trade-off. For this reason, accuracy is often also evaluated with the harmonic mean of precision and recall (known as the F_1 *score*).

The use of Boolean operators and the lack of ranking makes is difficult for non-experts to achieve good results with set retrieval. To simplify this process, most state-of-the-art search engines, for instance on the web, allow for unstructured textual queries and use a retrieval method named *ranked retrieval* [Tunkelang, 2009, Chapter 2.4]. This method places less importance on precision, but instead ranks the retrieved items according to relevance and presents the most relevant first. This approach blurs the line between relevant and not relevant items and leaves it up to users to decide up to which point in the result list they still deem results relevant to the query. The most prominent ranking methods for hypertext are PageRank [Page et al., 1999] and HITS [Kleinberg, 1999].

2.2 Information Seeking Strategies

The human information seeking process can be described by a variety of models such as Information Foraging [Pirolli, 2007] or Berrypicking [Bates, 1989]. Generally, information seeking makes use of a variety of strategies, ranging from exploratory search to keyword search for fact finding [Sutcliffe and Ennis, 1998].

An important distinction in information seeking strategies can be made based on search versus browsing. In general, search is carried out by means of a query and inspection of provided results. Successive queries can

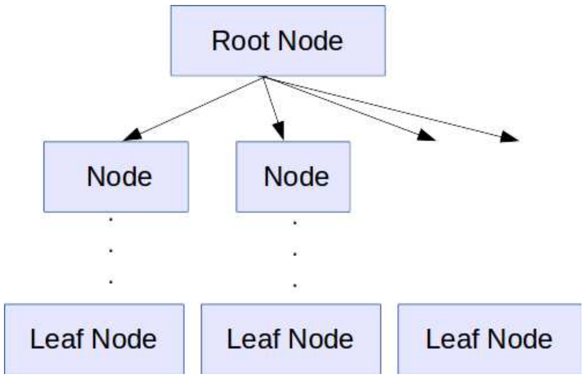


Figure 2.1: Example of a static category tree (chart created by the authors, adapted from [Tunkelang, 2009]).

target an arbitrary area of interest and are potentially independent of previous user actions. Browsing, on the other hand, consists of a series of related actions, such as following links [Marti A. Hearst, 2009, Chapter 3.5.3].

Research in cognitive psychology has shown that it is easier to recognize concepts than it is to explicitely express them [Marti A. Hearst, 2009, Chapter 3.5.3]. Generally, users find it easier to progress in an information retrieval process in small steps, allowing them to view information in context. Some users prefer navigation as an information-seeking strategy over keyword search, even when they know exactly what they are looking for [Teevan et al., 2004].

2.3 Categorization

Categories are sets that group related items and assign them a label. As such, categories are many times one of the main navigational aids in information systems. Often the first item users interact with, categories are useful in Web and search interfaces to divide content and narrow down the search space [Marti A. Hearst, 2009, Chapter 8].

A collection of categories can be arranged into a scheme. A hierarchical arrangement, with exhausting and non-overlapping categories is called a *taxonomy* [Andrews, 2015, Chapter 6.2][Rosenfeld and Morville, 2002, Chapter 9]. Figure 2.1 shows an example of a hierarchically arranged collection of categories. However, strictly hierarchical classification can be too rigid and might not even be always possible in an unambiguous way: in general, items can belong to multiple categories. For instance in a library, it seems equally valid to place a book about European history under category *Europe/History* than would be under category *History/Europe* [Tunkelang, 2009, Chapter 1.1]. A possible solution would be to use a *polyhierarchy* with overlapping categories arranged hierarchically. However, this poses new problems such as deciding what view of the hierarchy to offer to the user.

A different solution is to come up with a set of *facets*—properties of items. These facets can then be arrange into separate taxonomies, one for each facet (i.e., multiple arrangements like the one in Figure 2.1). Items are then described by a set of facets [Marti A. Hearst, 2009, Chapter 8.6]. Facets are generally thought to be orthogonal categories, such as the color and the size for a pair of shoes. An early example of a faceted classification scheme for libraries is the Ranganathan Colon Classification [Ranganathan, 1969]. In such a system, *Europe* and *History* would be a facet describing the location and the topic. The presence of categories allows users to more effectively narrow down the search space: categories are useful as attributes to sort data by and can be used to filter results to include only those matching certain criteria.

In the earlier days of the Web, so-called *directories* were used for information retrieval [Tunkelang, 2009, Chapter 2.5]. Directories such as the Yahoo! Web Directory or the Open Directory Project to divided the information space and allowed users to navigate it. One advantage over search is that in displaying subsets of items in the database (as categories), guidance can be provided to the user in what is available and what might be interesting subjects in the database. However, the problem again is that categories could be organized in ambiguous and multiple ways, and user might not find the right path to documents immediately. Web directories were also necessarily confined to a restricted set of descriptors and fail to address what is called the *vocabulary problem*: Different users use strongly differing words to describe concepts. An early study [Furnas et al., 1987] showed that any two users only use the same terms in around 20% of cases.

2.4 Faceted Information Retrieval

In what is called *parametric search* [Tunkelang, 2009, Chapter 3.1], users can specify several attributes to filter by in a type of Boolean search interface (e.g., a combination of "History" AND ("EUROPE" OR "ASIA") clauses). Depending on the selected property, some input fields are more suitable than others. However, one of the issues with this approach is that resulting sets can be empty on the one hand or too large on the other hand (for a query that is either under- or over-specified), and it is not always easy for users to recognize what parameters to specify.

These issues of parametric search can be overcome by using faceted categories. Empty result sets can be avoided by only offering choices that lead to non-empty result sets in the interface. Generally speaking, facets can be seen as an extension of filters to multiple category hierarchies and allow filtering along multiple dimensions simultaneously [**Whitenton2014**][Marti A. Hearst, 2009, Chapter 8.3]. Since facets describe items from several vista points, they also allow users to reach items on multiple different paths. Faceted categories enable *faceted navigation* and *faceted search*, which are described in the following.

2.4.1 Faceted Navigation

Facets enable *faceted navigation*, wherein users are guided in their queries. Instead of giving the input to the query all at once, the user can refine the search result step-by-step and gradually narrow down search results to a managable size [Andrews, 2015, Chapter 6.4][Tunkelang, 2009, Chapter 3.2]. After selecting a value for one facet, most faceted navigation interfaces constrain possible selection options for other, yet unspecified facets. In many cases, this eliminates dead ends, where users would be left with an empty result list for an unsatisfiable combination of search parameters.

2.4.2 Faceted Search

Faceted navigation is based on metadata (such as publication date or price). However, in most real-world scenarios, items are semistructured: they contain some structured data (e.g., metadata) as well as some unstructured (free-form) text [Tunkelang, 2009, Chapter 3.3][Andrews, 2015, Chapter 6.4]. For this reason, faceted information retrieval systems frequently integrate navigation with keyword search, allowing for *faceted search*: After an initial keyword search, the results are used as a seed set for faceted navigation. These results can then again be used to narrow down the search space just like in faceted navigation. In comparison the conventional search, faceted search improves the search experience by a ways to discover and explore the item database.

2.4.3 Notable projects

Perhaps the best-known pioneering work on faceted search is the *Flamenco* project, lead by Marti Herst in the 1990s at Xerox PARC [Marti A. Hearst, 2009, Chapter 8.6.1], which integrated browsing with keyword search using facets. Studies have shown that user prefer the Flamenco interface to a standard search interface for a range of domains.

Other notable projects include the *Relation Browser* by [Capra and Marchionini, 2008], mSPACE [Karam, Zhao et al., 2003] and *Parallax* [Huynh and Karger, 2009]. The latter, which was developed at MIT, is an rich user interface that allows for set-based browsing and can be used to explore a semantic web.

Chapter 3

Faceted User Interfaces

An important goal of search interfaces is to guide the users through the search process and allow them to specify their information needs. Depending on the domain, applications implement faceted search interfaces in different ways. Despite the large range of possible designs, making the user experience as easy, enjoyable and efficient generally can be seen as the main objectives. The way an application presents search results and facets is crucial for an effective search process [Tunkelang, 2009; M. Hearst, 2006].

This chapter analyzes the design and integration of faceted search interfaces within real-world applications. Furthermore, a taxonomy of common characteristics is be derived from the analyses.

