Generating Exploratory Search Interfaces for the Semantic Web

Michal Tvarožek, Mária Bieliková

Generating Exploratory Search Interfaces for the Semantic Web

Michal Tvarožek[1], Mária Bieliková[1]

[1] Institute of Informatics and Software Engineering,
Faculty of Informatics and Information technologies,
Slovak University of Technology,
Ilkovičova 3, 842 16 Bratislava, Slovakia
{ tvarozek,bielik }@fiit.stuba.sk

Abstract. At present, the promise of the Semantic Web has yet to be realized, partly because there are few real-world applications that allow end-users to access, view and process Semantic Web information. We aim to facilitate Semantic Web adoption by providing users with advanced exploratory search capability over Semantic Web data by providing a faceted exploratory search interface for arbitrary Semantic Web repositories. Our approach takes advantage of metadata describing the structure of the respective information spaces to construct facets that can be used to visually construct semantic queries. Subsequently, we generate result overviews to display individual search results and lastly generate an incremental graph-based view for individual resource exploration. We performed proof of concept validation of our interface generation approach and present the lessons learned by a small scale feedback gathering user study.

Keywords: exploratory search, facet generation, faceted browsing, personalization, Semantic Web, user interface generation, graph exploration.

1 Introduction

Although the Web has become an almost ubiquitous virtual information space providing information, services and communication tools, there are still (large) parts of the Web that are not generally accessible to end-users. The Deep Web which accounted for most of the data on the Web in 2005 [5] consists of many databases which can be accessed over the Web by end-users through querying interfaces. Typical search engines cannot index the Deep Web as its contents are hidden in databases and exist only temporarily in the form of web pages when a query is made.

The Semantic Web aims to provide better search and browsing capabilities by enabling machine readability of information on the Web taking advantage of ontologies and metadata [9]. Most Semantic Web content is part of the Deep Web and stored in publicly accessible semantic repositories (e.g., accessible via SPARQL endpoints), in the form of distributed Linked data or as metadata associated with legacy web documents. Despite continuous progress in semantic search engines such

as Sindice.com, the original promise of the Semantic Web still remains unrealized [9]. The main challenges for Semantic Web adoption lie in its:

- *Visualization* – Semantic Web contains raw information without any associated presentation templates thus offering no default way to render it in human readable form making end-user grade visualization difficult. Furthermore, resources can be associated with legacy web content (web pages, images, videos, etc.) or have many attributes and relations to other resources causing information overload.
- *Querying* – semantic queries resemble relational database queries rather than typical keyword queries used in web search engines making them impractical for most users. Manual construction of semantic queries (e.g., in SPARQL) is a complex task, which in addition to query language proficiency requires prior knowledge of the respective information space.
- *Exploration* – the Semantic Web is essentially a graph of resources and their attributes and relations, and also associated legacy web documents (e.g., web pages or multimedia). Exploratory search principles [7] stress open ended tasks, learning and understanding of information in context, not just finding a specific resource e.g., with a traditional search engine. Here orientation support, multiple navigation and/or visualization options and the ability to move towards a goal from different directions are needed to provide satisfactory user experience.

Our aim is to *facilitate Semantic Web adoption*, which is now seriously hindered by the lack of end-user grade search and exploration tools, by *providing an exploratory browser for the Semantic Web*. To achieve this goal we also have to address already existing problems associated with the present Web, such as information overload, the navigation problem or web dynamics (i.e., constant, uncontrolled changes).

In our previous work, we devised a faceted semantic exploratory browser taking advantage of Adaptive Web [2] and Social Web [11] approaches to provide personalized visual query construction support and address guidance and information overload [13]. In this paper, we *extend our browser with user interface generation* using metadata describing the presented information spaces to provide users with a smooth user experience accounting for dynamic changes in the information space.

We present an overview of related work in section 2, and *describe the novel generation of the respective parts of the browser's exploratory search interface* – result overviews, graph view and facets in sections 3 and 4 respectively. Next, in section 5 we present proof of concept validation of our approach in the domain of digital images, the feedback gathered in a small scale user study and the lessons learned so far. Lastly, we conclude and summarize our contribution in section 6.

