tFacet: Hierarchical Faceted Exploration of Semantic Data Using Well-Known Interaction Concepts

Sören Brunk[1] and Philipp Heim[2]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
soren.brunk@deri.org
[2] Institute for Visualization and Interactive Systems (VIS),
University of Stuttgart, Germany
philipp.heim@vis.uni-stuttgart.de

Abstract. Information stored in the Semantic Web is becoming more and more interesting for average Web users. Due to the complexity of existing tools, however, accessing it is difficult. In this paper we therefore introduce tFacet, a tool that uses well-known interaction concepts to enable hierarchical faceted exploration of semantic data for non-experts. The aim is to facilitate the formulation of semantically unambiguous queries to allow a faster and more precise access to information in the Semantic Web for the broader public.

Keywords: Faceted exploration, faceted search, Semantic Web, known interaction concepts, hierarchical facets.

1 Introduction

The amount of information available as semantic data is growing rapidly. As of June 2011, the *Linking Open Data (LOD)* cloud [1], for instance, contained more than 200 different datasets. The most popular dataset within the LOD cloud is the *DBpedia* dataset [2]. It contains structured information that is extracted from Wikipedia articles and converted into a semantic representation. It is publicly accessible via the *SPARQL* [3] query language, allowing for semantically unambiguous access. In comparison to the often ambiguous text search, it allows the formulation of more complicated requests accurately and thus supports targeted and quick access to information.

A disadvantage of using SPARQL is, however, that users have to learn the language first in order to access information through SPARQL queries; making it rather a language for experts. In order to meet the needs of all users, a simpler user interface is required to create semantically unambiguous and complex queries without the need to learn the complex syntax of a query language. One approach to solve this problem is based on the concept of *faceted exploration* [4]. In faceted exploration, the user always sees all remaining search options within the search process and can select them step-by-step in order to refine the query.

Several applications exist that implement the concept of faceted exploration in order to allow users to access semantic data. Examples are *mSpace* [5], *Parallax* [6], *Faceted Wikipedia Search* [7] or *gFacet* [8]. Most existing applications, however, don't use the full potential of faceted exploration in combination with semantic data and thus lack the ability to create powerful queries. Others allow the creation of more complex queries but are often difficult to use. mSpace and Faceted Wikipedia Search, for example, are easy to use, but they don't support the creation of hierarchical facets; that is, facets that are connected to the results through indirect attributes. In contrast, Parallax and gFacet do support hierarchical facets. However, the creation of those can be difficult for inexperienced users due to the use of new and unfamiliar interaction concepts within these tools.

For that reason we introduce tFacet, a tool that uses well-known interaction concepts in order to make the power of faceted exploration available also for inexperienced users.

2 tFacet

tFacet, like the other tools, uses the concept of faceted exploration to access semantic data. However, it thereby applies interaction concepts that are well-known from other applications and thus allows the widespread use of already existing knowledge in the formulation of semantically unique queries. It is implemented using the *Adobe Flex Framework* [9] and therefore runs in any Web Browser with the Flash plugin. Using SPARQL as the query language allows for faceted exploration of any RDF dataset accessible via a SPARQL endpoint. By default tFacet uses the DBpedia dataset but can be easily configured to access other datasets as well.

2.1 Initial Search Space Limitation

Each exploration within tFacet starts with an initial limitation of the search space. This step is necessary to reduce the number of possible results as well as the number of facets to a displayable amount. The user selects a base result class from a tree representation of all classes contained in the dataset. The better the user knows what he is looking for, the more precisely he can restrict the search space. For example in the DBpedia Ontology he could select "Eurovision song contest entry" as a very specific base class limiting the search space to 1054 objects. On the other hand, if he is not able to specify his search goal in detail in advance, he could choose a more abstract base class such as "film"; containing 53619 objects (see Figure 1).

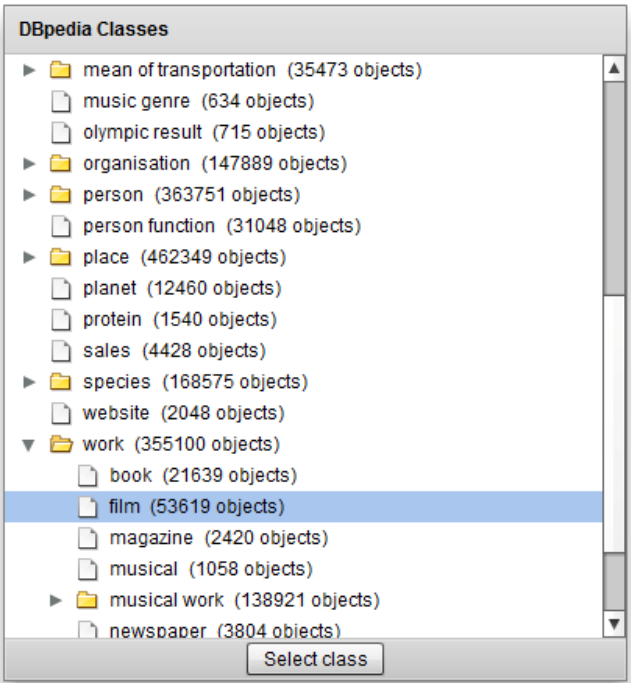

Fig. 1. Selection of base class "film" from the DBpedia ontology.

