Browsing-oriented Semantic Faceted Search[⋆]

Andreas Wagner, Günter Ladwig, Thanh Tran

Institute AIFB, Karlsruhe Institute of Technology, Germany
{a.wagner,guenter.ladwig,ducthanh.tran}@kit.edu

Abstract. Faceted search enables users to browse and discover relevant items from a large collection such as the Web of data. Existing faceted search solutions assume a precise information need, and thus optimise relevance, interestingness, and costs of fulfilling an information need. In this paper, we propose a complementary solution. Instead of assuming a *search scenario* (i.e., a user has a precise information need), our solution targets a *browsing scenario* (i.e., a user has a fuzzy need). We aim to support users in exploring an unknown collection of items, thereby allowing them to discover new or unfamiliar items of interest. Our approach comprises mechanisms for *grouping facets and facet values* and facet *ranking.* Via a task-based evaluation, we demonstrate that the proposed solution enables more effective browsing compared to the state-of-the-art, given fuzzy information needs.

1 Introduction

Recently, large amounts of structured data have been made publicly available on the Web (e.g., RDFa[1] or Linked Data[2]), allowing complex information needs to be addressed. For instance, consider the following example: Susan is a novice computer science student. She is eager to learn more about this vast research field and wishes to find "information about work of prestigious computer scientists". With traditional Web search, Susan searches via keywords and browses via hyperlinks to fulfil her information need. Observe the two paradigms in Susan's example: (1) *search* as a mean for goal-oriented retrieval of information (e.g., via keyword-based lookups) and (2) *browsing* as a mean for iterative exploration of a collection of items (e.g., via hyperlinks) [18, 17].

Searching Web data using structured query languages (e.g., SPARQL[3]) helps to address complex information needs (e.g., Susan's need). However, in order for such a goal-oriented search to be effective, users have to be familiar with the query language. Further, users have to know the item of interest and the underlying domain. Thus, the search paradigm allows for *precise* information needs only. However, real-world information needs are often *fuzzy.* There are two dimensions of fuzziness: (1) Users have *vague knowledge about the domain.* For instance, Susan cannot precisely describe the term "prestigious", whereas a domain expert (having precise knowledge) may look for researchers that won a Turing Award. (2) Users

⋆ Supported by the German Federal Ministry of Education and Research in the *CollabCloud* (grant 01IS0937A-E) and *iGreen* (grant 01IA08005K) project.

1 http://www.w3.org/TR/xhtml-rdfa-primer/

2 http://www.w3.org/DesignIssues/LinkedData.html

3 http://www.w3.org/TR/rdf-sparql-query/

only *vaguely know the item of interest.* For instance, "work" in our example may refer to publications or projects.

The browsing paradigm is more suitable for dealing with fuzzy needs. It does not assume users to have full knowledge regarding domain or item of interest. Instead, browsing allows users to explore a collection of items iteratively [18, 17]. For instance, Susan may start with a simple lookup search (e.g., a keyword query "computer scientist") to obtain some starting points and then browses the remaining collection to find "prestigious" scientists.

Faceted search implements the browsing paradigm, representing a promising approach towards exploring and addressing (possibly fuzzy) information needs [13, 5, 11]. Here, users explore a collection of items by browsing conceptual dimensions of the items (i.e., facets) and their values (i.e., facet values) [10, 21]. During an iterative process of selecting facets and refining the current result collection, users may construct complex, structured queries.

Related to faceted search is work on visual query builders [7, 22]. However, while the latter focuses solely on intuitive means for query construction, faceted search aims at exploration and understanding (during a process of iterative query reformulation).

State-of-the-art. Faceted search was proposed for querying documents [10, 9, 4], databases [8, 3, 2] and semantic data [19, 23, 14] (referred to as *semantic faceted search*). One research direction is concerned with *efficiency* aspects. Existing work includes indexes and algorithms for fast computation of facets and facet-related data [4, 9]. In this paper, we are concerned with the *effectiveness* of faceted search – efficiency aspects are orthogonal and unfortunately out of scope.

Given a large amount of facets associated with a collection of items, one major challenge we address is *facet ranking.* Widely used is *frequency-based ranking* [8, 20, 16]. It considers the number of items that are associated with a facet (its count). A facet is considered important, when its count is high. Based on the same idea, *set-cover ranking* has been proposed [8], which aims to maximise the number of distinct items that are accessible from the top-k facets. In [16], the authors assume a *relevance-based* ordering of items and propose ranking facets according to their likelihood of being associated with a relevant item. Further, the notion of *interestingness* has been incorporated into ranking [9], suggesting that facet relevance may be measured based on how surprising a facet is (given a certain expectation). The interestingness of a facet is defined as the aggregation of the interestingness of its facet values, which is based on rationales for what should be an expected facet value. In [8], the objective is to minimise *user costs* for finding a specific item of interest. Cost is defined as the time needed for reaching an item of interest. This time is computed as an aggregation of the times for reading facet headings, for browsing facet hierarchies and for correcting browsing mistakes. The authors of [3] propose to use the facet hierarchy (which the user traverses), as an approximation for the interaction time and cost, respectively. A ranking scheme is introduced to prefer facets with a hierarchy of low height. Thus, facets are ranked high, when they quickly lead to an item of interest [3].

For effective faceted search, besides facet ranking, facet grouping approaches have been proposed. A *facet tree* (i.e., a tree-shaped facet grouping) was employed in [8, 13, 5, 11]. Thus, users are able to browse multiple facets (forming a facet path), in order to explore a collection of items.

Contributions. We observe that (except for the generic, frequency-based ranking), existing ranking approaches assume a precise information need. That is, relevance, interestingness, and user costs (for fulfilling an information need) have been employed for measuring facet importance. Thus, we refer to such approaches as *search-oriented*. However, we propose a complementary approach, targeting a browsing scenario. Our solution supports users in addressing fuzzy needs, by enabling them to slowly explore an unknown collection of items. In particular, we provide mechanisms for grouping facets and facet values (i.e., an extended facet tree), as well as for ranking facets – both targeting an enhanced browsing experience. The contributions can be summarised as follows:

- For browsing facets with a large number of facet values, we propose an *extended facet tree*, which compactly captures both facets and facet values.
- We propose a ranking scheme, which supports users, given a fuzzy information need, in browsing a collection of items.
- We have implemented our approach and made the code[4] freely available. Further, we have conducted a task-based evaluation, showing that our approach outperforms the state-of-the-art on fuzzy information needs.

