VisiNav: A System for Visual Search and Navigation on Web Data

Andreas Harth [a]

[a] *Institute AIFB, Karlsruhe Institute of Technology, Germany*

Abstract

Web standards such as RDF (Resource Description Framework) facilitate data integration over large number of sources. The resulting interlinked datasets describe objects, their attributes and links to other objects. Such datasets are amenable for queries beyond traditional keyword search and for visualisation beyond a simple list of links to documents. Given that data integrated from the open web exhibits enormous variety in scope and structure, the mechanisms for interacting with such data have to be generic and agnostic to the vocabularies used. Ideally, a system operating on web data is easy to use without upfront training. To this end, we present VisiNav, a system based on an interaction model designed to easily search and navigate large amounts of web data (the current system contains over 18.5m RDF triples aggregated from 70k sources). In this paper we introduce a formal query model comprised of four atomic operations over object-structured datasets: keyword search, object focus, path traversal, and facet specification. From these atomic operations, users incrementally assemble complex queries that yield sets of objects as result. These results can then be either directly visualised or exported to application programs or online services for further processing. The current system provides detail, list, and table views for arbitrary types of objects; and timeline and map visualisations for temporal and spatial aspects of objects.

Key words: web data, exploratory search, faceted navigation, semantic search, visual analytics, structured data search

1. Introduction

Keyword search over hypertext documents is an established technology and is used by a large majority of web users [14]. Search engines are popular because i) users are accustomed to the concept of hypertext – documents and links – and ii) search engines employ a simple conceptual model: the engines return those documents that match the specified keywords. Search engines operate over millions of documents which have been collected automatically, however, the functionality is limited: the engine returns only links to sites but not directly the actual answer or data items sought. Typical keyword phrases used for search are insufficient to specify a complex information need since they consist mostly of only a few words [14]; moreover, information expressed in documents in natural language is ambiguous and thus hard to process automatically.

Standards such as RDF and OWL (Web Ontology Language) provide means of describing objects and integrating data about objects from large numbers of sources which may only loosely coordinate. However, there is the open question of how end users should express complex queries over such datasets. A promising approach is to use a menu-based dialogue system in which users construct the query incrementally [34] [38]. Offering only valid choices ensures that the user can only pose queries which can be satisfied by the available data, preventing empty result sets. Designing an interaction model and developing a useable system for interrogating collaboratively-edited datasets poses several requirements:

(i) Intuitive use: both occasional users and

Email address: harth@kit.edu (Andreas Harth).

subject-matter experts should be able to interact with the data immediately. The user interface should be consistent and allow users to quickly derive results with a few clicks.

(ii) Universality: previous attempts at using structured information have been restricted to manually crafted domain-specific datasets since the data on the web lacked quantity (no general-domain information available) and quality (no shared identifiers, no interlinkage). Users should be constraint as little as possible in constructing queries while keeping the system easy to use.

(iii) Zero configuration: data on the web comes in an abundance of formats and vocabularies. Consequently, manual intervention is a labour intensive task, rendering manual intervention infeasible.

(iv) Flexibility: web data is often noisy and may contain duplicates, erroneous items, malformed syntax and incorrect formatting, so the user interface should be able to deal with irregular data gracefully. The system has to be flexible enough to deal with diverse content, both in terms of schema and noisy data.

(v) Scalability: since we target the web as a data source the system has to scale competently, which has implications on the architecture and implementation of the system. Also, quick response times enhance the user experience.

(vi) User satisfaction: the system should be visually appealing and users should be able to import the results of their information seeking task into application programs to get a sense of achievement immediately.

In this paper, we describe VisiNav, a fully implemented system [1] based on a visual query construction paradigm. Given that VisiNav operates over data rather than documents, operations offered can go beyond simple keyword searches and result visualisations can be more elaborate than just showing ten result documents. The users of the system can construct complex queries from several atomic operations. Our system is unique in that it is the first system which offers these features over datasets consisting of millions of RDF triples collected and automatically integrated from the open web.

The initial step in the proposed interaction model is typically a keyword search to locate objects, leveraging existing familiarity of users with search engines. In subsequent steps, users refine their query based on navigation primitives; as such, the interaction model leads to an explorable system that can be learned through experimentation. Since the system calculates the possible next steps based on the current state, only legal choices are displayed and thus the user can only compose queries which the system can answer.

The contribution of this research is two-fold:

– We adapt browsing and interaction mechanisms for schema-less interlinked datasets collected from the web. Previous work on visual interfaces either assumes datasets with limited variance in schemas [32] [19], limited expressivity of query operations [24] [17], or a strict partitioning of datasets [18]. VisiNav allows for users posing expressive query operations over interlinked web datasets with a large variance in schemas used.
– We introduce the notion of trees (or nested relations) as result model for web data exploration due to the use of the path traversal operation. Previous work assumes single [38] or multiple sets [18] [22] as result model.