3.1 Characteristics of Faceted Search Interfaces

In what follows, faceted user interfaces are investigated along four characteristics of faceted user interfaces: *UI Focus*, *Placement*, *Widgets* and *Dynamic/Static Facets*. An overview of these, together with exeemplary values is shown in Table 3.1. Note that these characteristics can themselves be viewed as *facets* of interfaces, as they act as orthogonal taxonomies.

3.1.1 User Interface Focus

Faceted search interfaces offer an easier-to-use solution the more traditional free-text and parametric search interfaces (cf. Chapter 2). When integrating faceted search into an application, however, mostly a combination with these traditional search types is used. Mainly, there are two distinct approaches to the search process: an interface can either focus on an approach using free-text search first, or start the search process through category navigation. Both methods exhibit a certain way of initializing the query leading to a list of results, which can then be narrowed down by applying filters or selecting values for facets [Tunkelang, 2009].

An example for a search-box-based implementation is the e-commerce Web application of Amazon [Amazon, 2015a]. Figure 3.1 depicts Amazon's landing page, which prominently displays a search box on top. Apart from the search box, Amazon also offers a category selection menu located in the top left corner. Although the application offers both search box and navigation, the main focus clearly lies on the search input field.

By contrast, Figure 3.2 demonstrates the navigation-first approach implemented by Willhaben [Willhaben, 2015]. Willhaben's navigation is constructed from a tree of categories: the frontpage presents the four top-level

Interface aspect	Exemplary values
UI Focus	Navigation vs. Search
Widgets	Nominal/categorical, numerical, location-based, binned
Placement	Left sidebar, left & top, tabbed
Dynamic/Static Facets	Facet values change with query context

Table 3.1: The four aspects of faceted search interfaces investigated in this survey.

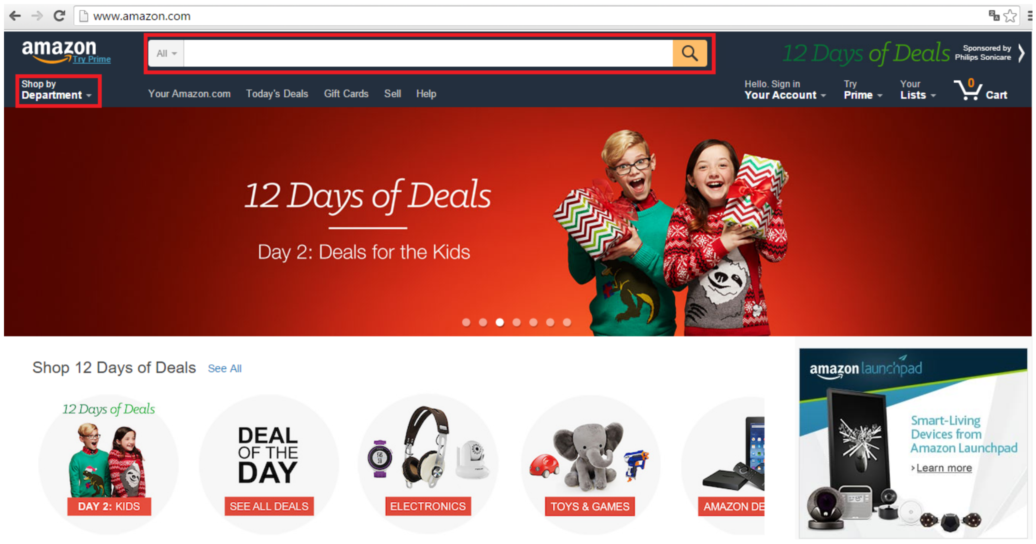


Figure 3.1: Amazon.com integrates faceted search using a search-box-based approach with a secondary category selection under the site's logo. Screenshot taken by the authors.

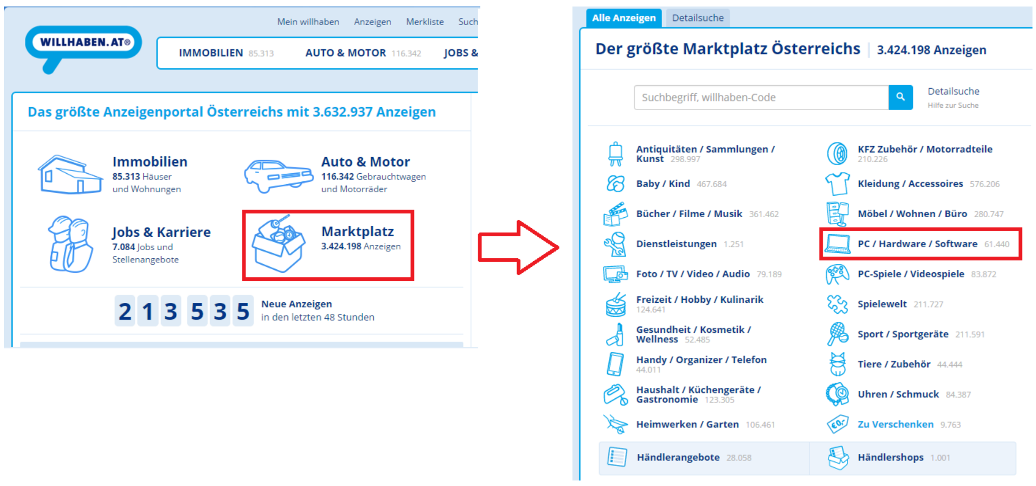


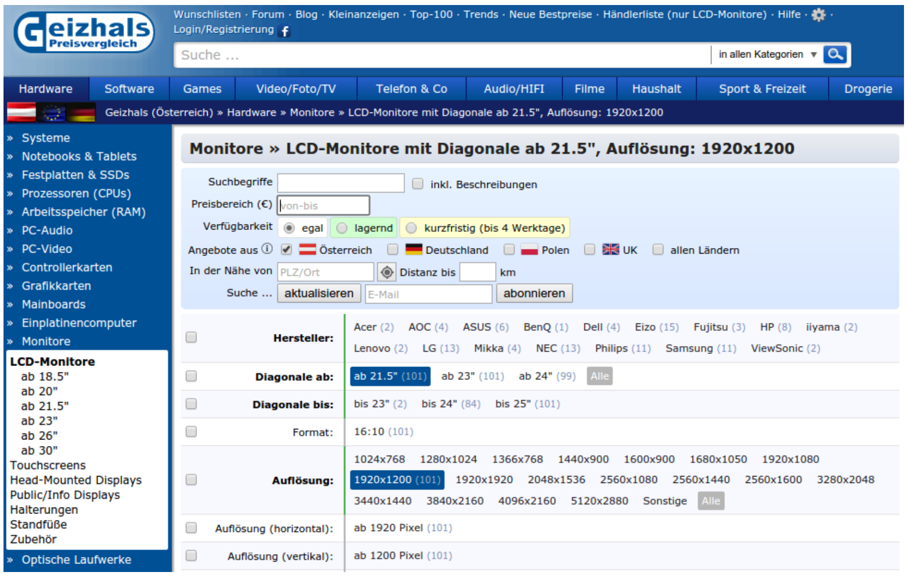
Figure 3.2: Willhaben.at implements a navigation-first approach by only showing the four top-level categories. Once users choose a category, a list of subcategories appears, allowing further refinement. Screenshot taken by the authors.

categories; further pages offer subcategories for selection. After selecting a subcategory, a list of results is shown. Like Amazon's implementation, Willhaben also offers an alternative way of navigation. In Willhaben's case this is done by offering a search box for the subcategory selection as well as the final result list.

3.1.2 Widgets

After initializing the search by either specifying a category or a free-text query, faceted interfaces permit to further refine the result set. This can be achieved by selecting values for facets (which corresponds to applying filters). Depending on the domain and properties related to the item collection, various types of facets exist [Jeffery Callender, 2015]. For the most part, numerical, categorical or domain-specific types are used. For example, facets like price, quantity or dimensions are typical numerical facets. As depicted in Figure 3.3, categorical facets are for instance brand or manufacturer names, technologies or materials. Domain-specific facet values are tailored towards the specific data of a domain and include for instance location data or date/time values.