Outline. In Section 2, we introduce the data, query and facet model. Section 3 discusses (large) facet value sets and an extended facet model for browsing such sets. Facet ranking is discussed in Section 4. In Section 5, we present a task-based evaluation. We conclude with Section 6.

2 Data, Query and Facet Model

Data and Query Model. As different types of structured Web data may be represented as graphs (including RDF), we employ a general graph-structured data model [24].

Definition 1 (Data Graph). *Let $\mathcal{L}_V$ and $\mathcal{L}_E$ be finite sets of vertex and edge labels respectively. A data graph is a tuple $\mathcal{G} = (\mathcal{V}^G, \mathcal{E}^G)$, where $\mathcal{V}^G$ is a finite set of vertices, $l_V : \mathcal{V}^G \mapsto \mathcal{L}_V$ is a vertex labelling function and $\mathcal{E}^G \subseteq \mathcal{L}_E \times l_V(\mathcal{V}^G) \times l_V(\mathcal{V}^G)$ is a set of labelled edges. The set of vertices is conceived as the disjoint union $\mathcal{V}^G = \mathcal{V}^G_E \uplus \mathcal{V}^G_D$, where $\mathcal{V}^G_E$ stands for* entities *and $\mathcal{V}^G_D$ are data values. We distinguish the set of* relation edges $\mathcal{E}^G_R = \{e(v_i, v_j) \in \mathcal{E}^G | v_i, v_j \in \mathcal{V}^G_E\}$ *from the set of* attribute edges $\mathcal{E}^G_A = \{e(v_i, v_j) \in \mathcal{E}^G | v_i \in \mathcal{V}^G_E, v_j \in \mathcal{V}^G_D\}$ *and $\mathcal{E}^G = \mathcal{E}^G_R \uplus \mathcal{E}^G_A$.*

Information needs in our setting correspond to conjunctive queries of the form $(x_1, \ldots, x_k).\exists x_{k+1}, \ldots, x_m.e_1 \wedge \ldots \wedge e_r$, where e_i are atomic formulae and $x_1, \ldots, x_k$ and $x_{k+1}, \ldots, x_m$ are called *distinguished* and *undistinguished variables*, respectively [24]. We focus on conjunctive queries with atomic formulae of the form $e(v_i, v_j)$, where v_i and v_j are either *variables* or *constants*. Since variables of a conjunctive query may interact in an arbitrary way, these formulae form a graph (so called *basic graph patterns*, representing a core feature of SPARQL). Further, a conjunctive query is denoted as $\mathcal{Q} = (\mathcal{V}^Q, \mathcal{E}^Q)$. Vertices of $\mathcal{Q}$ are $\mathcal{V}^Q = \mathcal{V}^Q_V \uplus \mathcal{V}^Q_C$ comprising a set of variables $\mathcal{V}^Q_V$ and constants $\mathcal{V}^Q_C \subseteq \mathcal{V}^G$. Edges of $\mathcal{Q}$ (called query predicates) are formulae $e(v_i, v_j)$, with $v_i \in \mathcal{V}^Q_V, v_j \in \mathcal{V}^Q$. A conjunctive

[4] `http://code.google.com/p/semanticfacetedsearch/`

query $\mathcal{Q}$ is processed as a graph pattern. Specifically, a result to $\mathcal{Q}$ on a graph $\mathcal{G}$ is a mapping from vertices of $\mathcal{Q}$ to vertices of $\mathcal{G}$, such that the substitution of variables (called variable bindings, denoted by $\mathcal{V}_x^R$ with $x \in \mathcal{V}_V^Q$) in $\mathcal{Q}$ would yield a subgraph of $\mathcal{G}$. Thus, every result represents a subgraph of $\mathcal{G}$. In fact, the entire set of results is a subgraph of $\mathcal{G}$, denoted by $\mathcal{R}(\mathcal{V}^R, \mathcal{E}^R)$ (called result set) [24].

Facet Model. Our conjunctive query model indicates the information needs we aim to support. However, via faceted search, users do not directly operate on this query model, but employ facets to construct queries [10]. With semantic faceted search [12, 1, 19, 11], conjunctive queries may be constructed. To formalise the ideas of semantic faceted search, we employ a facet model comprising three components: (1) facets, (2) facet values and (3) facet operations.

Definition 2 (Facets). *Let $\mathcal{Q}(\mathcal{V}^Q, \mathcal{E}^Q)$ be the query, $\mathcal{R}(\mathcal{V}^R, \mathcal{E}^R)$ be the result set for $\mathcal{Q}$ and $\mathcal{V}_x^R \subseteq \mathcal{V}^R$ be the particular set of bindings obtained for the variable $x \in \mathcal{V}_V^Q$. Facets $F(x)$ (for the variable x) are labels of edges, which capture direct connections between elements in $\mathcal{V}_x^R$ and other elements of the data graph, i.e., $F(x) = \{f | f(v_i, v_j) \in \mathcal{E}^G, v_i \in \mathcal{V}_x^R\}$. Facets can be associated with every variable $x \in \mathcal{V}_V^Q$. The set of facets for $\mathcal{Q}$ is $F(\mathcal{Q}) = \{F(x) | x \in \mathcal{V}_V^Q\}$.*

Facets can seen conceived as conceptual dimensions of some particular variable bindings. In particular, every facet $f \in F(x)$ corresponds either to a relation or an attribute edge label. Thus, values of f might be entities or data values:

Definition 3 (Facet Values). *Let $\mathcal{R}(\mathcal{V}^R, \mathcal{E}^R)$ be the result set and $\mathcal{V}_x^R \subseteq \mathcal{V}^R$ be the bindings for a query variable x, then the values of a facet $f \in F(x)$ are entities or data values that are directly connected to elements in $\mathcal{V}_x^R$ via f, i.e., $FV(f) = \{v_j | f(v_i, v_j) \in \mathcal{E}^G, v_i \in \mathcal{V}_x^R\}$.*