2 Related Work

Our work has a strong multidisciplinary background ranging from Information retrieval to Adaptive web-based systems and HCI with focus on faceted browsers and facet generation, exploratory search, information visualization and the Semantic Web.

Wilson and schraefel performed a study comparing three prominent exploratory browsers – Flamenco, mSpace and RelationBrowser++ [15]. While Flamenco and RelationBrowser++ are more traditional faceted browsers, mSpace takes advantage of

RDF data (native to Semantic Web) to provide users with a set of customizable filters that can be used to visualize a subspace of a high dimensional information space. The RelationBrowser++ is tailored to exploration of large statistical data and persistently displays all facets at the top unlike Flamenco, which hides exhausted facets [16].

Unlike all previous browsers, the faceted browser Factic stresses personalization as a vital feature enabled by user action tracking and evaluation [13]. In order to better understand user behavior in faceted browsers, Kules et al. performed a user study examining how searchers interact with individual parts of a faceted browser. The study discovered that users primarily explore the result list and the facets, while mostly ignoring the current query. Kules also argued that the design of exploratory search tasks as well as methodologies for evaluation of exploratory browsers were still in an early stage of development making thorough evaluation difficult [6].

The BrowseRDF faceted browser provides elementary facet generation capability over simple RDF data [8]. BrowseRDF automatically identifies facets in source data based on several statistical measures, but offers only very limited interaction options and does not consider semantic metadata provided in the more expressive RDFS and OWL formats. Other approaches include automatic multifaceted hierarchy generation from textual collections [3], and middleware solutions posing as proxies between databases and users providing a faceted interface by dynamically suggesting a number of facets using precomputed decision trees [1].

Neither of these approaches can be effectively used for complex interactive exploration of Semantic Web content, which in addition to faceted querying needs to support interactive information visualization and exploration of graphs (the Semantic Web being a graph). The VisGets interface allows users to perform interactive/exploratory web search by querying it in three dimensions – time, location and topic, while also providing advanced visualization of search results [4]. However, VisGets uses its own crawling and indexing engine and thus cannot be effectively used for general web search or Semantic Web exploration. Here, also graph visualization and interaction approaches must be considered as described in [10].

3 Exploratory Search Interface Generation

We previously devised a personalized faceted browser for semantically enriched information spaces [13], which used personalization to improve overall user experience. In order to facilitate exploratory search experience, we extended the base browser with support for multi-paradigm querying and exploration including keyword-based and content-based search (query-by-example), adaptive result overviews and incremental graph-based resource exploration [12]. This however was still not enough due to the dynamic nature of the information spaces (e.g., users constantly adding or modifying information) where the ability to automatically adapt to changes, e.g. by (automatically) generating user interfaces, is crucial.

Thus, our focus lies with the generation of exploratory search interfaces for the Semantic Web environment, although this can be somewhat generalized towards the Deep Web and even legacy Web environment.

In order to support exploratory search and achieve these goals, we need to support three parts of user experience:

- *Query construction*, which includes the initial construction of an exploratory search query, its modification and execution; to support multi-paradigm search and exploration based on our previous work, we need to support keyword-based, view-based (faceted) and content-based (query-by-example) query construction.
- *Result browsing*, which includes the rendering of suitable result overviews, selection of result ordering and the displayed result attributes, and support for effective selection of individual results for further exploration.
- *Resource exploration*, which includes the detailed presentation of individual resources, their attributes and relationships with other existing resources.

We address these issues by generating a set of user interfaces each supporting the individual stages of the exploratory search process. We generate:

- *Faceted browser interfaces* for advanced query construction and modification.
- *Result overviews* for effective presentation of selected result attributes.
- *Graph-based exploration views* for incremental horizontal exploration of semantic resources and their relations with other resources.

