Humboldt: Exploring Linked Data

Georgi Kobilarov
Hewlett-Packard Labs
Bristol, UK
georgi.kobilarov@gmx.de

Ian Dickinson
Hewlett-Packard Labs
Bristol, UK
ian.dickinson@hp.com

ABSTRACT

We present Humboldt, a novel user interface for browsing RDF data. Current user interfaces for browsing RDF data are reviewed. We argue that browsing tasks require both a facet browser's ability to select and process groups of resources at a time and a 'resource at a time' browser's ability to navigate anywhere in a dataset. We describe Humboldt which combines these two features in a single coherent interface. Our approach is based on the operation of *pivoting*, which enables the user to move the focus of a browsing from one set of resources to a set of related resources. With repeated use of the pivot operation the user can browse anywhere in the data. We describe a preliminary evaluation of our approach and discuss its implications for further development.

Categories and Subject Descriptors

H5.2 [**Information Interfaces and Presentation**]: User Interfaces; H5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia

General Terms

Design, Human Factors

Keywords

user interface, semantic web, faceted browsing, rdf, linked data

1. INTRODUCTION

As more and more large Linked Data sources such as DBpedia [5] have become available over the last year, the need for improved user interaction approaches for dealing with these highly interlinked RDF [21] datasets has become more obvious. The quantity of data has reached a level where users are overwhelmed and feel lost in a similar way as in earlier days of the web. That phenomenon was described as 'Lost in Hyperspace' [14].

Current research in the semantic web user interface area, especially in semantic web powered search engines, has tended to focus on *information retrieval* tasks. We characterize these tasks as those in which a user has a very specific question or query in mind which he tries to answer or solve. Companies like Powerset [4] focus on providing mechanisms to answer tasks such as e.g. 'what is the population of Berlin?', 'which books are written by author X?', etc. using semantic technologies.

In this paper, we focus on *exploratory tasks* and characterize them as follows: in exploratory tasks, a user has a more vague idea of the question he wants to answer. His interests depend upon the changing characteristics of the information context [7] [8].
We give two business related examples for this kind of tasks:

- A product manager plans a product update. He wants to get an overview of the new features of competitors' products. User reports of his product have been tracked by his companies support department. His development team came up with new ideas for innovative product features. The production department has published technical data on new methods of cost reduction factoring. The product manager needs to decide which changes will be made with the next product release.
- A financial analyst notices an irregularity in a dataset. He seeks for an explanation. The company works in a broad market and it might be a competitor's news as well as a companies news. The company is active in foreign markets and depends on certain suppliers. Political news as well as changes in the suppliers market might also have influenced the financial data.

Both examples highlight the difficulty of having a computer system 'solve' these problems by providing a single answer. These tasks require gathering together structured and unstructured data sources. In this paper, we focus on exploratory tasks using only structured data.

In the current document-centric web, a key mechanism of web browsers to support information exploration is the browsing history. Users can follow different links to other docu-

ments and, depending on the document's relevance, decide to further explore or go back and follow a different path. The possibility of returning to a previous decision and following a different path makes decisions less risky.
Browsing histories and bookmarking functionality help users to store discovered information in the system instead of having to memorize it for later reuse.

The Semantic Web with its access to more structured information makes data aggregation techniques more important in exploratory tasks. Structured data can more easily be aggregated, supporting better overview of large amounts of data.
In information retrieval tasks on web documents, a goal is to find a specific document or - in terms of graph-theory - finding a particular node in the graph of linked documents. While the path to that node or the edges connecting it are important for localizing the node, the node's meaning is mostly independent of which path was used to retrieve it.

That is different in Semantic Web tasks. The path used to locate a node or the relationship between nodes is more important. The meaning of just retrieving the node of a person named 'Ian' is different than retrieving it as a link from a person named 'Georgi' by following a `foaf:knows` link. The later represents 'Ian as a friend of Georgi'. With this example we want to highlight how and why exploratory tasks are different and why the user's exploration process and path is more meaningful that in the web of documents.