Using widgets to view and interact with facet values is common in state-of-the-art applications of faceted


Geizhals Preisvergleich

Wunschlisten · Forum · Blog · Kleinanzeigen · Top-100 · Trends · Neue Bestpreise · Händlerliste (nur LCD-Monitore) · Hilfe ·
Login/Registrierung

Suche … | in allen Kategorien

Hardware | Software | Games | Video/Foto/TV | Telefon & Co | Audio/HIFI | Filme | Haushalt | Sport & Freizeit | Drogerie

Geizhals (Österreich) » Hardware » Monitore » LCD-Monitore mit Diagonale ab 21.5", Auflösung: 1920x1200

» Systeme
» Notebooks & Tablets
» Festplatten & SSDs
» Prozessoren (CPUs)
» Arbeitsspeicher (RAM)
» PC-Audio
» PC-Video
» Controllerkarten
» Grafikkarten
» Mainboards
» Einplatinencomputer
» Monitore

LCD-Monitore
ab 18.5"
ab 20"
ab 21.5"
ab 23"
ab 26"
ab 30"
Touchscreens
Head-Mounted Displays
Public/Info Displays
Halterungen
Standfüße
Zubehör

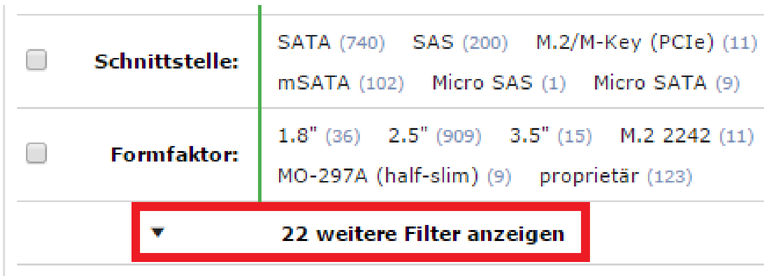
» Optische Laufwerke

Monitore » LCD-Monitore mit Diagonale ab 21.5", Auflösung: 1920x1200

Suchbegriffe	☐ inkl. Beschreibungen
Preisbereich (€)	von-bis
Verfügbarkeit	[✓ egal] lagernd kurzfristig (bis 4 Werktage)
Angebote aus ⓘ	[✓ Österreich] ☐ Deutschland ☐ Polen ☐ UK ☐ allen Ländern
In der Nähe von	PLZ/Ort Distanz bis km
Suche …	aktualisieren E-Mail abonnieren

Hersteller:	Acer (2) AOC (4) ASUS (6) BenQ (1) Dell (4) Eizo (15) Fujitsu (3) HP (8) iiyama (2) Lenovo (2) LG (13) Mikka (4) NEC (13) Philips (11) Samsung (11) ViewSonic (2)
Diagonale ab:	[✓ ab 21.5" (101)] ab 23" (101) ab 24" (99) Alle
Diagonale bis:	bis 23" (2) bis 24" (84) bis 25" (101)
Format:	16:10 (101)
Auflösung:	1024x768 1280x1024 1366x768 1440x900 1600x900 1680x1050 1920x1080 [✓ 1920x1200 (101)] 1920x1920 2048x1536 2560x1080 2560x1440 2560x1600 3280x2048 3440x1440 3840x2160 4096x2160 5120x2880 Sonstige Alle
Auflösung (horizontal):	ab 1920 Pixel (101)
Auflösung (vertikal):	ab 1200 Pixel (101)

Figure 3.3: Geizhals.at, a price comparison website uses an interface with a large number of facet filters. The facet value selection of geizhals.at: placed on top is a list of widgets containing specific control elements. For facets with a large set of values, geizhals utilizes a list of checkboxes. Facets that are important to a category are mapped into the category tree. Here, this applies to the screen size of the LCD screens. Screenshot taken by the authors.



Schnittstelle:	SATA (740) SAS (200) M.2/M-Key (PCIe) (11) mSATA (102) Micro SAS (1) Micro SATA (9)
Formfaktor:	1.8" (36) 2.5" (909) 3.5" (15) M.2 2242 (11) MO-297A (half-slim) (9) proprietär (123)

▾ **22 weitere Filter anzeigen**

Figure 3.4: Geizhals.at utilizes a "show more filters" option to reduce the complexity of the facets currently displayed. Screenshot taken by the authors.

navigation and search. Widgets group facets in a meaningful way to support the user during refinement. Utilizing appropriate control elements such as input fields, checkboxes, radio buttons or range sliders for facet selection is crucial for a valuable user experience. The choice of control elements also influences the behavior of faceted search systems. With radio buttons, only a single value can be selected at once, hence the amount of queries that can be formulated is more restrictive (as opposed to checkboxes, which allow multiple selections). The same applies to numeric input fields versus binned numeric value sliders. Figure 3.3 illustrates various examples of control elements used for interacting with facet values.

Although widgets are mainly used for refining results, some sites also use widgets to provide visual feedback about the current result set. For example, for items with location data attached as facet values, a map indicating their position can be valuable feedback to the user. Another form of visual feedback, especially when using numerical facets, is a histogram showing the distribution of the item's facet values. An example of widgets used for visual feedback is depicted in Figure 3.5

3.1.3 Placement on the Page

Typical layouts used in faceted user interfaces are composed of a main section and a fixed left-hand sidebar and a top header area. Some layouts omit the sidebar and choose to extend the header area instead. Based on this specific layout, facets are either placed in the sidebar or, when omitting the sidebar, in the extended

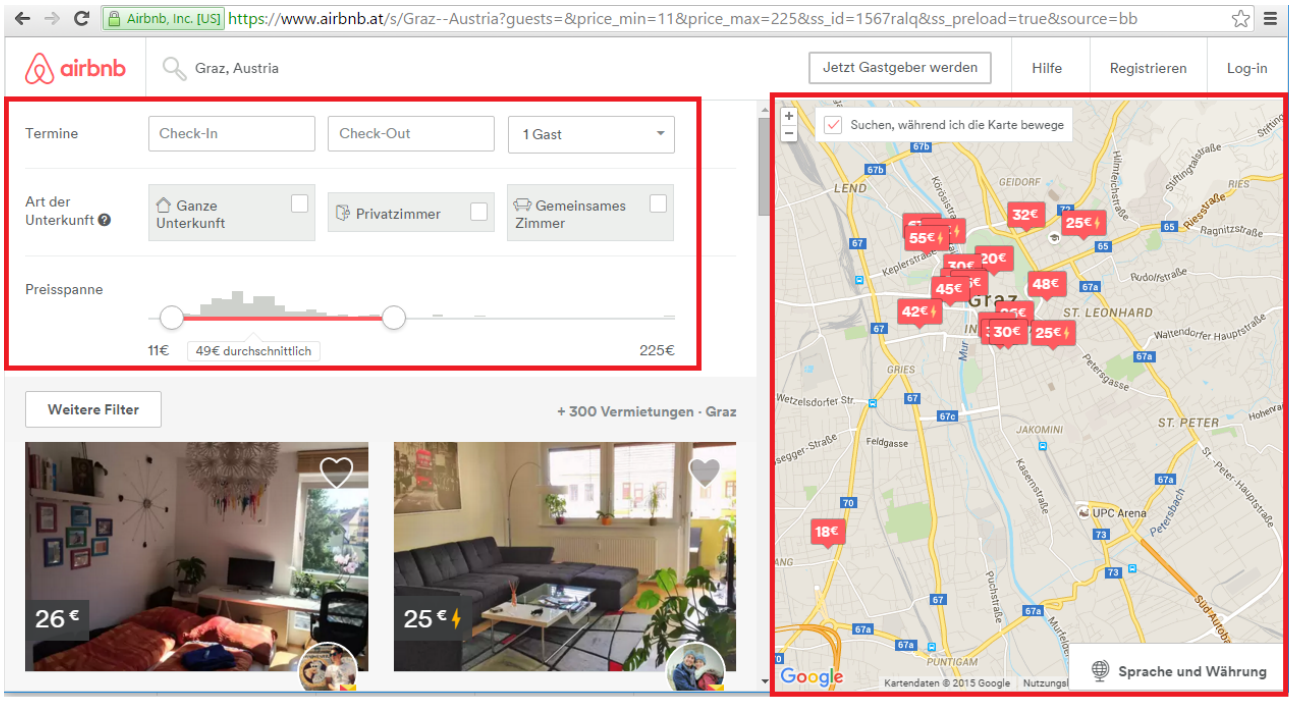


Figure 3.5: The site airbnb.com applies widgets not only for refinement, but also for visual feedback. For example, the price range selection features a histogram of the distribution of its facet values. Furthermore, a map with anchors for each item in the result set is used for the user to maintain an overview of the current state. Screenshot taken by the authors.

header. Results are generally positioned in the main section, which makes them more recognizable by the user and easier to focus on. Generally, as the user's main objective are the retrieved items, the main section is a favorable place for the results [Tunkelang, 2009].