The system satisfies a large fraction of the requirements set out for the Semantic Web Challenge. The data used in the current VisiNav system is under diverse ownership (data from over 70k sources), the dataset is heterogeneous (more than 21k different vocabulary URIs) and contains substantial quantities of real world data (more than 18.5m statements). Meaning is represented and processed using Semantic Web technologies (OWL and reasoning [16]). The system incorporates ranking [12] to prioritise data and data items, and utilises Web standards such as HTML, CSS and JavaScript for the user interface.

We provide an overview of the user interface in Section 2, present and motivate the choice of query operations and formalise the result model in Section 3, and discuss design choices in Section 4. Section 5 covers related work, and Section 6 concludes.

2. Overview and Preliminaries

In the following, we describe the characteristics of the target dataset collected from the web, present example queries, and introduce the conceptual model and selected tasks carried out over such datasets.

[1] `http://visinav.deri.org/`

2.1. *Web Data*

Common to data currently found on the Web in structured formats (microformats, XML, RDF) is that data publishers take a loosely object-centred view. RDF in particular uses URIs [2] as global identifiers for objects, which, if multiple sources reuse identifiers, leads to an interconnected object space encoded in a graph-structured data format. Currently, reuse of identifiers is particularly common in social networking and social media data, expressed in FOAF [3] for people, SIOC [4] for online community sites, and DC [5] for documents and other media. While a large number of current RDF files use a mix of these vocabularies, data publishers use a plethora of other vocabularies, too.

For example, the most diverse dataset used during our experiments contains data from 70k sources using over 21k different vocabulary URIs (classes and properties). The amount of variance in the dataset is enormous; as a result, manual intervention should be as minimal as possible, and the system should make as little assumptions about the schema used as possible to arrive at a general and universal method for exploring the dataset.

2.2. *Conceptual Model*

Norman [29] argues that the conceptual model of a system has to fit the user's own conceptual model about it. We therefore introduce first the assumed conceptual model, i.e. what the users have to know about the system before interacting with the system.

VisiNav's conceptual model for navigation assumes an object-oriented view, describing objects, their attributes and links to other objects. Attributes of objects are expressed using datatype properties, and links to other objects are specified using object properties [6]. Objects and properties are identified via identifiers (e.g. URIs). Attributes can have datatypes such as integer or date. Please note that there is no clear distinction between instance-level objects and schema-level ones – classes and properties can be instances themselves.

Users perceive and act on objects, in-line with early graphical user interfaces [25]. In general, there is a 1:1 correspondence between the objects in the dataset and the objects displayed to the user, loosely following the "naked objects" approach [7]. Users are able to search and navigate the objects in the dataset; a user query yields objects as a result. Users can choose to display the result set in detail, list, or table view; optionally, a timeline or map visualisation is available if the result objects contain suitable information. In addition, users are able to export the results to application programs or services.

2.3. *Tasks*

In the following, we introduce a set of example queries that are prototypical for the type of queries a user can pose to the system. Given the wide availability of information about people and communities, we use the social network scenario to study user interfaces on collaboratively-edited datasets. However, the interaction model and the implemented system are domain independent and thus applicable to any object-structured dataset. We list a number of example queries – that can be answered with currently available web data – with increasing complexity in Table 1 [7].

Query	Description
1	objects matching the keyword phrase "tim berners-lee"
2	information available about `timbl:i`
3	objects `foaf:made` by `timbl:i`
4	`sioc:Posts` `foaf:made` by `timbl:i`
5	people that `timbl:i` `foaf:knows`
6	objects `foaf:made` by people that `timbl:i` `foaf:knows`
7	locations where people that `timbl:i` `foaf:knows` are `foaf:based_near`
8	objects of `rdf:type` `foaf:Person`

Table 1
Example queries. Identifiers in `typewriter` are part of the query.

[2] Uniform Resource Identifiers, `http://www.rfc-editor.org/rfc/rfc3305.txt`
[3] Friend-of-a-Friend, `http://foaf-project.org/`
[4] Semantically Interlinked Online Communities, `http://sioc-project.org/`
[5] Dublin Core, `http://dublincore.org/`
[6] as specified in OWL, Web Ontology Language, `http://www.w3.org/2004/OWL/`
[7] `timbl:i` expands to `http://www.w3.org/People/Berners-Lee/card\#i`; throughout the paper we assume the standard namespace prefixes for `rdf`, `rdfs`, `owl`, `foaf`, `sioc` and `dc`

VisiNav supports three generic views for displaying results: detail, list, and table. The detail view shows all properties pertinent to a single object in a two column-format. The list view shows a few selected properties in a list of objects of typically ten objects at a time, similar to search engine results (Figure 1). In the list view, users can choose to download the data in RSS. The table view shows all datatype properties of the current result in a tabular view, typically up to hundred at a time, similar to spreadsheets. In the table view, the users can download a version of the data displayed in comma separated value (CSV) format for further processing.