2.2 Hierarchical Faceted Exploration

By pre-selecting a base class, the exploration is limited to objects of that class. Thus in Figure 2 only objects of the class "film" are displayed in the result set (A). On the left side a directory tree representation of facets is shown. It displays all facets that can be used to filter objects in the result set. At first, only the top hierarchy of the tree is visible, containing all facets related to direct attributes of the result set. In case of films, for instance, it contains facets such as directors or actors of the corresponding movies. In addition, some elements of the tree can be expanded in order to show lower hierarchy levels with additional facets. Those facets refer to indirect attributes of the result set. For movies these could be, for example, the birthplace of the director of the movie.

The user now can select individual facets from the tree to show the facet's detail view in the right area (Figure 2, C). Facets in the detail view are arranged vertically and always contain all remaining search options as selectable attributes to explore or filter the result set. By selecting individual attribute values, the result set is reduced to objects having that value. For instance, by selecting a director, the result set is filtered to show only movies from that director. Attributes in different facets are combined using the AND-operator while attributes within one facet are combined using the OR-operator. In this manner, the result set can be refined iteratively until the information wanted has been found.

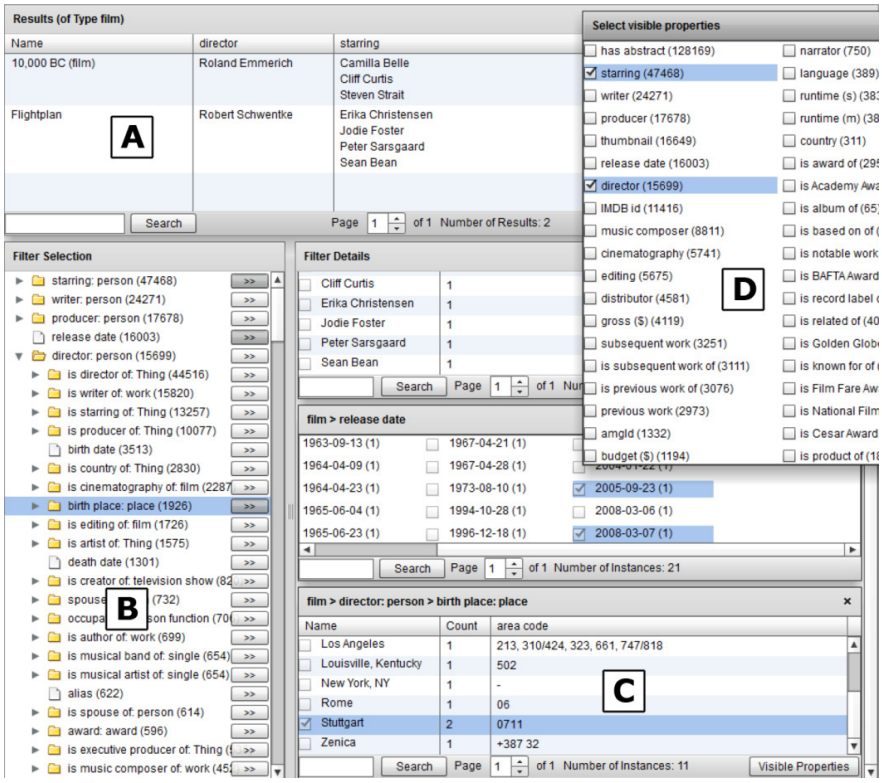

Fig. 2. The main view of tFacet is divided into three parts: Result set (A), facet tree (B) and facet details (C). Columns can be added interactively (D).

In addition to the result set, a filter operation also updates all other facets that are shown in the detail view. All attributes that would lead to an empty result set when selected are hidden. Furthermore, the expected number of results when selecting a certain attribute is shown in brackets besides this attribute. This feature helps to avoid dead-ends and allows a user to estimate the outcome of a specific filter operation.

All facets available in the directory tree can be extracted automatically from semantic data by using SPARQL queries. All properties of the selected base class are collected and displayed in the tree. If a property leads to a literal this is represented by a document symbol in the tree and cannot be further explored (see “release date” in Figure 2, B). If a property leads to objects of another ontological class it is represented by a directory symbol and can be further explored (see “director: Person” in Figure 2, B). The name of such a sub-directory is composed of two components: (1) the type of the property (here “director”) and (2) the class of the objects (here “Person”). Representing object properties as sub-trees allows also deeply nested information to be used for faceted exploration. In this way, the birthplaces of the directors of the movies in the result set can be used as hierarchical facet to show, for example, only movies that were directed by directors born in Stuttgart (Figure 2, C).