3.1 Information Space Representation

We work with semantically enriched information spaces, e.g. an ontological repository, where both metadata describing the structure of the information space and data are represented by ontologies (e.g., in RDFS or OWL). Thus our approach assumes a description of classes, individuals, relations and attributes describing a particular domain. For example, in the digital image domain, *di:Author* and *di:Photo* are classes; *di:Author_1* and *di:Photo_1* are individuals, while *di:createdBy* is a relation between *di:Photo_1* and *di:Author_1*. Similarly, *di:viewedCount* equaling *10* is an attribute of *di:Photo_1*, also defining the domain of the attribute as the class *di:Photo* and its range as an *xsd:int* (see Fig. 1).

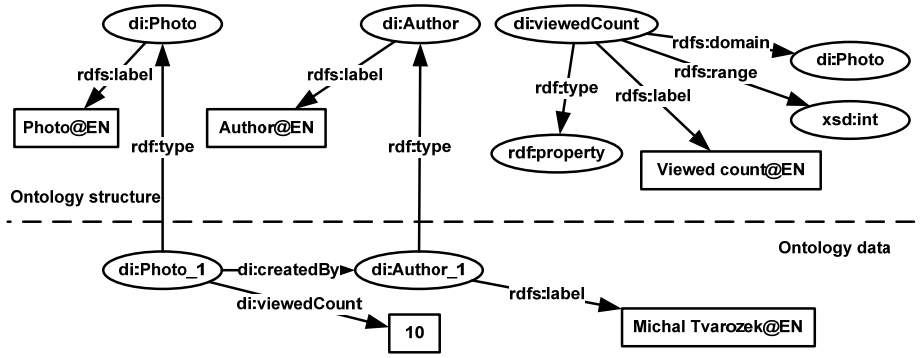

Fig. 1. Example of a simple domain ontology for the digital image domain. Metadata describing the domain model are shown at the top; individuals representing data are shown below. Round nodes denote complex resources with URIs, rectangular nodes denote literals.

As shown in the above example, a domain ontology as defined by W3C contains a detailed standardized description of classes, properties (relations and attributes) and the used data types, effectively defining a data model. Ontologies can also be populated with individuals, which conform to the specified domain model and materialize it in instances of classes and properties. Note that the ontology is in fact an oriented graph where nodes represent individual resources.

3.2 Result Overview Generation

We generate two result overviews – the *ListView* shows thumbnails and properties of individual results (see Fig. 2), while the *MatrixView* shows thumbnails, provides additional information in tooltips, and offers a generated editing pane for (batch) modification of individual result attributes (see Fig. 3).

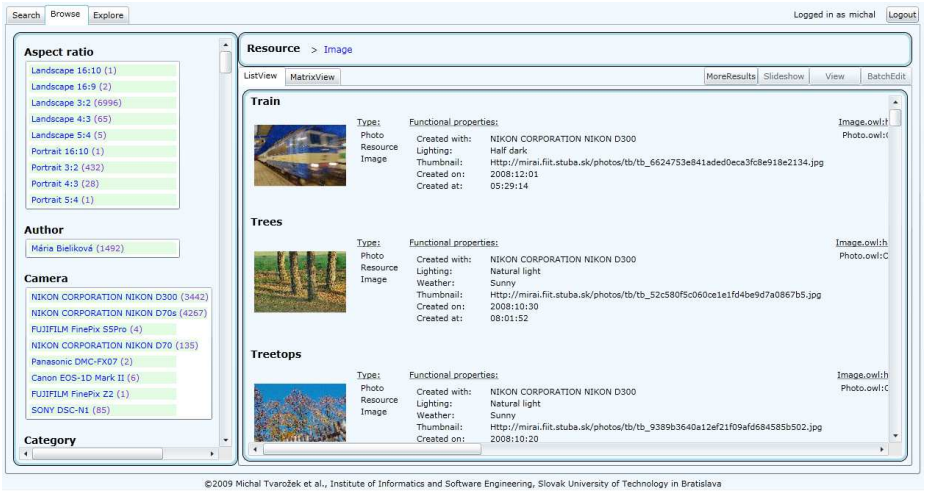

Fig. 2. Example of generated facets shown in the browser GUI (left) along with a list-based result overview showing all result properties (right). The Author facet corresponds to a direct object facet, while the Aspect ratio and Camera facets are indirect object facets associated via an EXIF helper object with the original photo.