There are three operations on facets that can be used to construct queries, i.e., to modify the bindings of variables $\mathcal{V}_{var}^R$ and thus, to modify the overall result set $\mathcal{R}$. These operations are: (1) *focus selection*, (2) *refinement* and (3) *expansion.*

With focus selection, users can change the focus to the variable (and thereby the set of bindings) they wish to modify. For instance, changing focus from y to x means to focus on facets contained in $F(x)$ (i.e., to focus on the entity set $\mathcal{V}_x^R$) instead of $F(y)$ (the entity set $\mathcal{V}_y^R$). In technical terms, it means that in faceted search, we have only one distinguished variable, which during the process, can be changed by the user to obtain different sets of results for refinement and modification.

Users can modify the set of bindings for the variable in focus by adding further query predicates. In particular, a refinement operation performed on a variable x (on the entity set $\mathcal{V}_x^R$, respectively) means adding a new query predicate $f(x, y)$, with $f \in F(x)$ and y as new variable. Instead of adding further query predicates, a refinement can also be performed by modifying an existing query predicate. Let $f \in F(x)$ be a facet corresponding to the query predicate $f(x, y)$ and let $FV(f)$ be its facet value set. Users can refine $\mathcal{V}_x^R$ by choosing a facet value $v \in FV(f)$, in order to obtain a subset of $\mathcal{V}_x^R$ containing only entities connected to v via f. This refinement operation (denoted by $(f : v)$) replaces y in $f(x, y)$ with a constant v. Analogously, users may expand a result set by removing a facet (i.e., removing a query predicate) or removing a facet value (i.e., replacing a constant with a variable).

3 Browsing-oriented Facet and Facet Value Spaces

In this section, we propose an extension of our facet model – the notion of a facet tree. For a result set, the basic facet tree (FT) compactly represents the space of all facets [12, 1, 19, 11], while our extended facet tree (FT_e) additionally also captures the space of all facet values.

Facet Tree (FT). First, let us define the basic facet tree. For that, we introduce a *browsing* operation, which allows users to explore the facet (and later also the facet value) space via navigation along facets. Analogous to the expansion and refinement operation, browsing consists of (multiple) facet selections. However, facets selected during browsing are not evaluated, i.e., the underlying query does not change and thus the result set is not modified. A sequence of browsing operations allows users to navigate from the result set to associated facet values, and via their facets, to facet values that are further away. Every such browsing sequence establishes a facet path. All possible facet paths, which may result from browsing, establish a tree of facets:

Definition 4 (Facet Tree). *Let $\mathcal{G}(\mathcal{V}^G, \mathcal{E}^G)$ be the data graph and $\mathcal{V}^R_x \subseteq \mathcal{V}^R$ be the binding set (for a query $\mathcal{Q}$) for $x \in \mathcal{V}^Q_V$, then the facet tree $FT(x)$ for x can be conceived as a set of possibly overlapping paths P. Each $p \in P$ is of the form $\langle \mathcal{V}^R_x, \ldots, \mathcal{V}^L_l \rangle$, connecting the root node $\mathcal{V}^R_x$ with a leaf node $\mathcal{V}^L_l$. While leaves $\mathcal{V}^L$ are sets of data values, every other set $\mathcal{V}_i \in p$ comprises entities. There is a path $p \in P$ if and only if we find $\mathcal{V}^R_x, \ldots, \mathcal{V}^L_l \subseteq \mathcal{V}^G$ and $(\exists v_1 \in \mathcal{V}^R_x, \ldots, \exists v_l \in \mathcal{V}^L_l)$. $\exists e_1 \in \mathcal{E}^G, \ldots, \exists e_l \in \mathcal{E}^G.\ e_1(v_1, v_2) \wedge \ldots \wedge e_l(v_{l-1}, v_l)$.*

A facet tree FT is derived from vertices and edges in the data graph. In particular, FT captures all entities and data values reachable from the result set via navigation along paths in the data graph. It can be constructed via breadth-first search from the result set (the root) to data values (the leaves). Note, in order to include an entity v with no outgoing edges, we add a new edge $e(v, l_V(v))$ (i.e., an edge pointing to its label).

In Fig. 1 we illustrate an exemplary tree for a result set containing four professors. From the professor entities, it is possible to navigate to their associated names, their universities and the university's age.

Note, our notion of a facet tree is slightly different from existing facet trees and graphs. In Parallax [13] and Tabulator [5], it is possible to traverse tree-like structures, while with gFacet [11] even graph structures can be explored. With such systems, users can navigate from one entity set to another by changing the focus, and refine (expand) it by adding (removing) facets. In effect, users navigate though the data. As opposed to that, we explicitly employ a browsing operation, allowing users to explore the data without modifying the underlying query or changing the focus. The user does not browse through the data, but though a compact description, the facet tree. Users only see facets (e.g., $works\ at$, age and $name$) and (in the extended facet tree) labels of facet value sets (e.g., $University$ or $[ann - paul]$). Thus, browsing is based on a compact intensional description, thereby allowing users to easily grasp the overall structure of the facet (facet value) space. In order modify the result set via facet paths, we introduce an extension of the refinement operation. Instead of adding one query predicate, users can now add a conjunction of query predicates that corresponds to a facet path $\langle e_1, \ldots, e_l \rangle$

in FT, selected by the user (e.g., $\langle works\ at, age\rangle : [70 - 300]$). Analogously, we allow an extended expansion that removes a conjunction of query predicates.

Extended Facet Tree (FT_e). Through browsing along the paths of a facet tree, users eventually will reach a set of data values. Given a large set of data values, users might be overwhelmed. Thus, current systems choose to present only few top-ranked facet values and cut off the rest [19, 23, 14]. However, selecting one single value requires users to have precise needs and knowledge regarding that facet, which is contrary to the browsing paradigm. We aim to enable users to explore and understand the entire facet value space.