When dealing with huge item collections, a very large amount of facets and facet values can be available and there may not be sufficient space provided by the layout for displaying all of them. A possible solution is to only show the top-ranked facets at first, with the option to view more filters. An example of this concept is illustrated in Figure 3.4. The idea here is to only reveal the essential so the user can complete his task successfully.

3.1.4 Dynamic vs. Static Facet Values

Another distinction of faceted search interfaces can be made based on whether they take the current query context into account and adapt the available facet values accordingly. During the refinement process, facet values are added or removed, which consequently changes the current query context. To simplify the process for users, dynamic methods adopt their facets to show only the available ones in the result set. An example of dynamic facet values can be seen in the selection of Figure 1.2. Hence it is not possible for the user to formulate a query which leads to zero hits. Static approaches, on the other hand, do not incorporate already chosen facets. Dynamic facets are usually considered superior to static values. However, static facets values can be useful for large item collections such as Flickr or when insufficient metdata is available to offer refined facets.

3.2 Examples

In this section, an analysis of examples of different faceted user interfaces found on the Web is conducted, applying the categorization described in the previous chapter, and pointing out differences in the implementation and unique features. This section first gives a detailed anylsis of four examples, followed by the summary of the properties of a total of 14 sites.

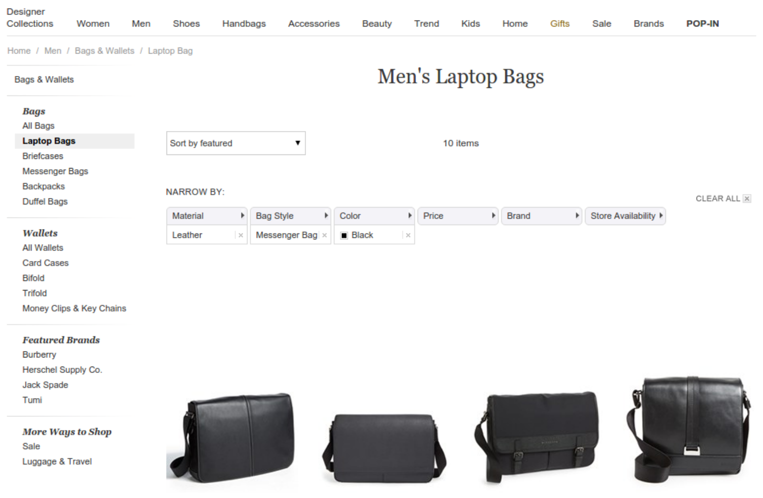


Figure 3.6: Nordstrom.com, an online clothing store. The site places a category tree to the left and facet filters on top of search results—a layout that is very common with e-commerce sites. Screenshot taken by the authors.

3.2.1 E-commerce: nordstrom.com

The Web site of fashion retailer Nordstrom [Nordstrom, 2015] was ranked first in a benchmark measuring product-list and filtering usability of e-commerce sites by Smashing Magazine in 2015 [Holst, 2015]. The site's main focus clearly lies on category browsing: Once a main category is selected from the front page, a multi-leveled category-tree can be found in the left sidebar, in which categories can be selected step-by-step. At each step, only a single level of category is visible, with additional categorical facets shown below that (depending on the category). In addition to the categories, a search function is available.

Other facet filters are visible above the search results, as depicted in the screenshot in Figure 3.6. All input elements of these filters are nominal lists. Where meaningful, multiple options can be chosen to form a logical disjunction ('OR') for the query. The facets itself are dynamic in their values—values which do not occur in the current query are grayed-out and can thus not be selected. By only using facet filters (and refraining from using the free-text search), empty result sets cannot be reached. If the query is started with a free-text search, query preview counts are displayed next to the categories of the tree in the left area. The values of facet filters themselves never display query preview counts.

3.2.2 Price Comparison: geizhals.at

Being a price comparison site predominantly for electronic consumer goods, Geizhals [Geizhals, 2015] has a strong focus on filtering items with a large number of specific facets, applicable to the various categories of electronic products such as computer hardware, digital cameras and office printers. Consequently, before any of the filters can be used, a category needs to be chosen from the multi-leveled hierarchy. Likewise, a query to the free-text search does not lead to the results being used in a faceted navigation style but also requires a category selection to proceed. The product category can be selected in a menu in the left-sidebar, with the top-level category mirrored into a second, horizontal menu in the header bar. All items in the second category level are displayed, as are potential sub-categories to the currently selected one. One important facet per category can be mapped into the category-tree, in the example screenshot (Figure 3.3) that depicts the category LCD screens, this is the screen size.

Most of the filter widgets are hidden by default, so that users are not overwhelmed by the sheer amount of input options (once fully expanded, the filter inputs can take up most of the screen area). The widgets use nominal lists as input and, for larger numbers of facet, shows only the most important ones at first. For some inputs, such as the price range, free-text input is used. Other than that, the facets are dynamic and query preview counts are displayed.

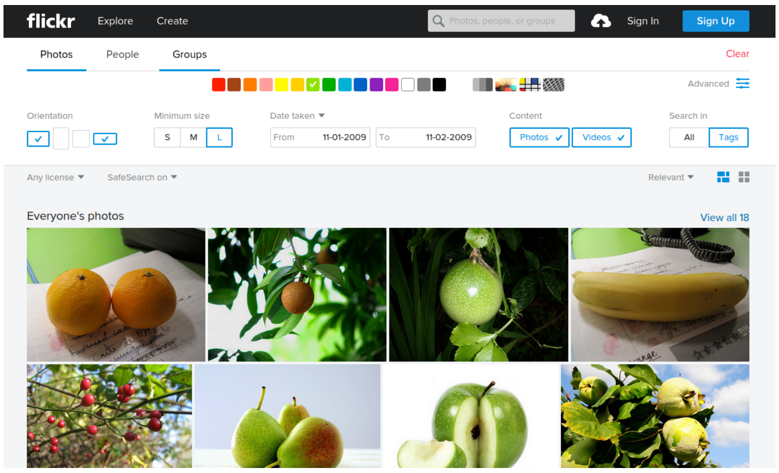


Figure 3.7: The flickr.com image search is an example for the search-centric layout pattern with facet filters on top. The input widgets below the color selection are hidden by default—a click on the button labelled "Advanced" revealed the additional filters. Screenshot taken by the authors.

3.2.3 Large Database : Flickr

Flickr [Flickr, 2015] is an online image storing and sharing site. Its interface clearly centers on free-text search, with additional filters to refine and further narrow down the results. For Flickr, the free-text search simply replaces any categorization regarding the contents or the topic of the images. Next to the free-text input, one or more options to filter by color can be chosen. Multiple selections for colors are combined by a logical conjunction ('AND').

All other filters are hidden by default and can be accessed by clicking on an *advanced* button. Most of the additional input widgets allow to select values from nominal lists, with values that are enhanced with icons. A date-picker is used for the date input (see screenshot in Figure 3.7) and some inputs are hidden behind drop down lists, e.g., the filter to select the desired license. The Flickr search provides static facets that remain the same for all queries and lacks query previews. This might be due to the very large number of results generally in play on the site. When narrowing down the result far enough, this means that empty result-sets can occur.

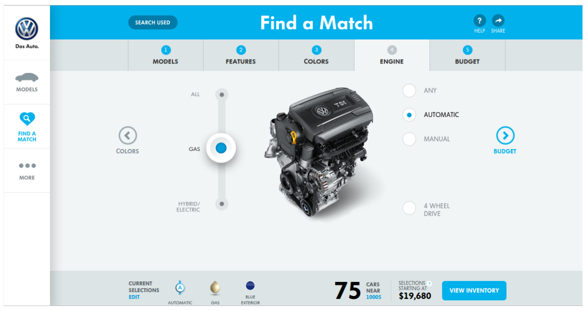
3.2.4 Product Configurator : Volkswagen

The product page for the American site of Volkswagen [Volkswagen, 2015], uses a faceted search interface that allows users to filter and narrow down the set of product models and configurations until a manageable number of results remain. In a sense, the interface can be seen as a product configurator, that matches configurations to available products.