In the above example, *ListView* shows properties of a specific result directly derived from the domain ontology visualized as *label-value* pairs. For multi-value properties such as *Type* in Fig. 2, a column with all values is shown. We either show all result properties to maximize information value or apply personalization to select only the most relevant properties (detailed personalization description was described in [13]).
In *MatrixView*, the generation principles remain the same except for the editing pane, which is generated separately. For each specific result type, we identify all applicable properties from the domain ontology metadata, construct editing widgets based on property types (e.g., text boxes with language selection or auto-complete combo

boxes with single/multi-value support). Properties with existing values are shown first, while properties without values are shown at the bottom (see Fig. 3).

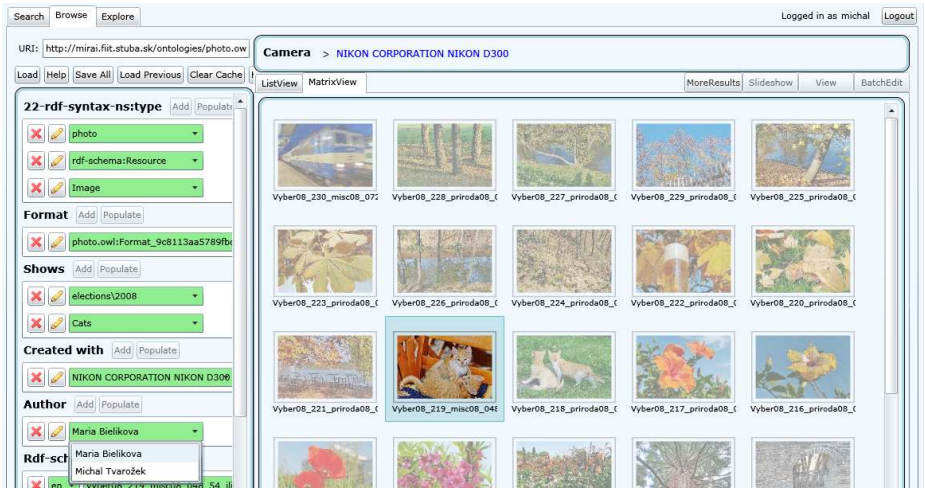

Fig. 3. Example of a generated matrix result overview showing image thumbnails (right), and the correspondingly generated annotation pane for collaborative content creation (left).

The generated graph exploration view consists of the graph visualization window, predicate filtering windows and an options toolbar (see Fig. 4). Users can access the view either directly by typing in the URI of the node they wish to explore or by exploring a result found in the faceted browser (see Fig. 3).

The graph itself is generated directly from the domain ontology showing only individuals (objects) and their relations. Dark nodes correspond to individual resources, white nodes correspond to relations between them; arrows denote relation directions, node attributes (i.e., values of literal properties) are normally hidden and only shown as tooltips after hovering over a node.

Relations are intentionally visualized as separate nodes connecting resources to reduce information overload as one relation can have multiple values and to improve graph layout. E.g. in Fig. 4, the relation *weather* shown on the right would otherwise have to be displayed on all edges making the graph less readable.

Each exploration sessions starts by showing the first dark node (individual) and its direct neighbors, which corresponds to a window or a view of the graph. The user can next move the visible window by selecting another central node, or incrementally expand the view by expanding one or more of the visible nodes. The view in Fig. 4 was initiated by showing the node *Trees* (left) and expanding the node *Sunny*.

To make the graph understandable, we employ a force-based layout algorithm, but also allow the user to fix and manually reposition nodes in the resulting graph. Apart from traditional view panning, users can use two zoom options – regular zoom enlarges or shrinks the view, advanced zoom spatially expands dense node clusters to make them less crowed. Lastly, as ontologies often contain much data, we also enable users to hide irrelevant nodes manually thus reducing information overload.