Therefore, we apply a *divisive hierarchical clustering* technique [15]. Using our clustering technique, a set of facet values (i.e., data values) $FV(f)$ is recursively split, resulting in a hierarchy of data value clusters. Since clusters at the same level do not overlap, the clustering process amounts to a partitioning of $FV(f)$. In order to decide where to split a cluster, a measure of dissimilarity between sets of data values is required. We use a distance measure for capturing the dissimilarity between two data values and employ a linkage criterion, which captures the dissimilarity of sets as a function of the pairwise dissimilarity of data values (contained in those sets). More precisely, we use the Euclidean distance for numerical data values and the Levenshtein distance for textual data values. For computing the dissimilarity between sets of data values, we use the *single-linkage* criterion, where the distance between two sets of data values is defined as the minimum distance of all pairs of data values from both sets [15].

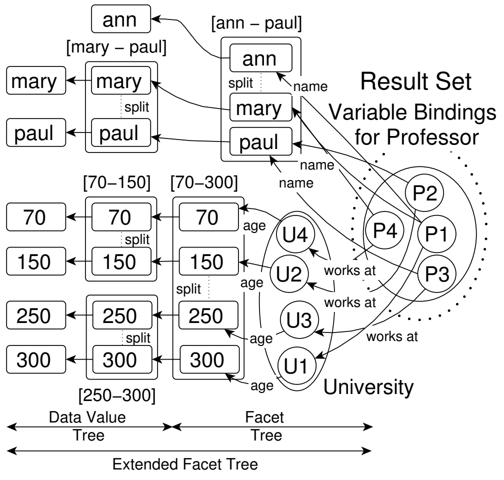

Fig. 1. Susan's (extended) facet tree

Now, we employ the clusters to extend the facet tree. Leaf nodes $\mathcal{V}_i^L \in FT(x)$ containing more data values than a given threshold are clustered, resulting in a set of *data value trees* (DT). The combination of facet tree and data value trees form an *extended facet tree* (FT_e). Note, clustering is performed only on demand (i.e., upon users browsing behaviour). More precisely, whenever a user reaches a facet tree leaf, associated facet values are clustered, and a data value tree is attached to extend FT.

Fig. 1 illustrates an exemplary extended facet tree. For instance, the set of names $\{ann, mary, paul\}$ is clustered, resulting in a tree of data values with $[ann-paul]$ as root. At the second level of the data tree, the set $[ann - paul]$ is split into two sets: $\{ann\}$ and $[mary - paul]$.

Compared to the state-of-the-art, the extended facet tree allows users to browse and explore the data based on a compact and hierarchical representation, using both facets (contained in FT) and facet value sets (contained in DT).

4 Browsing-oriented Facet Ranking

Current work on facet ranking assumes users to have precise needs (i.e., know the domain and item of interest) and thus, relevance, interestingness, or user costs have been employed as measures [8, 3, 9]. In this section, we present a facet ranking scheme targeting a different scenario. We assume users to have incomplete knowledge w.r.t. domain or item of interest (i.e., fuzzy need). Thus, such users need support in exploring and understanding the result set. In particular, we prefer facets that allow users to modify the result set via small and uniform facet operations.

4.1 Intuitions and Metrics for Browsing-oriented Facet Ranking

For ranking a facet $f \in F(x)$, we consider the facet and facet value space that can be reached via f and result set modifications, which can be performed via facet paths originating from f.

More precisely, we consider f's extended facet tree: For facet f (representing the query predicate $f(x, y)$) we use $FT_e(y)$ to capture the facet and facet value space reachable via f. That is, $FT_e(y)$ captures all facet paths that can be used to modify $\mathcal{V}_x^R$ ($\mathcal{V}^R$). We will now discuss the intuitions behind browsing-oriented facet ranking and concrete metrics used to measure them.

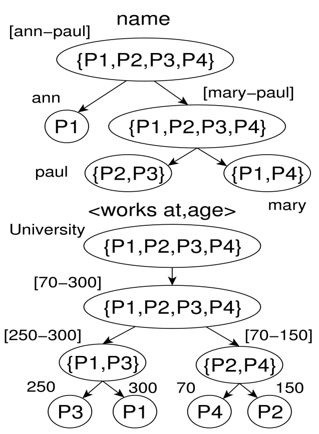

Fig. 2. Binding segments associated with facet *name* and *works-at*

Small Steps. Users modify the result set until reaching an item of interest. Given that the facet paths (leading to an item of interest) are unknown and have yet to be explored, major result modifications that quickly change the result set are likely to lead to mistakes (i.e., lead to irrelevant results). Via small result modifications, users get to know the result set bit by bit. These small changes can be comprehended more easily by users (thus, they are less likely to choose paths that lead to irrelevant results). We use two metrics to implement this intuition:

- **Maximum Height (h).** The maximum height of FT_e (i.e., the maximum edge distance between the root node and a leaf node), directly reflects the maximum number of possible facet operations. Given the number of current results are fixed, the higher the number of possible result modifications, the smaller are the average changes resulting from each result modification. Thus, the greater the height of FT_e of f, the higher we rank f. In our example, the tree associated with *name* has height $h = 2$, while *works at* has a tree with $h = 3$. Thus, we prefer *works at* w.r.t. height.
- **Minimum Branching Factor (b).** The branching factor measures the number of nodes, which FT_e has at a particular level. Trees with small branching factor lead to smaller result modifications, as such trees tend to be higher. Further, a small branching factor reflects a small number of possible choices at every level in FT_e. Compared to a situation with a larger number of (necessarily more fine-grained) choices, this situation is easier for the user to cope

with (as it does not require specific knowledge for making a decision). Therefore, we prefer facets having facet trees associated with a smaller branching factor. For instance, *name* and *works at* have the same rank in this regard because they have the same branching factor. At every level of both trees, Susan faces only two choices.