A user will want a semantic web interface to follow a similar flexible approach as browsing backwards and forwards in exploratory tasks. An interface supporting that approach would support user behaviour and exploration strategies, in which the user has a perceived low risk of making early decisions because he can go back and modify them later.

We outline a simple example for casual end-users: a user might want to explore a dataset of films in order to find an interesting one for a movie night. He does not know upfront which films are available and has no specific preference of the genre. Over the past, he has seen many movies and do have some preferences for particular directors and actors, but is willing to explore new ones.

The essence of this task is viewing the actual available data, which will influence his exploration strategy and might change his interests as described above.

2. OVERVIEW OF EXISTING APPROACHES

In this section, we review three different kinds of semantic web user interfaces according to their ability to support users in information exploration tasks. We identify the key principles of each approach, and will then further discuss how the advantages can be combined into one user interface design. We are interested in generic user interfaces rather than interfaces tailored to a specific domain. While there are certain visual representations for particular data types, e.g. calendars and timelines for time-related data and maps for geographical data, there are no such representations for most other types of complex resources, e.g. people, films, companies etc. Hence we treat those specific representations as additional features to a more generic user interaction approach and do not include them in our analysis.

With growing availability of Linked Data [9] on the Web, we argue that there is a need for generic approaches which do not restrict users to specific tasks or to specific data that could be used, as both might not be known a priori.

2.1 Browsing

There are various different semantic web browsers for Linked Data available, which use a similar interface design to Tim Berners-Lee's Tabulator [10]. A browser interface is typically designed to display one RDF resource at a time, and enables the user to browse from resource to resource. Tabulator's original design uses a tree to show the relationship between resources. Other Linked Data browsers including Disco [11] or the Zitgist DataViewer [15] use a design of one resource per page. The tree-view as well as the one-resource-per-page design enables the user to browse detailed descriptions of single RDF resources, while the tree-view additionally preserves the browsing path.

This design reflects a common user interaction on the web of documents. There, the smallest piece of information is a web page, and a web browser enables users to navigate from one page to another. This browsing approach has been applied to semantic web linked data, where the smallest piece of information is a RDF resource. We argue that this kind of interaction does not facilitate the key benefits of Semantic Web data. Much of the currently available Linked Data is either scraped from web pages, such as DBpedia, or also available as human readable webpage as well like Geonames [1] or Musicbrainz [3]. We do not see much benefit to users from viewing RDF data rendered by a Linked Data browser when a HTML representation of the same information is available, which is specifically tailored to fit the need of human readers. Linked data browser will need to provide higher benefit by giving more control to the user than web pages could do.

However, this might change as more and more RDF data becomes available from databases, where no corresponding web pages are available. Semantic Web technologies could serve as methods to create web documents, but the key in browsing and analysing this data lies in our opinion in aggregation of data, like the currently available Web2.0 mashups.

Single resource at a time RDF browsers enable users to explore large RDF datasets, but their design does not support the user in aggregation tasks. The browser design might enable aggregation tasks when a kind of basket is available to temporally store discovered resources, such as Piggybank [18]. Piggybank focuses on collecting resources, while we try to find ways of exploring these collections. Also, in their current design, single resource at a time browsers give no suggestions of which links to follow because of their lack of support in creating an overview for the user.

For example, given a user's FOAF [13] profile, browsers do not give hints for the user which links to friends might be interesting, when users naturally want to deal with collections of resources. A user strategy (see section 1) for exploring such a list of friends with a Linked Data browser is to browse to each friend, memorize details that might be of interest and repeat that procedure for other friends. This strategy is very time consuming and impractical for larger lists of friends. Here, faceted browsing or faceted filtering of the friendlist might serve as a helpful overview method.

2.2 Faceted Filtering

The method of faceted filtering [26] or clustering helps the user to get an overview of a given set of resources. The basic concept of facets is to partition the information space using orthogonal conceptual dimensions of the data. A widely used customer application with a faceted filtering approach is iTunes, Apple's music library and media player [20].
iTunes' music library can be filtered according to different dimensions of the music files' metadata. Users can filter the library on certain artists, albums or genres by selecting values from facets.