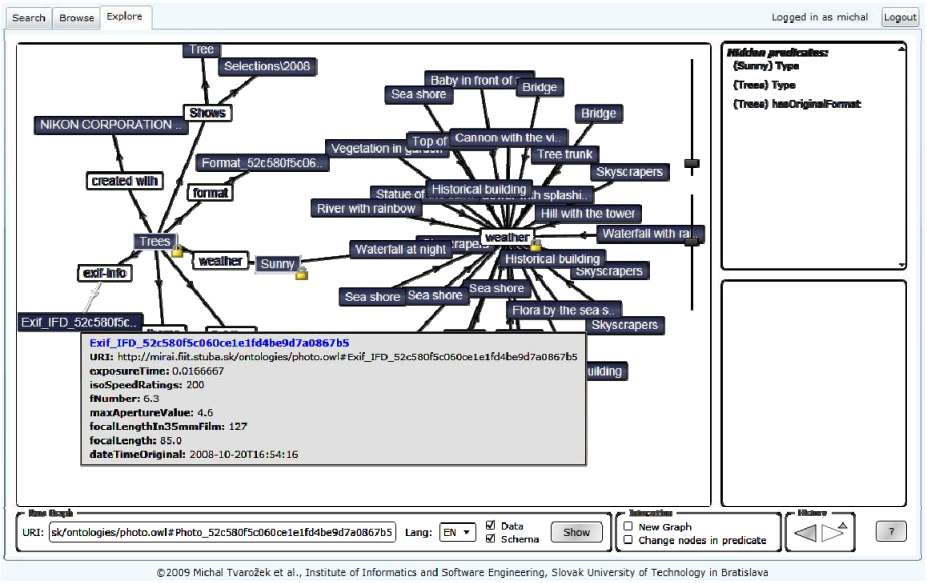

Fig. 4. Example of our generated graph-view exploration interface. Dark nodes represent individual resources, white nodes correspond to relations (top). Hovering over nodes shows the attributes of a node (center); additional tools include zooming, node hiding and history (right), with additional filtering options for languages and data/schema only visualization (bottom). Note that the color scheme has been modified for printing purposes.

4 Facet Generation

During facet generation, we examine metadata describing the information space, identify patterns corresponding to facets, construct facet restrictions based on the identified metadata and map the resulting facet onto the graphical user interface and the semantic back-end, which provides querying services. As such, facet generation must define these facet properties:

- A *facet template*, which corresponds to a pattern found in domain metadata and specifies the overall type and behavior of the facet.
- A *restriction template*, which defines how the individual restrictions in the facet are constructed and mapped onto the domain ontology.
- A *query template*, which defines how the back-end query engine creates database queries and maps them onto facet restrictions.
- A *visualization and interaction template* (i.e., the corresponding widget type), which binds the facet to the graphical user interface and handles user input.

The purpose of the facet generation process is to identify specific predefined patterns in the metadata and map them onto a set of predefined templates in three successive steps: facet identification, construction and mapping as described below.

4.1 Facet Identification

During the facet identification stage, we identify the facet template, restriction template and query template. We first search for eligible candidate properties by examining properties of individual instance types and their transitively associated properties, which can be used for automated facet construction from the domain ontology based on low-level metadata facet templates. We distinguish

- *object facet templates* that correspond to properties having complex object values (e.g., a class such as *di:Author*), and
- *literal facet templates* that correspond to properties having simple values (e.g., numbers, dates or strings).

In the above example (see Fig. 1), the *di:createdBy* is a suitable candidate property matching the *class-property-class* pattern between *di:Photo* and *di:Author*, resulting in a object facet template.

Similarly, we define several *restriction templates*:

- *enumeration*, which corresponds a flat list of restrictions (e.g., days of the week),
- *hierarchical taxonomy*, which corresponds to a hierarchical tree of values connected via a transitive property in the domain model (e.g., a hierarchy of geographical locations such as country-state-city-street).