Uniform Steps. We consider query modifications to be non-uniform, when they have varying impacts on the result set size. More precisely, one query modification might strongly favour one particular segment of the result set and discriminate other segments. However, given the lack of precise knowledge about the item of interest, all results are a priori of equal importance. Thus, it is not possible to prefer a query modification that leads to a smaller set of results over another resulting in a larger set of results. Likewise, longer facet paths cannot be preferred over shorter paths. When browsing, it is hard for users to choose between these non-uniform query modifications. Such query modifications are rather confusing and likely lead to irrelevant results. Consequently, trees containing uniform query modification steps shall be preferred. We use metrics as follows:

– **Height Balance (hb).** FT_e is perfectly height balanced, when all leaves are of equal edge distance to the root. We define height balance as $hb(FT_e) = \frac{c}{(dist_{max}(FT_e)+\epsilon)-dist_{min}(FT_e)}$, with c being a constant and $dist_{max}$ ($dist_{min}$) as maximal (minimal) edge distance from the root to a leaf. A facet with FT'_e associated is ranked higher than one with FT''_e, iff $hb(FT'_e) > hb(FT''_e)$. For instance, the facet tree associated with *name* is less height-balanced than the tree associated with *works at*.

– **Facet Value Set (sb_{fv}) and Binding Segment (sb_b) Size Balance.** We measure the size balance w.r.t. facet value sets and binding set segments, which may be reached via FT_e. Note that facet value sets are captured by leaf nodes $\mathcal{V}^L$ of FT_e. Let the leaf $\mathcal{V}^L_{max}$ ($\mathcal{V}^L_{min}$) contain the largest (smallest) number of facet values. The facet value set balance is $sb_{fv}(FT_e) = \frac{c}{(|\mathcal{V}^L_{max}|+\epsilon)-|\mathcal{V}^L_{min}|}$, with c being a constant. Further, every refinement operation (corresponding to a node $\mathcal{V}_i \in FT_e(x)$) actually leads to a binding segment $\mathcal{V}^R_{x_i} \subseteq \mathcal{V}^R_x$. Thus, corresponding to the facet tree, we have a tree of binding segments. Fig. 2 illustrates the trees of binding segments corresponding to the facet trees associated with *name* and *works at*. We consider the size of the binding segments at leaf level. For instance, *works at* has four binding segments at leaf level ($\{P1\}, \ldots, \{P4\}$); *name* has $\{P2, P3\}$ and $\{P1, P4\}$. Our variable binding segment size balance is $sb_b(FT_e) = \frac{c}{(|\mathcal{V}^A_{x_{max}}|+\epsilon)-|\mathcal{V}^A_{x_{min}}|}$, where c is a constant and $\mathcal{V}^A_{x_{max}}$ ($\mathcal{V}^A_{x_{min}}$) is the largest (smallest) variable binding segment (at leaf level). While the binding segment size is perfectly balanced for *works at*, this is not the case for *name* (one segment contains one professor, while the others contain two).

Comprehensible Result Segments. For users who are unfamiliar with a result set, it is important that a facet operation leads to obvious and comprehensible result modifications. Each operation should lead to a true result modification, i.e., an observable result refinement (expansion). Further, different operations should lead to different result modifications. Our metrics are:

– **Binding Distinguishability (d).** To assess whether facets lead to an observable result modification, we use the notion of distinguishability adopted from [3]. A facet has a high distinguishability, when it leads to facet values that precisely identify variable bindings. Ideally, leaves $\mathcal{V}^L$ are associated with binding segments consisting of exactly one element. Accordingly, our distinguishability metric is $d(FT_e) = \frac{|\mathcal{V}_x^R|}{\sum_{\mathcal{V}_i \in \mathcal{V}^L} |\mathcal{V}_{x_i}^R|}$. For instance, facet *name* leads to *mary*, which is associated with $\{P1, P4\}$. *Paul* is the name of $\{P2, P3\}$. Both pairs of professors share the same name, so it is difficult for Susan to distinguish them. Using the facet *works at*, she is able to distinguish between all four professors, as they work for different universities. Overall, *works at* has maximum distinguishability of 1, while the distinguishability for *name* is 0.8.

– **Minimal Binding Segment Overlap (o).** Binding segments with minimal overlaps are preferred to ensure that facet operations along a tree FT_e lead to different result modifications. Binding set overlap can be computed by considering the binding segment overlap at leaf level: $o(FT_e) = \frac{|\mathcal{V}_x^A|}{|\bigcap_{\mathcal{V}_i \in \mathcal{V}^L} \mathcal{V}_{x_i}^A| + \epsilon}$. In our example, the refinements ($name : ann$), ($name : mary$) and ($name : paul$) split the binding set into segments that overlap on $\{P1\}$ (as $P1$'s name includes *ann* and *mary*). Refinements via ($\langle works\ at, age \rangle : 70$) etc. split the binding set into segments with no overlaps. Thus, Susan can observe two sets of professors, i.e., one set $\{P2, P4\}$ working at older universities, while the other set $\{P1, P3\}$ is associated with younger universities.

4.2 A browsing-oriented Ranking Function

We now provide a scoring function $\mathcal{S}$, which incorporates the proposed metrics. We aim to rank a facet $f \in F(x)$, where f represents $f(x, y)$, based on its facet tree $FT_e(y)$. We distinguish facets that correspond to attributes from facets corresponding to relations.

Definition 5 (Attribute-based Scoring Function). *Given an attribute facet f, its facet tree FT_e, the score of f is defined as $\mathcal{S}(f) = ag(h(FT_e), b(FT_e), hb(FT_e), sb_{fv}(FT_e), sb_b(FT_e), d(FT_e), o(FT_e))$, with ag as a monotonic aggregation function.*

The set of attribute (relation) facets at level k that can be reached via the facet tree FT is denoted by $F_f^A(k)$ ($F_f^R(k)$). The score of a relation facet f at level k is computed based on the scores of facets $F_f^R(k+1)$ and $F_f^A(k+1)$.