Exhibit [19] is a popular interface which provides a faceted filtering view on graph based data. Exhibit's user interface presents different facets for a given set of resources. The key user interaction is faceted filtering, i.e. a list of resources can be filtered by selecting values within facets. E.g. films might be filtered by selecting 'Steven Spielberg' in a 'Director' facet of a list of films. This action will filter down the list of films to those, whose director is Steven Spielberg. Exhibit uses a declarative language to create a faceted view on a list of resources. Other faceted browser like Longwell [2] or MSpace [23] calculate facets automatically.

Faceted Filtering supports exploration tasks as it enables the user to quickly get an overview of what data is present. Compared to the previously described design of one resource at a time, faceted browsers use a multi resource at a time design. Even larger sets of resources can easily be clustered or filtered by certain dimensions in order to quickly see what data is available. For example, a large list of people might be clustered by their birthplace or by the organization they work for. The user can then filter the set of people down to a smaller set according to these facets.

Current implementations work with a fixed list of resources on which faceted filtering is performed. With growing datasets, the number of potential facet values also grows. A set of thousands of films might have a facet with hundreds of different actors. Hierarchical facets are one solution to further cluster facet values.

2.3 Query Builder

Graph based query builders try to give the user the most expressivity in querying an RDF graph. An common approach is to provide a triple based user interface which helps to build triple pattern to express queries. One example of such an interface is the DBpedia Query Builder [6]. It supports the user in building up triple pattern by using auto-completion for triple predicates. For example if the users starts with the triple pattern `?film rdf:type film` , auto-completion for next triple pattern will only show predicates which are used with actual instances in the repository of type film, ordered by their usage number. This supports the user in exploration tasks as he sees which relationships exist between instances. But a user has to have an explicit question in mind which he wants to formulate into a query. While the interface helps to analyse the relationships, it does not show actual data while constructing a query. In addition, users do not always have explicit queries upfront, but need to explore the available data first in order to find out what information might be interesting to them.

Using actually available data for query construction is a concept known in database environments as *Query by Example* (QbE) [27]. Users can create queries by entering example data and conditions, and the system then translates these examples into a more formal query according to the actual data schema. The Microsoft Access query builder uses an interface which refers to the QbE concept in order to enable casual users to construct database queries.

3. OVERVIEW OF OUR APPROACH

We present an interface design in which we combined the three previously discussed interaction approaches: Browsing, Faceted Filtering and Query Building. We integrated the following three characteristics in our exploratory interface: multi-resource at a time, browsing and implicit query building. We argue that the multi-resource interface design is most helpful for exploration tasks as it supports overview techniques and data aggregation as shown in faceted browsers. Therefore we also integrate a faceted filtering mechanism similar to Exhibit. Our interface approach does not relate specificly to RDF. It is applicable to any graph-based data structure. For our experiments and demonstration we use a highly interlinked dataset extracted from DBpedia (see figure 1). This dataset contains data related to the movie domain and consists of films, directors, actors, cities, and companies. We modified the dataset in order to provide a rich linking structure with n:n relations between instances, as well as circular relationships. Films have many actors, actors play in different films of different directors. Directors get awards for films and work for companies which produce films.

4. INTERFACE DESIGN

We now describe the main elements of our interface (see figure 2). The interface is presented as a fixed workspace of three different elements: result list, facet list and history.

4.1 Result List

The core of our interface is the result list. The result list presents a collection of RDF resources as unstructured list

```
prefix : <http://dbpedia.org/resource/>
prefix p: <http://dbpedia.org/property/>

:Catch_Me_if_You_Can p:has_director :Steven_Spielberg .
:Catch_Me_if_You_Can p:starring :Tom_Hanks .
:Catch_Me_if_You_Can p:starring :Leonardo_DiCaprio .

:The_Departed p:has_director :Martin_Scorsese .
:The_Departed p:starring :Jack_Nicholson .
:The_Departed p:starring :Leonardo_DiCaprio .

:Martin_Scorsese p:birthPlace :Queens .
:Steven_Spielberg p:birthPlace :Cincinnati .
:Leonardo_DiCaprio p:birthPlace :Los_Angeles .

:The_Departed p:distributor :DreamWorks .

:DreamWorks p:keyPeople :Steven_Spielberg .
```