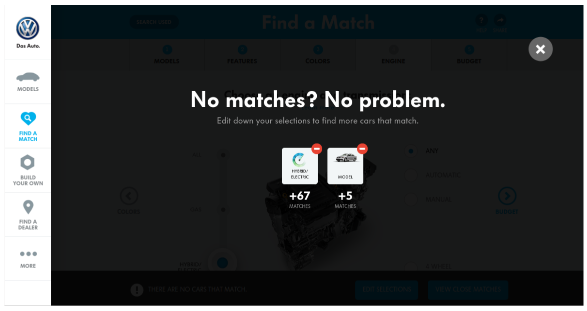
The site uses a presentational layout that diverges from the patterns that are otherwise most often encountered (see Figure 3.8): for once, the input widgets take up most of the screen, whereas the result list is collapsed to a mere counter (the result-set is presented in full detail on the final page).

Second, it uses a tabbed interface, which does not only give room to the rich input widgets, but also guides the user through the steps of the search process. The tabbed search interface is an interesting pattern that might also be useful in cases where the screen size is generally a limiting factor, such as with the smaller screens of mobile devices [Zimmer, Kuehn and Schlegel, 2014].

The widgets, although enhanced with graphics, still mostly provide selections from lists of nominal values. When selecting or de-selecting a value, the change in the number of total results is animated with moving digits, communicating to the user the quality of the facet value that was chosen in terms of the availability of the products and their distribution in the result set. Other than that, there are no query-preview counts displayed next to the facet values. Facets are not dynamic, so by entering certain combinations users can end up with



(a) The interface uses tabs to give the input widgets more space. The widgets mostly input for nominal values, albeit enhanced with graphics that are suitable for the topic.



(b) Once a query is constructed that yields zero results, a screen is displayed that helps the user to broaden the search again.

Figure 3.8: Product finder for Volkswagen. It uses a tabbed interface (a), and a helper view for the case where a query result is empty (b). Screenshots taken by the authors.

empty results. Should this be the case, the interfaces displays all selected facet values of the current search query, alongside the number of potential results which would be gained by de-selecting that facet-filter.

3.2.5 Analyis of 14 examples

This section provides the results of the analysis of 14 Web sites with faceted interfaces, including the four examples presented in the previous subsections. For these 14 examples, the interfaces can be divided into classic e-commerce sites, large-document databases, and sites with a more customized layouts that tend to put focus on novel input widgets or novel ways to present result sets. The list of sites, along with the categorization regarding certain aspects of interfaces and search functionality is depicted in Table 3.2 and Table 3.3.

Nearly all of the interfaces display the number of results of the current query. Some kind of choice regarding the order of results can usually be applied, for instance the results can be sorted by relevance or by date of publication. For e-commerce sites, usually a selection of a category as starting point for search is equally possible to that of a free-text search. In many cases though, a category has to be selected first in order to begin drilling down by free-text search or applying facet filters, as can be seen with geizhals.at. Nevertheless, the free-text search is available in nearly all of the interfaces. Especially larger item databases (such as google.com or flickr.com) tend to have it as the main starting point for the query.

The most common layout patterns that were found are to place the category tree to the left of the search results, and additional facet filters on top of the search results, or having both input parts of the interface in the left sidebar. Where a topic or category is a less prominent facet, the facet interface is placed on top of the result list, without any content in sidebars. Some recent sites tend to use highly customized layouts and widgets, for instance airbnb.com or vw.com.

In dealing with large categories, options described by [M. Hearst, 2006] such as the use of a ’fly-away’ (show on hover) menus were encountered, but more commonly, some part of the facet values is hidden at first, and only shown by request of the user. The same is also very common with inputs for less important facets, those are often hidden by default. For categories with multiple hierarchies, mostly a step-by-step drill-down process is chosen over the display of a large hierarchical tree.

Name	URL	Type	Placement
Airbnb	`http://airbnb.com`	lodging	custom
Amazon	`http://amazon.de`	shop	left
BlueTomato	`http://blue-tomato.com`	shop	left
Flickr	`http://flickr.com`	search	top
Geizhals	`http://geizhals.at`	metashop	left/top
Google	`http://google.com`	search	top
National Aquarium	`http://aqua.org/explore/animals`	item collection	left
Nike	`http://store.nike.com`	shop	left
Nordstrom	`http://shop.nordstrom.com`	shop	left/top
UB.tugraz	`http://ub.tugraz.at`	library	left
Volkswagen	`http://vw.com`	shop	tabs/custom
Willhaben	`http://willhaben.at`	small ad	left
Yelp	`http://yelp.com`	reviews	top
Zoomsquare	`http://zoomsquare.com`	lodging	left

Table 3.2: An overview of the faceted search interfaces that analyzed and catalogued for this survey. The table shows the name, URL, the type of the Web site and the placement of the faceted interface on the site.

Name	UI Focus	Dynamic Facets	Query Preview	Free-Text Search
Airbnb	Search	partially	n	y
Amazon	Search or categories	y	y	y
BlueTomato	Search or categories	y	n	y
Flickr	Search or facets	n	n	y
Geizhals	Categories	y	y	y
Google	Search	n	n	y
National Aquarium	Facets	n	n	n
Nike	Categories	y	partially	n
Nordstrom	Search or categories	y	partially	y
UB.tugraz	Search	y	y	y
Volkswagen	Categories	n	partially	n
Willhaben	Categories	y	y	y
Yelp	Search	y	n	y
Zoomsquare	Search or categories	n	n	y

Table 3.3: Comparison of the functionality of the faceted interfaces. A *y* means yes/present, an *n* means no/missing. The aspects shown are the main UI focus and starting point for the query, the presence of dynamic facets, the presence of query preview information and lastly, if a free-text search was provided.

Chapter 4

Faceted Search Solutions

This chapter surveys current solutions for faceted user interfaces and reports on their features. This is done in three parts: Firstly, available open source solutions and their features are described. Secondly, existing commercial solutions are listed and their most important advantages (or differences) when compared to the open source alternatives are discussed, as well as the key differences among the core solutions. Thirdly and finally, this chapter presents summarization tables with the currently most relevant faceted search software options and respective capabilities.

4.1 Open Source Faceted Search Software

There are several established open source faceted search solutions that can be used at an enterprise level [Zheng, Zhang and Feng, 2013; Today, 2015]. These solutions can be differentiated from each other mainly in terms of maturity, facet term extraction (manual or automatic) and search paradigm. Other differentiation metrics exist, such as facet ranking or hierarchy construction. It is also possible to discuss the efficiency of results returned (relevance metrics, cost-based metrics) when comparing different types of search approaches [Marti A Hearst, 2006]. Some search solutions combine different sources of data with real time analytics or even data visualization. In what follows, the major open source solutions are discussed.

4.1.1 Apache Lucene

Apache Lucene is an information retrieval library written in Java and an open-source project freely available for download. This library is at the core of other search solutions such as Solr, Lucidworks and others [Apache, 2015a].

Apache Lucene features ranked searching or query types such as phrase queries, wildcard queries, proximity queries and range queries. It also includes fielded searching such as, title, author, contents in the form of fast, memory-efficient and typo-tolerant suggesters. Apache Lucene is a technology suitable for most types of application that require full text search, especially useful for cross-platform projects.

4.1.2 Apache Solr

Solr is a open-source library based on Apache Lucene and offers enterprise search. Like Lucene, Solr is written in Java and runs as a standalone full-text search server. It uses the Lucene search library at its core for full-text indexing and search [Solr, 2015].

Solr supports full text search, hit highlighting, faceted search, auto-completion (as visible e.g., at the Kershaw webstore: [Kershaw, 2015]), near real-time indexing, dynamic clustering, database integration, rich document handling and geospatial searches. Communication with Solar is done via REST-like HTTP/XML or a JSON API. *Apache Tika* [Apache, 2015c] (used for metadata and text extraction from different file formats) and *Apache OODT* [Apache, 2015b] (used for crawling new data from different sources) can be combined to automate the loading of data into Apache Solr. Targeting an enterprise context, Solr also provides distributed indexing, replication and load-balanced querying, automated failover and recovery as well as a centralized configuration.