Definition 6 (Relation-based Scoring Function). *Let f be a relation facet at depth $k = 0$ and let F_f^A (F_f^R) be the set of directly connected attribute (relation) facets (i.e., $F_f^A(1)$ and $F_f^R(1)$), k_{do} is the total edge distance to be considered, k_{did} is the edge distance considered so far, and $\delta(k)$ is a monotonic decreasing weight function discounting the score of facets more distant from f, then the score of f is recursively computed using the formula (starting at $k_{did} = 1$ and $k_{do} \geq 1$):*

$$\mathcal{S}(f) = \begin{cases} \delta(k_{did}) ag_{f_a \in F_f^A} \mathcal{S}(f_a) & if\, k_{do} = 1 \\ ag_{f_r \in F_f^R} \mathcal{S}(f_r)_{k_{do}-1, k_{did}+1} + \delta(k_{did}) ag_{f_a \in F_f^A} \mathcal{S}(f_a) & otherwise \end{cases}$$

In the current implementation, ag is a summation. We use $k_{do} = 1$, i.e., the score of a relation facet is simply an aggregation of the scores of reachable attribute facets.

5 Evaluation

We decided to conduct a task-based evaluation, which has gained acceptance in the IR community, especially for assessing approaches that go beyond IR-style document retrieval[5]. The goal is to find out whether browsing (as supported by our approach) helps to accomplish a set of predefined tasks (effectiveness) and how much time is needed (efficiency).

Participants. 24 participants took part in our evaluation: 6 were non-technical users, while the remaining 18 participants had a computer science background. All were familiar with faceted search (as used in Web search engines). The participants were given an introduction to the system, similar to an available screencast.[6]

Tasks. 24 tasks were chosen by domain experts and comprised both precise and fuzzy information needs. Tasks were followed by a series of questions to assess the user's understanding and exploration of the result set. A complete listing of tasks can be found in an extended technical report.[7]

Data. For the evaluation, we used the (complete) DBpedia dataset, which covers a large amount of broad-ranging knowledge [6]. DBpedia allowed us to design evaluation tasks that are not targeted at a particular domain.

System. We made use of the *Information Workbench*[6], a system for interacting with the Web of data. The proposed faceted search approach was implemented using Oracle Berkeley DB Java Edition and Apache Lucene, based on the design and indexes reported in [4, 9]. We employed caching strategies to speed up cluster and rank computation and thereby guaranteed a fluent system interaction during the evaluation. Users were provided with a keyword search interface to obtain a starting point by typing in keywords. From the initial set of results, users continued via faceted search, i.e., via browsing, refinement and expansion operations. Results were visualised as introduced in [24], facets and facet values were presented as in [10], (extended) facet trees were represented as trees. Due to space reasons, we had to omit screenshots; however, they are included in our technical report[7]. The backend, including the keyword and faceted search modules, was implemented in Java 6 and the frontend is based on Ajax technologies running on a Jetty server. Experiments were carried out in a supervised manner on a PC with a T7300 Intel CPU and 4 GB memory, running on Microsoft Vista. We recorded the search process for each user and task using a screencast software.

5.1 Extended Facet Tree

Tasks. We prepared four tasks (C1-C4) for investigating the effects of our data value trees (i.e., data value clustering) on browsing. Task C3 is a precise need that involves a specific item of interest: *'Related to Berlin, find the Berlin Philharmonic orchestra'*. In contrast, the remaining 3 tasks involved fuzzy needs. In particular, tasks were fuzzy in the sense that the item of interest was specified imprecisely. Thus, participants had to browse and explore in order to fulfil these tasks. For instance, consider C2: *'Related to London, find all artists born some time in November 1972'*. Here, participants did not know what kind of artist or

[5] `http://trec.nist.gov/`
[6] `http://iwb.fluidops.net/`
[7] `http://www.aifb.kit.edu/web/Misc3004`

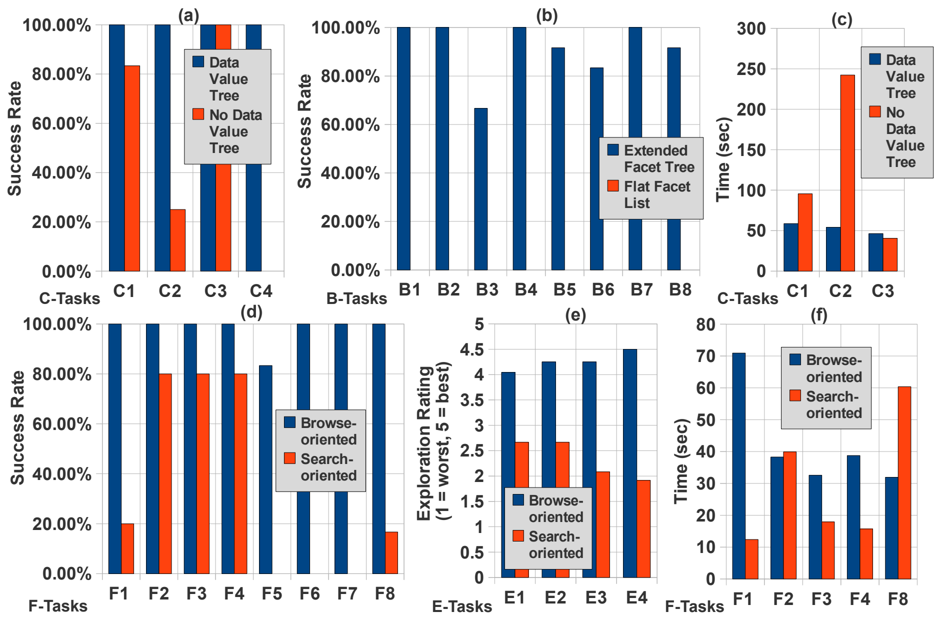

Fig. 3. Results of task-based evaluation

what concrete birthday to look for. The second class comprises eight complex browsing tasks (B1-B8). They have been designed to assess the quality of browsing based on the facet tree, compared to the baseline featuring a flat facet list. For accomplishing these tasks, users had to browse several facets to find a suitable facet path (length ≥ 2) for refinement (e.g., B1 *'Related to Paris, find all works having an actor, who is also a writer'*).