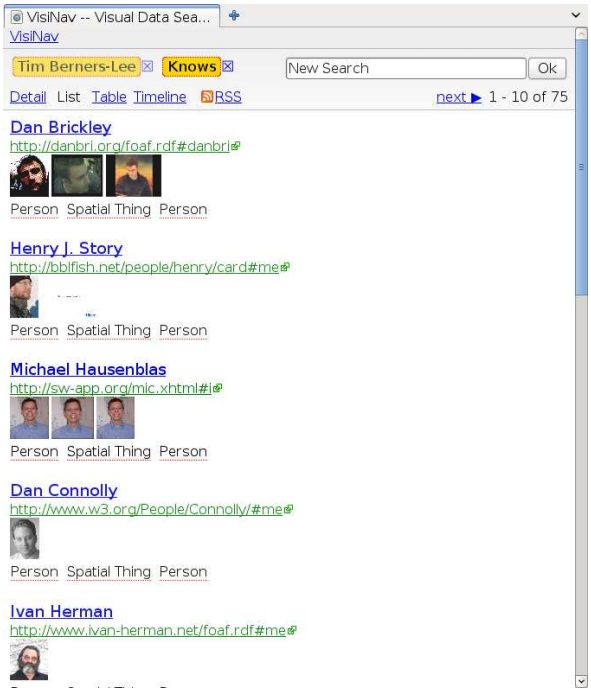

Fig. 1. List view of the query "people that `timbl:i` knows" (Query 5).

VisiNav currently supports visualisations for temporal and spatial aspects of objects. The system renders temporal attributes (i.e. those with datatype `xsd:date` or `xsd:dateTime`) in a SIMILE timeline view [8]. In the timeline view, users can choose to download the data in iCal format and thus import the data into calendaring applications. Spatial properties (i.e. objects with `geo:lat` and `geo:long` properties) are rendered in a Google map view [9]. The map view displays all objects in a result set that have both `geo:lat` and `geo:long` properties attached to them.

3. Query Operations and Result Model

In the following we introduce VisiNav's query operations and describe how to compose complex queries from atomic operations, and present a mapping of query results encoded as RDF into a generic (i.e. schema-agnostic) object model for manipulation within a programming language.

3.1. *Query Operations*

To construct queries, users select operations that are common to existing faceted browsing and navigation systems. We list the feature matrix of these systems in Section 3.2; we argue that by using a set of features used in existing systems we capture the community consensus of operations that are deemed necessary and useful for interacting with object-structured datasets.

- **Keyword Search** ($\mathcal{K}$)
  quad Users may specify keywords to pinpoint objects of interest. The operation leads to an initial set of results based on a broad matching of string literals connected to objects. We perform matching on keywords without manually extending the query for synonyms or other natural language processing techniques. Rather, we leverage the noise in web data, i.e. the fact that the same resource might be annotated using different spellings or different languages. Keyword search has the interesting property that the users do not need to know the schema of the data, enabling users to pose queries without previous domain knowledge.
- **Object Focus** ($\mathcal{O}$) The operation is similar to following a hypertext link in a web browser. From a set of results or a single result, the user selects an object which is used to create a new query and returns a result set containing the object. The result of a object focus selection operation is always a result set with a single object.
- **Path Traversal** ($\mathcal{P}$) Rather than arriving at a single result by selecting a focus object, users are also able to navigate a path along an object property to establish a new set of results. Users can select an object property which allows them to perform a set-based focus change, i.e. they follow a certain path, either from a single result or a set of results.

[8] `http://www.simile-widgets.org/timeline/`
[9] `http://maps.google.com/`

– **Facet Selection** ($\mathcal{F}$) Another way of restricting the result set is via selecting facets. A facet is a combination of a property and a literal value or an object (distinguishing between datatype and object properties). Facets are calculated relative to the current result set. Based on derived facets, the user can reformulate the query and obtain increasingly specific result sets.

3.2. *Feature Survey*

Faceted browsing [38] has become popular as an interaction form for sites which contain structured data, such as Ebay.com and Yelp.com [13]. In addition, the faceted browsing model has been adopted in systems which operate over RDF datasets.

Our choice of a few core operations represents a trade-off decision between complexity of queries and ease of use. Table 2 compares the core features of browsing systems taking an object-oriented perspective. Our system uses the core set of query primitives common to a range of established browsing and navigation systems for semi-structured data, providing evidence that the selection of features in our system represents a consensus in the community. This suggests that a sizeable user community is able to conceptually grasp the query operations offered by VisiNav.