We distinguish the following *query templates* based on the instance-property relation:

- *Direct query template*, which corresponds to the direct property pattern: *{instance} property {value}*
- *Indirect query template*, which corresponds to the indirect property pattern: *{instance}* $property_1$ *{} ... {}* $property_N$ *{value}*

Our facet identification algorithm tries to match these predefined templates and their variations onto the domain ontology metadata, evaluates possible matches and forwards successful matches to the successive facet construction stage.

4.2 Facet Construction

After a facet has been identified, its internal representation must be constructed before it can be used in the browser. The facet construction stage applies the templates identified in the previous stage, constructs facet restrictions based on the restriction template, and persistently stores facet metadata for future use.

The crucial step of facet construction is the initialization of facet restrictions using the restriction template and the definition of the interaction mode. Normally a facet can work as a list of restrictions from which users can select one or more values, or as a search box where users can search for and select a specific restriction. We determine the interaction mode based on the overall number of potential restrictions; *list mode* is used for a small number of predefined values (e.g., days of the week), *search mode* is used for large numbers of values (e.g., all cities on Earth). If an ordering of values is defined in the ontology for object values, we can also create restriction intervals to cover continuous values (e.g., real numbers or dates).

4.3 Facet Mapping

The last facet mapping stage selects a suitable user interface widget to render the generated facet in the faceted browser, and maps the constructed facet and restriction values onto the widget. The widget provides facet visualization (see Fig. 2) and handles user interaction forwarding events and facet metadata to the back-end search services, which use the *query template* and the user selection in the facet to construct SPARQL queries in order to retrieve results corresponding to the generated facet.

Although a broad range of potential interface widgets could be developed, such as lists, histograms, maps, timelines, etc., their detailed description is beyond the scope of this paper as automated discovery of what specific visualization/interaction to use would likely prove difficult. Thus we only employ list widgets at this time and leave the use of more advanced widget types as one possible direction of future work.

5 Validation

We reworked our original personalized faceted browser prototype [14] with the interface generation principles described above as a client-side Silverlight application working inside a web browser to minimize deployment effort. This allowed us to move user specific functionality onto the client and also provide interactive features not supported by HTML (e.g., the interactive graph view). We performed several experiments to validate individual parts of our approach in the digital image domain.

Our goal was to validate two primary aspects of our solution – our facet and result overview generation approach, and the novel graph-view Semantic Web exploration approach. Consequently, we performed two groups of experiments:

- A proof of concept experiment with the facet generation approach, where the goal was to verify that our approach generates meaningful and usable facets for our personalized faceted browser. Note that the goal was not to generate the best possible set of facets, but rather a good enough set to use for personalization.
- A user study with our graph exploration approach, the goal being to gather user feedback on the generated GUI and its usefulness for Semantic Web exploration.

Data. Our domain ontology of images is based on the popular Kanzaki EXIF ontology (http://www.kanzaki.com/ns/exif) and contains about 8 000 manually and semi-automatically annotated images. The entire ontology consists of 35 classes, 50 properties (including relations and attributes), more than 32 000 individuals and in excess of 150 000 facts. For individual photos, the ontology describes EXIF metadata as supplied by the camera, information about formats in which the photos are available (e.g., resolution, aspect ratios), and optional additional annotations such as the author, the object and background of the photo, the place, overall theme and expression, lighting conditions, weather and the event to which the photo belongs.

Methodology. In the proof of concept experiment, we generated facets from the available data and examined how the original browser behaved in practice and whether the interface was still usable for its intended purpose in terms of usability and

performance. We performed several experiments with and without personalization, and also after some changes in the information space have been made.

In the user study with our graph exploration interface, we made our browser available to a target group of 10 end-users aged between 20 and 25 years with an IT background. As none of the users had previous knowledge of Semantic Web principles nor had used similar graph-based tools before, each user was given a brief introduction about the functionality of the browser. Next, the users were asked to complete a set of 5 tasks using the browser which also counted the time and number of clicks made (e.g., finding a specific image, discovering image properties or getting a better understanding of the domain). Lastly, each user was asked to fill out a questionnaire with the results of the tasks and his experience with the browser.