4.1.3 Elasticsearch

Elasticsearch is based on Apache Lucene and is a flexible open source, distributed, real-time search and analytics engine. Elasticsearch provides some important key differences from other solutions. Its clusters are resilient and will detect and remove failed nodes and reorganize themselves to ensure that the data is safe and accessible. Elasticsearch has a robust set of APIs and a powerful domain-specific language (DSL) [Elastic, 2015].

Elasticsearch proposes that facets are deprecated and will be replaced by what it calls *aggregations* in a future release. They explain the reason for this as follows: the core purpose of a full-text search engine is to return a number of documents matching a query. As support for this, facets provide aggregated data based on a search query. In the simplest case, a term's facet can return facet counts for various facet values for a specific field. However, Elasticsearch tries to goes beyond this simple support and provide more complex facet implementations, such as statistical or date histogram bins, which are termed aggregations.

The fields used for facet calculations can be numeric, date/time or they can be analyzed as a single token (one can give the facet a custom name and return multiple facets in only one request). Elasticsearch was designed from the start to support scalable and distributed search applications. Like Solr, Elasticsearch provides full-text searching, complex query patterns and other advanced search features. However, Elasticsearch stores documents in JSON format and all fields are indexed by default.

Elasticsearch does not require a schema specification prior to loading documents and can automatically detect the document structure from the JSON documents directly. This is in contrast to Solr, which requires users to specify which fields are indexed. Like Solr, Elasticsearch also provides a RESTful API. Elasticsearch is known for supporting its users very actively and also offers add-on products, such as Marvel, a dashboard and analytics tool for managing the Elasticsearch cluster. However, the system it is not, yet, as mature as other available solutions (namely Solr).

4.1.4 Constellio

Constellio is an enterprise search engine built on top of Apache Solr that allows companies to search and find information through a single interface [Constellio, 2015].

Constellio features federated search of all data inside an organization, a faceted search discovery tool, automatic categorization of content, collaborative search engine (wiki capabilities) and management of the order of appearance of results.

4.1.5 Searchdaimon

Searchdaimon is an open source enterprise search engine for full text search of structured and unstructured data [ES, 2015].

Searchdaimon supports hit highlighting, faceted search, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling and full document level security.

4.1.6 Sphinx

Sphinx is an open-source full text search engine. Sphinx can be used as a stand-alone server, which can be used to communicate with other database management services by using native protocols of MySQL, MariaDB and PostgreSQL, or simply by using Open Database Connectivity (OBC). Sphinx can also be used as a storage engine for MySQL and its forks [Sphinx, 2015].

Sphinx features batch and incremental (soft real-time) full-text indexing, support for non-text attributes (scalars, strings, sets, JSON), direct indexing of SQL databases, XML documents indexing support, distributed searching support out of the box and a full text searching syntax.

4.1.7 Bobo Browse

While Lucene is good with unstructured data, Bobo Browse (much like Searchdaimon) fills in the missing piece to handle semi-structured and structured data and provides support for navigational browsing of a semi-structured dataset. Bobo Browse also provides the facets from this point of browsing [Bobo, 2015].

Bobo Browse has no need for cache warm-up for the system to perform, multi-value sort (sort documents on fields that have multiple values per doc, e.g., tokenized fields) and offers fast field value retrieval (over 30 times faster than standard IndexReader.document or int docid). It further supports facet count distribution analysis, runtime faceting results a merge library for distributed facet search, all while having a small and stable memory footprint.

4.1.8 Open Semantic Search

Open Semantic Search is a provider of a free software to power up a customized search engine, data explorer and research tools based on Apache Lucene/Solr or Elastic Search [Opensemanticsearch, 2015].

Open Semantic Search features full-text search, interactive filters (faceted search), exploratory search (overview of aggregated search results and text mining preparation), tagging and annotation, data visualization, alerts (newsfeeds), support for different file formats (doc, csv, pdf, images, open office), multiple datasources integration, automatic text recognition, responsive design, metadata management, file system monitoring, open semantic web ready. This solution appears to be one of the most up-to-date, scalable and complete open-source solutions available.

4.2 Important Commercial Players

This section surveys the most important commercial solutions that aim at combining text mining and concept discovery with enterprise sources of data, analytics for big data, data exploration and visualization, together with seamless easiness of configuration and use.

4.2.1 LucidWorks

LucidWorks offers supporting services for Lucene and Solr and also develops add-ons, such as a named entity extraction module for more advanced text analytics [Fusion, 2015].

LucidWorks generally aims at providing high accuracy of search results and integration with any data sources. It offers components and services dedicated to the analysis of big data as well to data visualization and real-time monitoring. This effort is also evident by the company's motto: ”The right data to the right people at the right time. Every single time.“ [Fusion, 2015].

4.2.2 Oracle Endeca

Endeca was one of the pioneers of faceted user interfaces (later bought by Oracle) and has developed into the most powerful enterprise data discovery platform. Oracle Endeca aims at empowering business user independence in balance with IT governance and offers powerful solutions for agile data discovery [Endeca, 2015].

Oracle Endeca provides fast and intuitive access to both traditional analytic data (leveraging existing enterprise investments) and non-traditional data (including external and unstructured information). It also uses keyword queries that enable users to articulate a specific direction for their information need, and faceted navigation that enable users to explore, refine and iterate on the results of search queries. Business users can upload and combine diverse data for agile discovery on structured and unstructured information, drag-and-drop configuration, easy layouts and intelligent defaults offer fast application creation. Endeca uses text analytics to extract key themes and has state-of-the-art search and guided navigation. Its agile data-driven approach requires no up-front modeling and provides sophisticated data integration and transformation.

4.2.3 DataSax Enterprise

DataStax Enterprise supplies built-in enterprise search functionality (based on Apache Solr) on top of Cassandra (a NoSQL database). It scales and performs in a way that meets most of the search requirements of the modern Web enterprise applications [Enterprise, 2015].

DataSax includes general full-text and faceted (categorization) search. It provides hit prioritization and highlighting, log mining, rich document (PDF, MS Word, etc.) analysis, geospatial as well as social media match ups. It also includes features such as always-on search (which ensures constant uptime and availability for search operations), live search, that makes new data instantly available for search operations (which is not the case with most other search systems).

Another important feature is the integration with CQL and Spark (a general engine for large-scale data processing and cloud computing). Search/Solr syntax is integrated with the Cassandra Query Language (CQL) and SparkSQL, which enhances both CQL and Spark in a way that allows them to operate as full featured search languages. Solr syntax may be passed directly through a CQL or SparkSQL WHERE clause, so that data can be searched for via CQL and SparkSQL in addition to the native Solr API's.

Finally, it includes the so-called online elasticity, where additional search capacity can easily be added online to scale to whatever data or user demands an application might have.

4.2.4 Searchify

Searchify offers full-text search-as-a-service. It allows ways to easily add custom full-text search, without the cost or complexity of managing search servers [Searchify, 2015].

Searchify offers a powerful hosted search (with a simple API), true real-time (updates are immediately searchable), location-aware (geo search - sort and filter by distance). It combines faceted search for easy end-user navigation, customized searches, custom scoring and sorting functions, snippets and highlighting for professional-looking results. It is fast, hosted and and scalable (scalable from small projects to millions of documents).

Searchify claims to be easier than Solr/Lucene and Sphinx in terms of configuration and usage. Client libraries in Ruby, Python, Java, PHP and other languages are provided.

4.2.5 Searchblox

Searchblox provides support for enterprise search and text analytics by using faceted search. The search is integrable with existing Web sites and is used to discover information across file systems, websites, databases, cloud or even in custom content [Searchblox, 2015].

Searchblox combines text analytics with data science algorithms to provide predictive analytics and to support business operations. It is possible to understand customer experience and employee engagement levels through analysis of the text information available in surveys and market research studies.

This software allows the creation of predictive monitoring for real businesses and enables the aggregation of multiple data sources, to analyze text sources and market variables. It processes, filters, sorts and combines text for analysis and provides visualization capabilities for text content and data variables, for the tracking of real-time insights and information changes.

Searchblox allows the creation of custom predictive models based on the different business objectives. It makes usage of eCommerce, intranet, cloud, hosted, Salesforce, big Data and eDiscovery searches.

4.3 Differences Between Core Solutions

This section compares the major solutions and presents a short discussion about the most important differences.

4.3.1 Lucene/Solr vs Cassandra

Like it was referred before, Cassandra is a NoSQL data store and it was designed for achieving high performance when handling huge amounts of data (terabytes and beyond). It also contains a powerful query language.