2.3 Additional Columns

Until now we have used the linked structure of the Semantic Web only for faceted exploration. It is also possible to use this structure to enhance the presentation of the result set by additional details. To demonstrate this, tFacet implements a functionality that allows users to add additional columns to both the result set as well as the facets and also to use those columns for sorting.

In order to add an additional column, the user can click the button "Visible Properties" to see all properties and choose the ones that should be visible in the list (Fig. 2, D). For example, in this way it is possible to use information about the actors of a movie ("Starring") both to filter and sort the result set. Furthermore, having the properties available as additional columns allows the relations between information to be directly visible for the user (e.g. the relations between movies and actors) in contrast to the separated representation via facets.

But often there is more than one value for a property (e.g. usually more than one actor plays in a movie). In that case, the current implementation of tFacet uses a simple list to display multiple values. This can cause problems however, if many values exist for a property, for example, if a movie has many actors. To avoid this problem an abbreviated list could be shown initially or just the number of entries, with the possibility to expand the view if necessary. With numerical data it would also be possible to show an aggregated view.

In general, the idea of additional columns is not only limited to direct properties, but can be implemented also for indirectly related properties. For that, one could imagine displaying a directory tree like in Fig. 2 D to select information only connected indirectly. However, one problem with such an implementation would be how to maintain the clarity of the presentation as many trees and their hierarchical directory structures could confuse the user more than help him create search requests.

2.4 Well-known interaction concepts

tFacet uses several interaction concepts that are known from common applications in order to ease the use of faceted exploration. The most important ones are:

- **Directory tree**: In many applications, directory trees are used for the navigation in and the management of hierarchical data. A logical conclusion was therefore the representation of hierarchical facets in a directory tree within tFacet. Like in popular file managers, nodes shown as folder symbol can be explored further while nodes shown as file symbol represent a leaf node.
- **Subdivision of the user interface into three parts**: Also, a partitioning of the user interface into an overview (directory tree), a detailed view (right pane) and a result view is frequently used. The possibility to show multiple detail views (in this case facets) at the same time is not widespread, for the definition of filters in more than one facet, however, necessary.
- **Organization into columns:** In many grid-based applications (for example, in Windows or Mac OS applications) the user can determine individually, which columns are of interest for him and have these displayed. In a similar way in tFacet

he can display a menu of all properties he might be interested in and add or remove individual columns. The same is true for sorting by individual columns by clicking on the column header.

3 Summary and Future Work

In this paper, we proposed tFacet, a new application to enable all users to create semantically unambiguous search requests using the concept of faceted exploration. The aim was to enable the full potential of this concept by keeping its usage as simple as possible through the use of well-known interaction concepts. Through the use of a directory tree even remote information can be included in the query as hierarchical facets and the possibility to show connected properties in additional columns facilitates control over the level of detail of the information displayed.

Thus, tFacet offers interesting approaches to make the potential of semantic data accessible for everyone. A key part of this is the strategy to transfer already known concepts into the specific interaction environment of the Semantic Web.

In the future we plan further development of tFacet to make its usage even simpler. For example, data-specific visualizations, such as sliders or maps could be used to ease the creation of queries and improve the presentation of results. An integration of the approaches discussed in Section 2.3 to extend the columns functionality to use hierarchical properties could be useful for a consistent and powerful implementation of this concept. Furthermore, an evaluation of the user interface would be helpful, especially to see if the use of known interaction concepts can help users to reach their goal faster. Including a comparison to similar tasks and tools in order to measure empirically, whether the intended goal of tFacet, a simplified usage, has been achieved.

References

1. LOD cloud (2010): http://richard.cyganiak.de/2007/10/lod/lod-datasets_2010-09-22.html
2. Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: Proc. of the 6th ISWC 2008, 2008, pp. 722-735.
3. SPARQL (2008): http://www.w3.org/TR/rdf-sparql-query/
4. Yee, K.-P.; Swearingen, K.; Li, K.; Hearst, M.: Faceted metadata for image search and browsing. In: Proc. of the CHI 2003, ACM Press, 2003; pp. 401-408.
5. Schraefel, m.c.; Smith, D.; Owens, A.; Russell, A.; Harris, C.; Wilson, M.: The evolving mSpace platform: leveraging the semantic web on the trail of the memex. In: Proc. of Hypertext 2005, ACM Press, 2005; pp. 174-183.
6. Huynh, D. and Karger, D.: Parallax and companion: Set-based browsing for the Data Web (2009). http://davidhuynh.net/media/papers/2009/www2009-parallax.pdf
7. Hahn, R.; Bizer, C.; Sahnwaldt, C.; Herta, C.; Robinson, S.; Bürgle, M.; Düwiger, H.; Scheel, U.: Faceted Wikipedia Search. In: Proc. of BIS 2010, pp. 1-11.
8. Heim, P.; Ertl, T.; Ziegler, J.: Facet Graphs: Complex Semantic Querying Made Easy. In: Proc. of the 7th Extended Semantic Web Conference (ESWC2010), Part I, Springer, 2010, pp. 288-302.
9. Adobe Flex (2011): http://www.adobe.com/de/products/flex/