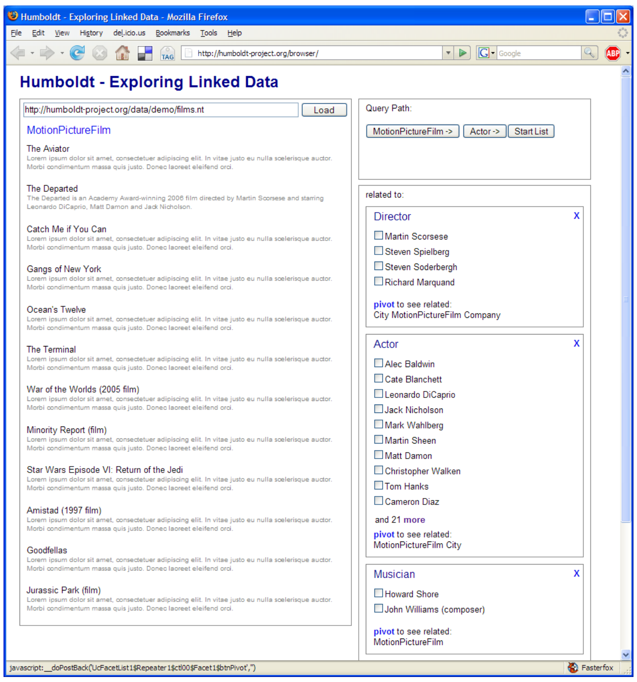
Figure 1: dataset sample

of items without exposing their relations. The result list is used in two ways: for displaying an external RDF document and for displaying query results from within the browser. When an external document is loaded, all resources in this document get rendered into a plain list independently of their relationship. A tag cloud on top of the list can be used to filter the resources by their `rdf:type`. This can be used to render the output of other applications, e.g. semantic search interfaces. These applications can provide the URI of a RDF document which contains a list of resources to be displayed and browsed with Humboldt.

4.2 Facets

The main point of interaction in our interface design is a list of facets, presented on the right hand side of the result list. Facets are computed from all resources that are related to resources in the result list and grouped by their `rdf:type`. This method differs from other faceted browsers, where facets are usually constructed from a predicate that connects two resources. By using `rdf:type` instead we want to test the hypothesis that introducing an additional level of abstraction reduces the number of facets. This new level of abstraction should help the user to get an overview over the available data more quickly. During this exploration phase, the user does not need to see all details of existing relations.

For example, a list of people (results) in a movie related dataset can be connected to a list films (facets) with many different predicates like `directed`, `stars in`, `has produced`, `has written`, `has edited`, and `was awarded for`. If the dataset also contains cities that are related to the people, facets like `born in`, `lives in`, `died in`, etc would appear. By grouping by type, the only facets are film and city. The user can get an overview of the available data and later drill down by splitting one facet into several ones based on particular predicates or sub-classes. He thereby restricts the relationships between results and values in a certain facet.


Humboldt - Exploring Linked Data - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
http://humboldt-project.org/browser/
Google
Load
MotionPictureFilm -> Actor -> Start List
X
Fasterfox

Humboldt - Exploring Linked Data

http://humboldt-project.org/data/demo/films.nt

MotionPictureFilm

The Aviator
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

The Departed
The Departed is an Academy Award-winning 2006 film directed by Martin Scorsese and starring Leonardo DiCaprio, Matt Damon and Jack Nicholson.

Catch Me if You Can
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Gangs of New York
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Ocean's Twelve
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

The Terminal
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

War of the Worlds (2005 film)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Minority Report (film)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Star Wars Episode VI: Return of the Jedi
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Amistad (1997 film)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Goodfellas
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Jurassic Park (film)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In vitae justo eu nulla scelerisque auctor. Morbi condimentum massa quis justo. Donec laoreet eleifend orci.