NoSQL databases or data stores have limited capabilities when it comes to queries: they do not have JOIN queries, as this would slow down the system. It is possible to read/write pretty fast and the data can be easily queried. It has a flexible schema, you can push sparse data into these NoSql databases. In other general sql databases you push NULL for an empty entry, whereas with NoSql, you do not need to. However, it is not possible to perform a full text searching (which might be a problem in a real-world application) [Apache, 2015a; Solr, 2015; Cassandra, 2015].

Solr on the other hand, is a TF-IDF full text search engine with a flexible schema that allows to simply mark fields that are not required. Solr helps in tokenizing, parsing and indexing the data with a high performance. It returns XML and allows to parse the XML to create data that is representable. Read queries are very fast.

Therefore, the ultimate strength lies in the combination between solutions like Lucene/Solr and NoSql databases like Cassandra.

4.3.2 Elasticsearch vs Lucene/Solr

With Solr, one of the biggest problems usually reported is that many of the new features in SolrCloud are very immature, in particular the Overseer/Queue management [Gifford, 2014], and there are some real problems with stability when running the “out of the box” SolrCloud distribution.

The Collections API in Solr is brand new and very primitive, while Elasticsearch has native, robust and battle-tested index management. Both have sensible default behavior for shard allocation to cluster nodes, but the routing framework in Elasticsearch is significantly more complete and stable than the Collections API in Solr [Gifford, 2014].

In terms of indexing, Solr is a clear winner. Solr has the advantage in both size and activity. But Elasticsearch is growing faster in terms of market share.

The Elasticsearch API is elegant and much more in line with comparable REST services. While the native hierarchical/pivot facets in Solr are useful, the out of the box experience in Elasticsearch is nevertheless much simpler than Solr. The plug-in support for both is good in Elasticsearch, but Solr supports more complete plugins.

Solr’s architecture was not built for realtime search applications and it is expected that some of the weaknesses of Elasticsearch will be quickly fixed in future releases (the releases are more frequent in the case of Elasticsearch) [Elastic, 2015; Apache, 2015a; Solr, 2015].

4.3.3 Lucene vs Solr

Solr is built on top of Lucene Core, but when we just want to embed search functionality into a desktop application, Lucene is the most appropriate choice.

For situations, when one has very customized requirements, requiring low-level access to the Lucene API classes, Solr may be just one more extra layer of indirection.

However, Solr greatly extends the core functionalities of Apache Lucene, and these functions are provided out of the box [Tutorial, 2015].

4.4 Summary of All Discussed Search Solutions

In this section we summarize the most important features of both open source and commercial solutions, that are discussed in our survey.

In the following tables (Table 4.1 and Table 4.2) we summarize the most debated [Wang, 2008; Roy et al., 2009] and discussed [Elastic, 2015; Apache, 2015a; Solr, 2015] distinctive features of both open source and commercial faceted based search solutions.

Name	First Release	Main Features
Apache Lucene	1999	*Ranked searching*. Phrase, wildcard, proximity and range queries. Field searching, *typo-tolerant suggesters*. Suitable for integration in applications that require full text search.
Sphinx	2001	Batch and incremental full text indexing. *Support for non-text attribute*s (scalars, strings, sets, JSON). Direct indexing of SQL databases and XML documents. *Distributed searching support out of the box* and a full text searching syntax.
Solr	2004	*Near real-time indexing, dynamic clustering*, DB integration, rich documents and geospatial searches. *Enterprise context* (distributed indexing, load-balanced, centralized configuration).
Elasticsearch	2010	*More facet implementations*, such as statistical or date histogram facets. *Automatically detects the doc structure from JSON.*
Searchdaimon	2006	*Hit highlighting*, *dynamic clustering*, database integration, *rich document* (e.g., Word, PDF) handling and full document level security.
Bobo	2009	*No need for cache warm-up for the system to perform*, multi value sort, fast field value retrieval, facet count distribution analysis, stable and *small memory footprint*, support for runtime faceting results and a merge library for *distributed facet search.*
Open Semantic Search	2009	Interactive filters (faceted search), overview of aggregated search results and text mining), *tagging, annotation, dataviz, alerts*, multiple file formats and DBs, *text recognition, responsive design*, metadata management, files monitoring, *open semantic web*.
Constellio	2011	*Federated search of all data inside an organization.* A faceted search discovery tool, *automatic categorization of content*, *collaborative search engine* and management of the appearance order of results.

Table 4.1: Name, First release (measuring maturity) and main features of the discussed open-source solutions for faceted search, sorted by first release.

Name	First Release	Main Features
Oracle Endeca	1999	*Structured and unstructured information. Drag-and-drop.* Text analytics, *extracts key themes and sentiment* in multiple languages. *Data integration and ETL, integration with Oracle Business Intelligence, exploration and visualization*
Searchblox	2003	*Provides predictive analytics and predictive monitoring*. It enables the aggregation of multiple data sources. It provides visualization capabilities and real time tracking of changes.
LucidWorks	2007	*High accuracy of search results and integration with any data sources*. Components and services dedicated to: *analysis of big data, data visualization and real time monitoring*.
DataSax Enterprise	2010	*Log mining and social media match ups. Always-on search, live search, Solr compatible. Search/Solr syntax, Cassandra query language and SparkSQL* and *online elasticity*.
Searchify	2010	*Updates are immediately searchable*, location-aware. *Easier than Solr/Lucene/Sphinx in terms of configuration and usage*.

Table 4.2: Name, First release (measuring maturity) and main features of the discussed commercial solutions for faceted search, sorted by first release.

Next, we present a list of important keywords related with: automatic generation of facets and common algorithms used by search solutions.

4.4.1 Automatic Generation of Facets

In general the following strategies are implemented to allow the automatic generation of facets:

- *Field Value Faceting Parameters:*

  Facet field name, field prefix, contains, limit, offset, method

- *Range Faceting and Interval Faceting:*

  Count docs in interval, facet date ranges, start, end facet and var: mincount=1, to avoid empty results

- *Pivot Decision Tree Faceting and Local Parameters for Faceting (i.e., doc type):*

  Field Statistics and Aggregations (min, mean, max, stddev, sum, groups)

- *Machine Learning:*

  Classification, Clustering, Learning to Rank/on Graph, Information Extraction, Similarity Learning

4.4.2 Component Technologies on Core Solutions

The basic algorithms usually implemented into core search solutions, that are used to create faceted search interfaces, are as follows:

- *Query and Document Understanding:*

  Query Refinement and Block Analysis, Topic Modeling and Wrapper Generation, Title Extraction and Classification and BrowseRank

- *Query-Document Matching, Search Results Presentation and Others:*

  BM25 and Markov Random Field, Ranking and Search Result Clustering

  Crawling, Indexing, Anti-Spam, Search Log Mining

Chapter 5

Conclusion

This paper gave an introduction to the field of faceted user interfaces as well as a survey of solutions to implement it. In detail, the contributions of this survey are three-fold:

Firstly, it highlighted the motivation for faceted user interfaces and the problems that can be mitigated by using it. In general, faceted user interfaces address the problem of results sets that are either too large (unmanageable) or too small (empty) for users and provide a tool to narrow down result sets. State of the art search interfaces guide the user through the search process by means of facets (multiple orthogonal taxonomies). By selecting values for facets, result sets can be effectively filtered, while at the same time keeping the complexity of the interface low.

Secondly, this paper developed a classification to categorize the various factors that one must take into consideration when implementing a faceted search interface. The distinction between standard techniques based on free-text and faceted navigation approaches is important to understand how indeed faceted interfaces contribute to the creation of better user experiences. The four main aspects of the classification are (i) the focus of the user interfaces (on search or categories), (ii) the choices of widgets used to control the facets, (iii) the importance of the right placement of facet filters and other interface elements and (iv) the contrast between static and dynamic creation of facets.

Thirdly and finally, an overview of state of the art was given by discussing open source and commercial technologies that allow the setup of powerful faceted search solutions. The most important open-source solutions were Apache Solr and Elasticsearch, while the pioneering commercial application was Oracle Endeca. These technologies are usually supported and augmented by big data analysis and advanced machine learning implementations. This work surveyed the current search strategies that help guide users in the exploratory process of exploring and finding the right piece of information.

Bibliography

Amazon [2015a]. *amazon.com*. English. Dec 2015. http://www.amazon.com/ (cited on page 7).