Baseline. In order to evaluate faceted search based on our extended facet tree, we used as baseline an implementation that represents the work in [23, 1, 19]. That is, browsing was solely based on a flat list of facets that are directly associated with the current result set. More precisely, we used two systems: (1) one system supported browsing on a flat list of facets without data value clustering and (2) our system that supported browsing based on the extended facet tree. Further, both systems employed a standard frequency-based ranking [8, 20, 16]. However, note, we designed clustering (C) and browsing (B) tasks in a way, that we were able to compare the effects of data value clustering en- or disabled and facets grouped in lists or trees. More precisely, in order to fulfil C tasks, users used made only use of a single facet and its associated data value tree or flat value list. Browsing tasks were designed analogously. Thus, we could observe the effects originating from facet trees versus lists and data value clustering versus no clustering.

Effectiveness. We observed that, if users were provided with a precise information need, the data value tree had no effect. Fig. 3a illustrates this result for task C3 – all participants could accomplish their assignment no matter the system. However, the fuzzier the information need, the more users depend on browsing the data value tree to solve their tasks (C1, C2, C4). More precisely, given a fuzzy need such as in C4 (*'Related to Hamburg, find all places having* *a name starting with [K-U]'*), we observed that without data value clustering, some users were not able or not willing to fulfil their assignment, because of the substantial effort needed. With clustering enabled, participants achieved a high success rate, as shown in Fig. 3a. This result suggests that extending the facet tree with data value trees is helpful for browsing, enabling users to handle fuzzy needs via exploring both facets and facet values using a hierarchical structure.

Further, we observed that no participant managed to complete their browsing tasks using the baseline (featuring a flat facet list) (Fig. 3b). A number of users did realise that these complex tasks can solved by executing several searches and browsing several facet lists. However, they also noticed the substantial time required and were not willing to complete their tasks (e.g., B3: *'Related to London, find all works, having as subsequent work a television show'*). Using our system (providing the facet tree for browsing), participants achieved a high overall success rate of above 67% (Fig. 3b). This result is promising as all tasks involve complex information needs that can only be satisfied using complex structured queries. Participants solved them by exploring facet paths (often path length ≥ 3) and combining them in an iterative fashion.

Efficiency. Without data value trees and given fuzzy needs, we observed that, if participants completed their tasks, they had to use a brute-force strategy to search through a large set of facet values. The brute-force approach led to more system interactions and notably higher costs, when compared to our system (with data value trees). Fig. 3c shows this effect for C1 (*'Related to Paris, find all places, having names starting with [I-K]'*) and C2. Regarding tasks that involve precise information needs, many participants used search (based on keyword queries) as a strategy to complete their tasks (e.g., C3). This resulted in a performance comparable to the use of a data value tree. In fact, we observed our approach to be slightly more expensive in case of C3 (Fig. 3d), as the browsing operations performed on the data value tree took more time than search.

Concerning the complex browsing tasks (B1-B8), we already pointed out that no participant succeeded, when using the baseline (Fig. 3b).

In conclusion, the experimental results suggest that the use of a hierarchical facet model (like our extended facet tree) improves the efficiency and effectiveness of the task completion, concerning complex, fuzzy tasks. Search is more efficient and equally effective, with regard to precise and simple needs only.

5.2 Browsing-oriented Ranking

Tasks. We prepared 12 tasks, which are divided into two classes: *find* (F) and *explore* (E) tasks. Class F consists of 8 tasks (F1-F8), which involve precise and fuzzy information needs (e.g., F8: *'Related to Seattle, find some international Airport'*). Class E (E1-E4) comprises 4 tasks, where users had to explore a result set (fuzzy need), i.e., find outliers, interesting or strange results (e.g., E4: *'Explore interesting entities related to Seattle'*). For E tasks, after a time threshold (5 minutes), users were asked a set of questions, in order to assess the users' understanding of the result set. Via these questions, users judged their understanding of the result set and rated the exploration experience and the knowledge that they could acquire on a scale from 1 (worst) to 5 (best).

Further, we divided the participants into two groups: For F tasks, the first group performed the tasks via search-oriented ranking, while the second group used browsing-oriented ranking. Thus, users solved each task only once. This is crucial, as the knowledge acquired from solving tasks using one system would impact on experiments with the other system. For E tasks, participants used both strategies, so that they could compare the exploration.

Baseline. Current ranking approaches are either generic w.r.t. information needs or assume precise needs. Our work is contrary to the latter approach and thus we compared our browsing-oriented ranking to such search-oriented approaches. More precisely, we used the metrics h and d (Section 4) and aggregated them to capture the intuition behind search-oriented ranking ($\overline{\mathcal{S}}$). Corresponding to the main idea (i.e., users having complete knowledge and therefore wish to minimise their search effort), $\overline{\mathcal{S}}(\hat{h}, \hat{d})$ aims at reducing the costs for fulfilling an information need, measured based on the number of refinement and expansion operations [8, 3, 9]. Thus, minimal tree height h is preferred to minimise the number of required facet operations. Further, facets should be discriminative, allowing users to quickly refine a result set. Thus, facets with high distinguishability score d are preferred. Overall, top-ranked facets aim at enabling users to perform rapid refinements and thereby reach an item of interest via few facet operations.

Effectiveness. For comparing the effectiveness of the two ranking strategies, we compared the average success rate for F tasks and the average browsing experience rating for E tasks. The results are depicted in Fig. 3d and Fig. 3e.

Concerning the success rate, given fuzzy needs, we observed that via search-oriented ranking, participants succeeded in tasks, where relatively small result sets (result size in the order of tens) had to be explored. Here, participants chose facets for browsing and refinement in a brute-force manner (e.g., F1 *'Related to Karlsruhe, find some city not located in Germany'*, or F3, F4 and F8). Concerning tasks with larger results (result size in the order of hundreds) and fuzzy needs, participants were not able to accomplish their assignments (e.g., F5 *'Related to Barcelona, find a strange educational institution'*, or F6 and F7).

Given precise needs, i.e., participants had precise background knowledge, users could solve their tasks equally effective with both rankings (e.g., for F2 *'Related to Karlsruhe, find a close-by airport'* some users knew that particular airport).