Query Path:

related to:

Director
☐ Martin Scorsese
☐ Steven Spielberg
☐ Steven Soderbergh
☐ Richard Marquand

pivot to see related:
City MotionPictureFilm Company

Actor
☐ Alec Baldwin
☐ Cate Blanchett
☐ Leonardo DiCaprio
☐ Jack Nicholson
☐ Mark Wahlberg
☐ Martin Sheen
☐ Matt Damon
☐ Christopher Walken
☐ Tom Hanks
☐ Cameron Diaz

and 21 more

pivot to see related:
MotionPictureFilm City

Musician
☐ Howard Shore
☐ John Williams (composer)

pivot to see related:
MotionPictureFilm

javascript:__doPostBack('UcFacetList1$Repeater1$ctl00$Facet1$btnPivot','')

Figure 2: Humboldt Browser

A list of films could be related to a facet of people. Users want to be able to restrict these relations to certain properties, e.g. to people who directed or produced a film and exclude people who acted in a film. This operation could be understood as a drill down into the result-facet relationship.

As in Exhibit, we designed facets for filtering the result list. The user can select a value in a facet to filter down the result list to all resources that are related to the selected resource. This will cause all other faceted to be recomputed based on the now filtered list of results.

For example, having a list of films loaded into the result list, the user can select Steven Spielberg from the Director facet. This will filter the list of films to those, which are related to Steven Spielberg by any predicate. The user could later drill down into the Director facet to select only directed by and awarded for as relations between these films and Steven Spielberg.

4.3 Pivot Operation

In order to combine a faceted layout with browsing capability, we use the *Pivot* operation, which is similar to Siderean Seamark [24]. Based on the operations in data drilling [16], our approach allows multiple representations of data according to different dimensions. In our faceted interface design, a facet or dimension of the dataset is treated as a first class object. Pivoting on a facet means to handle the values of this facet as results and rebuild a faceted view around these results (see figure 3). In this way, we added a browsing functionality by which the whole dataset can be explored. This interaction has been called *refocusing* in a *nested faceted browsing* prototype by David Huynh [17].

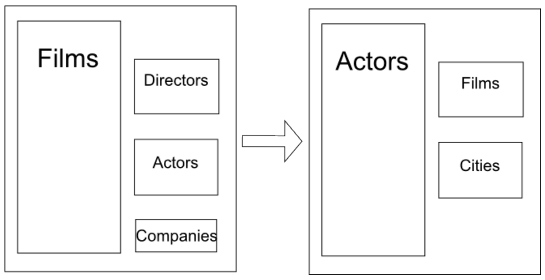


Figure 3: Pivoting example

Furthermore, we apply selected filters to the whole dataset and all pivoting steps. When a resource list is filtered by certain facet values and the user then pivots on another facet, the interface uses the result of the filtering for building the next view. This enables the user to build up a query while browsing actual data and instantly seeing the result of each incremental step of building up his query. Thereby, we support the implicit construction of queries.

E.g. a user who filters films on certain directors and then pivots on actors will see all actors in the result list, who are related to any film in the previous result list.

4.4 History

In order to support the user in building an implicit query over a number of connected list (of facets), we suggest tracking the user's path through the graph. We use a separate History control, which displays the path the user has taken via pivoting facets from the starting point to the current page. We chose to display the path as a linear sequence in the history control, which enables to step back to previously visited pages. While this represents the implicitly built query, filters that are applied on a graph node affect other nodes as well.

When a user pivots from films to actors, filters these actors by their birthplace, and then steps back to films, the filtered actors themselves affects the list of films presented. This enables drilling-down into facets. As described previously, facets are partitions of data according to different dimensions (see section 2.2). Actors are one dimension of film data, but actors themselves have different other dimension, e.g. their places of birth. Starting with film, users can drill into their actors and their places of birth, and afterwards zoom out to films again using the history control.