Amazon [2015b]. *Search for SSD on amazon.com*. English. Dec 2015. http://www.amazon.com/s/ref=nb_sb_noss?url=search-alias=aps&field-keywords=SSD (cited on pages 1, 2).

Andrews, Keith [2015]. Lecture Notes for Information Architecture and Web Usability, Graz University of Technology. 2015 (cited on pages 4, 5).

Apache [2015a]. *Apache Lucene Core*. English. Apache. Nov 2015. https://lucene.apache.org/core/ (cited on pages 15, 19).

Apache [2015b]. *Apache OODT*. English. Dec 2015. https://oodt.apache.org/ (cited on page 15).

Apache [2015c]. *Apache Tika*. English. Dec 2015. https://tika.apache.org/ (cited on page 15).

Bates, Marcia J. [1989]. "The Design of Browsing and Berrypicking Techniques for the online search interface". *Online Information Review* 13.5 (1989), pages 407–424 (cited on page 3).

Bobo [2015]. *Bobo*. English. Nov 2015. http://javasoze.github.io/bobo/ (cited on page 17).

Capra, Robert G and Gary Marchionini [2008]. "The relation browser tool for faceted exploratory search". In: *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*. ACM. 2008, pages 420–420 (cited on page 5).

Cassandra, Apache [2015]. *Apache Cassandra*. English. Nov 2015. http://cassandra.apache.org/ (cited on page 19).

Constellio [2015]. *Constellio*. English. Nov 2015. http://constellio.com/?lang=en (cited on page 16).

Elastic [2015]. *Elastic*. English. Nov 2015. https://www.elastic.co/ (cited on pages 16, 19).

Endeca, Oracle [2015]. *Oracle Endeca*. English. Nov 2015. http://www.oracle.com/us/solutions/business-analytics/business-intelligence/endeca/overview/index.html (cited on page 17).

Enterprise, DataSax [2015]. *DataSax Enterprise*. English. Nov 2015. http://www.datastax.com/products/datastax-enterprise-search (cited on page 18).

ES, Searchdaimon [2015]. *Searchdaimon ES*. English. Nov 2015. http://www.searchdaimon.com/ (cited on page 16).

Whitenton, Kathryn [2014]. http://www.nngroup.com/articles/filters-vs-facets/. 16th Mar 2014.

Flickr [2015]. *Flickr*. Dec 2015. http://flickr.com (cited on page 12).

Furnas, George W. et al. [1987]. "The vocabulary problem in human-system communication". *Communications of the ACM* 30.11 (1987), pages 964–971 (cited on page 5).

Fusion, Lucidwork [2015]. *Lucidwork Fusion*. English. Nov 2015. `https://lucidworks.com/products/fusion/` (cited on page 17).

Geizhals [2015]. *geizhals.at*. 2015. `http://geizhals.at` (cited on page 11).

Gifford, Jon [2014]. *Why Loggly Chose ElasticSearch Over Solr*. English. Jul 2014. `https://www.loggly.com/blog/loggly-chose-elasticsearch-reliable-scalable-log-management/` (cited on page 19).

Hearst, Marti [2006]. "Design recommendations for hierarchical faceted search interfaces". In: *ACM SIGIR workshop on faceted search*. Seattle, WA. 2006, pages 1–5 (cited on pages 7, 13).

Hearst, Marti A [2006]. "Clustering versus faceted categories for information exploration". *Communications of the ACM* 49.4 (2006), pages 59–61 (cited on page 15).

Hearst, Marti A. [2009]. *Search User Interfaces*. 1st edition. Cambridge University Press, 2009. ISBN 0521113792. `searchuserinterfaces.com` (cited on pages 3–5).

Holst, Christian [2015]. *Current state of e-commerce filtering*. English. Smashing Magazine. Apr 2015. `http://www.smashingmagazine.com/2015/04/the-current-state-of-e-commerce-filtering/` (cited on page 11).

Huynh, David F and David Karger [2009]. "Parallax and companion: Set-based browsing for the data web". In: *WWW Conference. ACM*. Citeseer. 2009, page 6 (cited on page 5).

Jeffery Callender, Peter Morville [2015]. *A list apart - Design Patterns: Faceted Navigation*. English. Dec 2015. `http://alistapart.com/article/design-patterns-faceted-navigation` (cited on page 8).

Karam, Maria, Shengdong Zhao et al. [2003]. "mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia" (2003) (cited on page 5).

Kershaw [2015]. *Solr example (autocomplete, ranking and custom results view configuration: list vs grid): Kershaw Webstore*. Dec 2015. `https://kershaw.kaiusaltd.com/knives` (cited on page 15).

Kleinberg, Jon M [1999]. "Authoritative sources in a hyperlinked environment". *Journal of the ACM (JACM)* 46.5 (1999), pages 604–632 (cited on page 3).

Nordstrom [2015]. *nordstrom.com*. Dec 2015. `http://nordstrom.com` (cited on page 11).

Opensemanticsearch [2015]. *Opensemanticsearch*. English. Nov 2015. `http://www.opensemanticsearch.org/` (cited on page 17).

Page, Lawrence et al. [1999]. "The PageRank citation ranking: bringing order to the Web." (1999) (cited on page 3).

Pirolli, Peter [2007]. *Information Foraging Theory: Adaptive Interaction with Information*. Oxford University Press, 2007 (cited on page 3).

Raghavan and Schatze [2008]. "Introduction to Information Retrieval" (2008) (cited on page 3).

Ranganathan, Shiyali Ramamrita [1969]. "Colon classification" (1969) (cited on page 4).

Rosenfeld, Louis and Peter Morville [2002]. *Information Architecture for the World Wide Web*. 2nd edition. O'Reilly Media, 2002 (cited on page 4).

Roy, Senjuti Basu et al. [2009]. "Dynacet: Building dynamic faceted search systems over databases". In: *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE. 2009, pages 1463–1466 (cited on page 19).

Searchblox [2015]. *Searchblox*. English. Nov 2015. `http://www.searchblox.com/` (cited on page 18).

Searchify [2015]. *Searchify*. English. Searchify. Nov 2015. `https://www.searchify.com/` (cited on page 18).

Solr [2015]. *Solr*. English. Nov 2015. `http://lucene.apache.org/solr/` (cited on pages 15, 19).

Sphinx [2015]. *Sphinx*. English. Nov 2015. `/sphinxsearch.com/` (cited on page 16).

Sutcliffe, Alistair and Mark Ennis [1998]. "Towards a cognitive theory of information retrieval". *Interacting with computers* 10.3 (1998), pages 321–351 (cited on page 3).

Teevan, Jaime et al. [2004]. "The perfect search engine is not enough: a study of orienteering behavior in directed search". In: *CHI'04*. ACM. 2004, pages 415–422 (cited on page 4).

Today, PredictiveAnalytics [2015]. *Top open-source big-data enterprise search software*. English. Nov 2015. `http://www.predictiveanalyticstoday.com/top-open-source-big-data-enterprise-search-software/` (cited on page 15).

Tunkelang, Daniel [2009]. "Faceted Search". *Synthesis Lectures on Information Concepts, Retrieval, and Services* 5 (2009). doi:10.2200/S00190ED1V01Y200904ICR005 (cited on pages 1, 3–5, 7, 10).

Tutorial, Lucene [2015]. *Lucene vs Solr*. English. Dec 2015. `http://www.lucenetutorial.com/lucene-vs-solr.html` (cited on page 19).

Volkswagen [2015]. *Volkswagen*. Dec 2015. `http://vw.com` (cited on page 12).

Wang, Haidong [2008]. *DynaCet: A Minimum-effort Driven Dynamic Faceted Search System Over Structured Databases*. ProQuest, 2008 (cited on page 19).

Willhaben [2015]. *willhaben.at*. English. Dec 2015. `http://www.willhaben.at/` (cited on page 7).

Zheng, Bweijunl, Wei Zhang and Xiaoyu Fu Boqin Feng [2013]. "A survey of faceted search". *Journal of Web engineering* 12.1&2 (2013), pages 041–064 (cited on page 15).

Zimmer, Bianca, Romina Kuehn and Thomas Schlegel [2014]. "A Visualization Concept for Mobile Faceted Search". In: *Human Interface and the Management of Information. Information and Knowledge Design and Evaluation*. Springer, 2014, pages 128–136 (cited on page 12).