Browsing-oriented ranking outperforms the baseline on all tasks, especially those with large result sets and fuzzy needs. It seemed that for solving these tasks, users prefer general facets ranked high by the browsing-oriented strategy (e.g., *type* or *genre*), over fine-grained facets ranked high by the search-oriented strategy (e.g., *birthday* or *name*). As participants often had no precise knowledge regarding suitable values for such fine-grained facets (e.g., a specific birthday), they were not able to use them for exploration. Further, in many cases we observed participants using *type* for their initial exploration. *Type* helped them to get familiar with the current result set.

Concerning E tasks, the proposed ranking also performed well (Fig. 3e). Overall, exploration via browsing-oriented ranking was rated between 4 and 4.5, whereas exploration using search-oriented ranking was rated between 2 and 2.7.

Efficiency. For measuring user effort, we recorded the necessary time for all relevant system interactions, i.e., browsing, refinement and expansion operations. The time span for a browsing operation is defined as the time interval from the completion of the last facet operation, until the user browses to the next node in the facet tree (or decides to abort browsing). The time span for a refinement (expansion) operation is defined as the time from the last facet operation, until the user performs the next refinement (expansion).

On average, users needed 8 seconds for a refinement, 18 seconds for an expansion and 4.4 seconds for a browsing operation. The average time for each F task is illustrated in Fig. 3f. Note, F tasks that users did not complete for one of the systems are not shown (F5-F7).

Similar to the effectiveness study, we observed that the amount of results affects the performance of both strategies. When having a small result set (result size in the order of tens) and thus few facets, users solved their tasks on average with equal or less effort via search-oriented ranking (F1-F4). In particular for F2, users could exploit precise background knowledge (precise need), in order to refine the results set in an efficient, goal-directed manner. For the tasks F1, F3 and F4, users had few facets and thus were successful in guessing the appropriate facets that lead to their item of interest. Via browsing-oriented facets, on the other hand, more refinement and browsing operations were necessary, as high-ranked browsing-oriented facets restrict the result set in much smaller steps than search-oriented facets.

However, given fuzzy needs, when facing larger result sets (in the order of hundreds) and thus more facets, users were not able to guess suitable facets (F5-F8). More time had to be invested, as facet exploration was mere brute-force.

Thus, we can conclude that while browsing-oriented ranking might not provide the most efficient way to an item of interest, it is suitable for scenarios with no precise need and large result sets (thus, large facet and facet value spaces) to be explored.

6 Conclusion

Current faceted search approaches imply a precise information need and thus, focus on the search paradigm. We target the browsing paradigm, where users only vaguely know the domain or item of interest. To this end, we proposed the extended facet tree, which supports browsing based on a compact and hierarchical representation of the facet and facet value space. Based on the extended facet tree, we designed several metrics and incorporated them into a ranking scheme, which allows users to browse in small and uniform steps leading to observable and comprehensible result set modifications. We evaluated the proposed ranking and extended facet tree based on experiments with 24 tasks and 24 users. Our solution clearly outperformed the state-of-the-art on tasks, which involve fuzzy information needs and require dealing with large number of results. As future work, we plan to integrate search- with browsing-oriented solutions, allowing varying types of information needs. In particular, we will study how to switch between search- and browsing-oriented ranking. Further, we will address efficiency aspects of the proposed ranking and facet tree computation.

References

1. Simile: Longwell rdf browser. `http://simile.mit.edu/longwell/`.
2. B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. Parag, and S. Sudarshan. Banks: browsing and keyword searching in relational databases. In *VLDB*, pages 1083–1086, 2002.

3. S. Basu Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *CIKM*, pages 13–22. ACM, 2008.
4. O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. Shekita, B. Sznajder, and S. Yogev. Beyond basic faceted search. In *WSDM*, pages 33–44. ACM, 2008.
5. T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.
6. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.
7. T. Catarci, P. Dongilli, T. D. Mascio, E. Franconi, G. Santucci, and S. Tessaris. An ontology based visual tool for query formulation support. In *ECAI*, pages 308–312, 2004.
8. W. Dakka, P. G. Ipeirotis, and K. R. Wood. Automatic construction of multifaceted browsing interfaces. In *CIKM*, pages 768–775. ACM, 2005.
9. D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM*, pages 3–12. ACM, 2008.
10. M. Hearst, K. Swearingen, K. Li, and K.-P. Yee. Faceted metadata for image search and browsing. In *CHI*, pages 401–408. ACM, 2003.
11. P. Heim, T. Ertl, and J. Ziegler. Facet graphs: Complex semantic querying made easy. In *ESWC*, pages 288–302. Springer, 2010.
12. M. Hildebrand, J. van Ossenbruggen, and L. Hardman. /facet: A browser for heterogeneous semantic web repositories. In *ISWC*, 2006.
13. D. F. Huynh and D. R. Karger. Parallax and companion: Set-based browsing for the data web. In *WWW*, 2009.
14. E. Hyvönen, E. Mkel, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. Museumfinland – finnish museums on the semantic web. *Journal of Web Semantics*, 3(2):25, 2005.
15. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
16. J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *WWW*, pages 477–486. ACM, 2008.
17. G. Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.
18. G. Marchionini and B. Shneiderman. Finding facts vs. browsing knowledge in hypertext systems. *Computer*, 21(1):70–80, 1988.
19. E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for rdf data. In *ISWC*, pages 559–572, 2006.
20. E. Oren, R. Delbru, K. Möller, M. Völkel, and S. Handschuh. Annotation and navigation in semantic wikis. In *Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics*, 2006.
21. S. R. Ranganathan. *Colon Classification.* Madras Library Association, 1933.
22. A. Russell and P. Smart. Nitelight: A graphical editor for sparql queries. In *ISWC*, 2008.
23. Schraefel, M. Wilson, A. Russell, and D. A. Smith. mspace: improving information access to multimedia domains with multimodal exploratory search. *Commun. ACM*, 49(4):47–49, 2006.
24. T. Tran, P. Haase, and R. Studer. Semantic search — using graph-structured semantic models for supporting the search process. In *ICCS*, pages 48–65, 2009.