5. IMPLEMENTATION

We have implemented a prototype of our interface as server-side web application in C#. The core of the implementation is a command queue which tracks all user interactions. This implementation design follows the Command Pattern [25]. This programming pattern is often used to implement undo/redo functionality in graphical user interfaces. We used this data structure to track user interactions in order to represent the history internally in a flexible and extendable way. Each command queue is associated with one data source, a per user-session in-memory triple store.

A controller provides methods to compute all result list items and all facets on a given command queue and data source at any given time. The command queue thereby represents the implicit construction of a query via different pivoting steps without having to store intermediate result sets. On every event causing a page refresh, the result list is computed by parsing the whole command queue and executing SPARQL queries on the data source, which is populated via HTTP GETs on RDF documents.

In the current prototype we do not have a mechanism in place to dynamically fetch further RDF data by deferencing URIs. This could be archived by using an approach similar to the Semantic Web Client Library [12]. Such a mechanism will be included in a future version of the prototype. The primary objective of the current phase of work was to evaluate the effectiveness of the user interaction model.

In our prototype there are four different commands which can be enqueued: SelectCommand, FilterCommand, PivotCommand, and StepCommand. The SelectCommand includes or excludes resources from the resultlist or filters the resultlist by type. The FilterCommand restricts results by values in a certain facet. The PivotCommand tracks the user's pivot operations and the StepCommand tracks steps in the history.

Through the following chapters, we will use the following example set of interactions:

- filter a list of film by director Steven Spielberg
- pivot on facet 'Actors'
- pivot on facet 'Films'
- filter on director Martin Scorsese

A prototype of our user interface and an example dataset is available at `http://humboldt-project.org`.

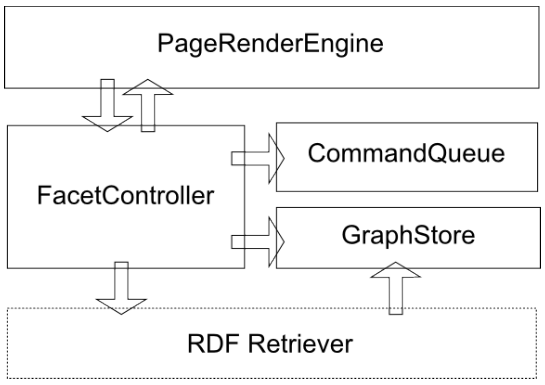


Figure 4: Humboldt's architecture

6. USER FEEDBACK

In this section we describe a preliminary evaluation our user interface approach. We used the previously mentioned film-related dataset extracted from DBpedia for an evaluation of our interface prototype. A small group of semantic web researchers were given the task to explore the dataset by setting different filters and pivoting between facets. They could filter films by particular directors or actor, pivot on a list of actors to filter them by birthplace or by other films they stared in, etc.

We will discuss the issues our evaluation group had with our prototype. We elected not to use a formal quantitative evaluation process, as our interface design is in an early stage of development and we were interesting in getting qualitative feedback from users while using the interface instead of quantitatively measuring results.

6.1 Filtering

While using Humboldt, users easily understood the faceted filtering operation. They were able to predict and understand the result of selecting resources in facets and appreciated the newly introduced level of abstraction of type-based facets. However, they missed the feature of narrowing a type based facet to certain predicated describing the relationship between e.g. a list of films and a list of directors. This feature (see section 4.2) was not implemented in the prototype shown during our user study.

As described in section 4.2, we introduced a higher level of abstraction using facets that are based on the type of resources that are directly related to current resources in the result list.

6.2 Pivot

When using the pivoting operation for the first time, our users expressed some confusion. The pivoting operation causes every control on the screen to update. The result list is populated with new items and new facets appear. Even if the user was familiar with the concept of pivoting, they still showed that reaction. We showed a different prototype to each user, in which we used an animation in order to make the pivoting operation easier to understand. In this animation, when clicking on 'pivot' the items in the resultlist and all facets except the pivoted one slowly disappear, the pivoted facet then 'flies' into the result list and finally the new facets appear. The users appreciated this animation and noted that it makes the pivot operation itself much easier to understand.