Results and Lessons Learned. The experiments with facet generation proved the approach was viable for interface generation with minimal performance impact. We successfully managed to distinguish facet and restriction templates, direct query templates, and construct and map facets to interface widgets and use them in our exploration interface without any significant negative impact over manually created facets due to facet generation.

Based on our experiments, we want to point out these lessons learned:

- Identification of direct query templates resulted in many facets being generated, which we expected to handle at the personalization stage later in the browser. However, this had negative impact on performance and we had to employ selection metrics (e.g., based on significance) already during the facet identification stage.
- The identification of indirect query templates was limited due to the complexity of selecting viable options. Consequently, either the identification algorithm must be further refined or a workaround via indirect nested facets (i.e., facets in facets) needs to be used complicating facet generation and mapping.
- During facet generation and result overview generation, blank nodes and helper objects in the domain ontology caused problems as, e.g., empty, meaningless or unnamed interface items were generated and had to be accounted for.
- Some generated facets such as location eventually had too many restrictions (e.g., hundreds) making them unusable and significantly decreasing performance. This required the change of the interaction mode from *list mode* to *search mode*, where users could type in their desired restriction instead of selecting from a list of hundreds of items. This problem could also be alleviated by prior hierarchical structuring of the information space before facet generation.
- The users preferred *alphabetical restriction ordering* in facets; other orderings such as relevance based or potency based had negative impact on user experience as users were unable to seek in the restrictions which were in an unexpected order.
- Using type/information specific facet widgets instead of list widgets would likely improve usability in specific cases, such as date selection via calendars, location selection via maps or timeline selection via histograms as was done in [4], but effectively generating mappings for advanced widgets would be more complex.

The user study with the graph exploration interface showed that 9 out of 10 users managed to find the specified image, although the time required varied widely – 141 seconds and 8 clicks were required on average, although the fastest user needed less

than 50 seconds while the slowest one required almost 5 minutes. Overall, the users managed to answer 75% of the questions correctly leaving 25% false answers (this also includes answers that were close to the correct ones, but not exactly right).

Based on these results, we conclude that graph-based exploration is viable for Semantic Web browsing as most users were able to accomplish the given tasks despite having no prior experience with a similar interface. Still, improvements to layouting and node selection are necessary to improve understandability and task times, which was also confirmed by user feedback which indicates that non-expanded graphs are easy to understand (rating 4.5 on a 5 level Likert scale), while expanded graphs are less readable (rating 3.4). Further feedback indicates that although response times were generally acceptable, some operations took too long to complete (e.g., loading the new graph after expanding a node took sometimes too long).

6 Conclusions

We described our novel approach to exploratory user interface generation for the Semantic Web with specific focus on facet generation, result overview generation and graph view generation. Based on our experiments in the digital image domain, we argue that the approach works and is viable for its intended purpose. Consequently, we see our primary contribution in:

- Devising an *all-around method for (semi)automated generation of Semantic Web exploratory search interfaces* addressing interfaces for query formulation, result overview browsing and individual result exploration.
- *Enabling Adaptive Social Semantic Web exploration* and thus facilitating the adoption of the Semantic Web by end-users.

Despite the positive feedback we got, we encountered scalability issues with exploratory search approaches over remote repositories, as network delays increase the response time of both interface generation (if performed online) and of actual end-user exploration of the respective information spaces.

Although the described method was primarily intended for Semantic Web repositories and possibly Linked data exploration, most of the described principles could be extended to a Deep Web relational database, provided that metadata describing the structure of the information space were available.

Even more interesting is the extension of the proposed approach for legacy web content for specific pages (e.g., personal browsing history) or for whole web sites (e.g., generating a faceted browsing interface for a typical corporate web site). This could be accomplished by taking advantage of contextual/navigational links between pages and entity extraction approaches *ultimately providing a seamless search and browsing experience for both legacy web and semantic web content*.