We tried to test users' understanding of multiple pivoting operations by asking them to express their expectations. We tested that by using the following sequence of actions:

1. filter films by director Steven Spielberg
2. pivot on actors
3. pivot on films

Every user was able to articulate a correct description of what every single operation would cause. They correctly described operation 3 as such, that it will show a list of all films in which one or more of the currently shown actors has starred in. However, some users showed surprise when actually executing operation 3 and viewing the result. Some users remembered that operation 1 filtered the list of films to Steven Spielberg related ones, and were thereby surprised to see more values in the director facet. Other users had forgotten about operation 1, but showed the same surprise when they were reminded by us about that filtering operation. Even with an understanding of the interface operation and the structure of the dataset, only a few users were able to correctly predict the outcome of the described sequence of operations.

We remind readers that the dataset contains a n:n relation between films and actors, and therefore the result set of the sequence of all three operations is a superset of the result set of operation 1 (above).

6.3 History

Our users quickly understood that each pivoting operation adds a field to the history control shown in the top right corner of the screen. We asked them to note that the current representation of the history as a one-dimension sequence is just a test and that this layout can not represent all pivot operation. Although we could have chosen to represent the history as a graph in order to give it the more expressivity, we decided to just test how users would react to the possibility of navigating previous pivoting steps.

The users' understanding upfront was mixed. It was not clear to them whether using the history would take them to the same resultset as it was at that time, or whether the later added filtering operation would be incorporated. It was unclear if the history preserves the state or navigates one large query.

6.4 Review of user feedback

In this section we discuss our interface design and the results of our user feedback. We were satisfied overall with the introduction of our pivoting operation. This approach has shown a way to enable users to more easily explore an interlinked RDF dataset at a more aggregated level compared to single-resource at-a-time browsing. Using animations helped to understand the pivoting operation and we will incorporate these animations into our main prototype.

The history and its navigation functionality complemented the pivoting operation successfully. However we came to the conclusion that the way the history is currently presented is not optimal. Our initial concern was that having a linear history could be a major disadvantage and changing its layout to a graph based design could serve the users needs much better. We are now considering that having a separate history control in the user interface might not work at all.

Our users did not only demand a history functionality which represents the pivoting path, but also presents previously performed filtering operations. In combination with our concern about the linear layout of the history, we conclude that the history's function needs to be better incorporated into the overall interface design instead of being a separate and for the user somehow disconnected user control.

Our current interface design is layout as a fixed workspace (result list facets), where data is put into (supported by our animations). For the next experiments we will try a design of a moving workspace on top of a two-dimensional dataset layout. While previous discussions such as Schraefel and Karger [22] have highlighted issues of graph visualisation tools, we argue that using a faceted graph visualisation layout instead of an instance based layout could produce better results.

7. EVALUATION

We now discuss how well our user interaction approach serves the in section 1 described needs of users performing exploratory tasks on interlinked RDF data. As outlined in section 1, exploratory tasks are performed with greater user satisfaction if the following characteristics are present:

- the interface design is consistent with the user's behaviour and expectations
- the user does not need to memorize details of his exploration, because the application supports him by tracking his interactions
- the risk of following different paths is low

Our user study showed that with minor tweaks, the overall behaviour of our interface is predictable for the user. We argue that some of the issues described in the previous section, e.g. the issue with n:n relations of films and people, might be omitted with some user training. These issues arose out of the richness of our underlying dataset, which will be common in many RDF datasets and users might get a better understanding over time.

By implicitly tracking the users interaction and providing a mechanism to review previous navigation steps, we also satisfy the second need. Even if users were able to memorize all their previous steps, they appreciate if they are not expected to do so. We argue that with larger datasets users were not even able to memorize their actions, and our history mechanism is thereby helpful as well as necessary.
Most importantly, we successfully lowered the user's perceived risk of bad decisions by disconnecting the building of a query path and fine-tuning the filtering operations. We want to highlight this argument with an example. Given the previously used sequence of interaction

- filter a list of films by one particular director (Steven Spielberg)
- pivot to all actors of these films
- pivot to their films

We additionally assume that the user might have set additional filters on actors and on their films.
If the user now discovers that his first decision of filtering only on Steven Spielberg films was incorrect, he could easily go back and change that filter (maybe by adding another director) while retaining all other operations.
Hence, he can use a strategy of exploring a dataset and building up a query graph by navigating the dataset first and afterwards fine-tune the result set by adding, changing or removing filters. Enabling this interaction strategy in the user interface for graph-based data is the key contribution of this paper.

We conclude that this strategy is especially helpful and useful for exploratory tasks and thereby our interface design successfully addressed our outlined hypothesis.

8. CONCLUSION

In this paper, we presented a new approach for browsing Linked Data that combines the benefits of being able to explore throughout a dataset offered by single resource browsers such as Tabulator with the ability to manage groups of resources offered by faceted browsers such as Exhibit or Longwell. We have reviewed current most common user interface designs for RDF data and identified their key principles and benefits in order to developed a user interaction design based on these principles which is tailored to data exploration strategies.

9. FUTURE WORK

In the future we would like to try optimizing the workspace layout in order integrate the history functionality in a more consistent way. We want to evaluate that layout in a more quantitatively measured usability study. We will also add the previously described facet splitting functionality and integrate the Semantic Web Client Library for dereferencing URIs at runtime.

10. REFERENCES

[1] Geonames. http://www.geonames.org/.
[2] Longwell. http://simile.mit.edu/wiki/Longwell.
[3] Musicbrainz. http://musicbrainz.org/.
[4] Powerset. http://www.powerset.com.
[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data.
[6] S. Auer and J. Lehmann. What have innsbruck and leipzig in common? extracting semantics from wiki content. In *ESWC*, pages 503–517, 2007.
[7] M. Q. W. Baldonado and T. Winograd. Sensemaker: an information-exploration interface supporting the contextual evolution of a user's interests. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 11–18, New York, NY, USA, 1997. ACM.
[8] M. J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review, v13 n5 p407*, 1989.
[9] T. Berners-Lee. Linked data, 2006. http://www.w3.org/DesignIssues/LinkedData.html.

[10] 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.*, 2006.

[11] C. Bizer and T. Gauss. Disco - hyperdata browser, 2007. http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/.

[12] C. Bizer, T. Gauss, and R. Cyganiak. Semantic web client library. http://www4.wiwiss.fu-berlin.de/bizer/ng4j/semwebclient/.

[13] D. Brickley and L. Miller. Foaf: the Šfriend of a friendŠ vocabulary, 2004. http://xmlns.com/foaf/0.1/.

[14] D. M. Edwards and L. Hardman. Lost in hyperspace: cognitive mapping and navigation in a hypertext environment. In *Hypertext: theory into practice*, pages 90–105, Exeter, UK, UK, 1999. Intellect Books.

[15] F. Giasson. Zitgist dataviewer. http://dataviewer.zitgist.com/.

[16] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *IEEE 12th International Conference on Data Engineering*, pages 152–159, 1996.

[17] D. Huynh. Nested faceted browsing. http://people.csail.mit.edu/dfhuynh/projects/nfb/.

[18] D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. *Web Semant.*, 5(1):16–27, 2007.

[19] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 737–746, New York, NY, USA, 2007. ACM.

[20] A. Inc. itunes. http://www.apple.com/itunes/.

[21] F. Manola and E. Miller. Rdf primer, Februar 2004.

[22] m.c. schraefel and D. Karger. The pathetic fallacy of rdf. In *International Workshop on the Semantic Web and User Interaction (SWUI) 2006*, ["lib/utils:month_12911" not defined] 2006.

[23] m.c. 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] Siderean. Seamark navigator, 2007. http://www.siderean.com.

[25] Wikipedia. Command pattern. http://en.wikipedia.org/wiki/Command_Pattern.

[26] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM.

[27] M. M. Zloof. Query-by-example: a data base language. *IBM Systems Journal*, 1977.