Acknowledgments. This work was partially supported by the grants VEGA 1/0508/09, KEGA 028-025STU-4/2010 and it is the partial result of the Research & Development Operational Programme for the project Support of Center of Excellence for Smart Technologies, Systems and Services, ITMS 26240120029, co-funded by the ERDF.

References

1. Basu Roy, S., Wang, H., Nambiar, U., Das, G., and Mohania, M.: DynaCet: Building Dynamic Faceted Search Systems over Databases. In: Proceeding of International Conference on Data Engineering, pp. 1463-1466, IEEE CS, (2009).
2. Brusilovsky, P., Kobsa, A., Nejdl,W., eds.: The Adaptive Web: Methods and Strategies of Web Personalization. LNCS, vol. 4321. Springer, Berlin (2007).
3. Dakka, W., Ipeirotis, P. G., Wood, K. R.: Automatic Construction of Multifaceted Browsing Interfaces. In: Proceedings of the 14th ACM international conference on Information and knowledge management. pp. 768-775, ACM Press, New York, USA, (2005).
4. Dörk, M., Carpendale, S., Collins, C., Williamson, C.: VisGets: Coordinated visualizations for web-based information exploration and discovery. IEEE Transactions on Visualization and Computer Graphics, 14(6) 1205–1212, (2008).
5. Gulli, A., Signorini, A.: The indexable web is more than 11.5 billion pages. Special interest tracks and posters of the 14th Int. Conf. on WWW. pp. 902-903, ACM, New York, (2005).
6. Kules, B., Capra, R., Banta, M., and Sierra, T.: What do exploratory searchers look at in a faceted search interface?. In Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries. JCDL '09. ACM, New York, pp. 313-322, 2009.
7. Marchionini, G.: Exploratory search: from finding to understanding. Communications of the ACM, 49 (4), 41-46, (2006).
8. Oren, E., Delbru, R., Decker, S.: Extending Faceted Navigation for RDF Data. In Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P. et al. (eds.) ISWC 2006: Proc. of the 5th Int. Sem. Web Conf.. LNCS, vol. 4273, pp. 559-572. Springer, Heidelberg (2006).
9. Shadbolt, N., Berners-Lee, T., Hall, W.: The Semantic Web Revisited. IEEE Intelligent Systems, 21 (3), 96-101, (2006).
10. Schulz, H.-J., Schumann, H.: Visualizing Graphs - A Generalized View. In IV 2006: Tenth International Conference on Information Visualization. pp. 166-173, IEEE CS, (2006).
11. Staab, S., Domingos, P., Mika, P., Golbeck, J., Ding, L., Finin, T., et al.: Social Networks Applied. IEEE Intelligent Systems, 20 (1), 80-93, (2005).
12. Tvarožek, M., Bieliková, M.: Collaborative Multi-Paradigm Exploratory Search. In: Proceedings of the Hypertext 2008 Workshop on Collaboration and Collective Intelligence, pp. 29-33. ACM Press, New York, (2008).
13. Tvarožek, M., Bieliková, M.: Visualization of Personalized Faceted Browsing. In: Forbrig, P., Paternò, F., Pejtersen, A. M. (eds.) IFIP: Human-Computer Interaction Symposium. IFIP 272, pp. 213-218. Springer, Heidelberg (2008).
14. Tvarožek, M., Bieliková, M.: Reinventing the web browser for the semantic web. In: WI-IAT ’09: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, IEEE CS, pp. 113–116, (2009).
15. Wilson, M. L., schraefel, M. C., and White, R. W.: Evaluating advanced search interfaces using established information-seeking models. In: Journal of the American Society for Information Science and Technology, 60 (7), 1407-1422, (2009).
16. Zhang, J., Marchionini, G.: Evaluation and evolution of a browse and search interface: relation browser. In: Proceedings of the 2005 national conference on Digital government research. pp. 179-188, dg.o, vol. 89. Digital Government Society of North America.