System	$\mathcal{K}$	$\mathcal{O}$	$\mathcal{P}$	$\mathcal{F}$	Results
Magnet [32]	x	x	-	x	set
MuseumFinland [19]	x	x	-	x	set
GRQL [2]	-	x	x	o	set
SWSE [17]	x	x	-	-	set
/facet [15]	x	x	-	x	set
BrowseRDF [30]	x	x	-	x	set
ESTER [3]	x	x	-	x	set
TcruziKB [27]	x	x	x	-	set
Tabulator [24]	-	x	x	-	tree
Falcons [6]	x	x	-	o	set
Humboldt [22]	x	x	x	x	sets
Parallax [18]	x	x	x	x	sets
ECSSE [8]	x	x	-	-	set
VisiNav	x	x	x	x	trees

Table 2
Comparison of query operations (Keyword Search $\mathcal{K}$, Object Focus $\mathcal{O}$, Path Traversal $\mathcal{P}$, Facet Selection $\mathcal{F}$) of systems operating over object-structured data (x = yes, - = no, o = only `rdf:type` facets).

3.3. *Formal Query Model*

An important aspect of usability of user interfaces is consistency and predictability [28][29]. We present a formalisation of query operations which are used to implement the system to ensure the system behaves consistently and predictably, resulting in improved usability.

We assume a collection of objects U identified via URIs or blank nodes. Objects are described using datatype properties P_D with literal attributes L and object properties P_O denoting links to other objects U. Properties can be objects as well: $P_D \in U$ and $P_O \in U$. We assume one relation $quad(s, p, o, c) = (U \times P_D \cup P_O \times L \cup U \times U)$, modelling the source of subject/predicate/object triples as fourth field. We also assume a relation $text(t, s) = (T \times U)$ which models an inverted index over tokens T derived from the literals L using a string tokenisation method [10] .

Users start a query building process via specifying a keyword, and then iteratively refine the query, or start a new query based on the available choices offered by the result set. Figure 2 illustrates the high-level interaction cycle. During each interaction step, the users select one of the operations, and the system returns with a page containing a visual representation of the query, state, and query results. The query results are first displayed in a familiar list view, but users can choose other views or visualisations depending on the properties of the objects in the result set.

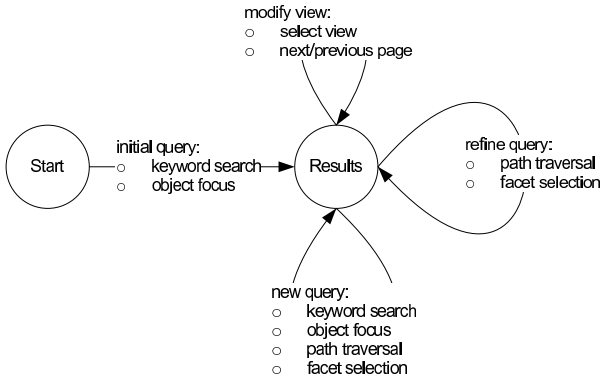

Fig. 2. Interaction flow diagram.

A query Q expressed over the object set U described by the *quad* and *text* relations returns as re-

[10] A number of tokenisation methods have been proposed in information retrieval – our approach is agnostic to the tokenisation method used

sult a number of sets $R_{0...n}$ where n is the number of path traversal operations. The individual operations are defined in terms of relational algebra[1] as follows:

- Keyword Search: the k operation returns a set of objects which satisfy the keyword selection criteria, and an empty result if there is no match. The operation is defined as:
  $k(k_{0..m}) = \pi_s(\sigma_{t=k_0} text) \cap \ldots \cap \pi_s(\sigma_{t=k_m} text)$
- Object Focus: the o operation returns an object with the specified URI, if the object exists in the dataset, and an empty result otherwise. The operation is defined as:
  $o(s') = \pi_s(\sigma_{s=s'} quad)$
- Path Traversal: the p operation returns a set of objects which satisfy the path selection criteria. p always returns a result, since the user can only choose valid paths. The operation is defined as:
  $p(p') = \pi_{s,o}(\sigma_{p=p'} quad)$
- Facet Selection: the f operation returns a set of objects which satisfy the selection criteria. f always returns a result, since the user can only choose valid facets. The operation is defined as:
  $f(p', o') = \pi_s(\sigma_{p=p' \wedge o=o'} quad)$

3.4. *Result Trees*

A single result set is sufficient for keyword searches, object focus (specifying an initial result set), and facet selection (reducing the size of the result set). The path traversal operation is different: traversing a path restricts the old result set (to the objects with the specified property) and generates a new result set (with the objects that are connected via the specified property). Thus, iterative application of the restriction and navigation operations leads to sets of focus nodes $R_{0...n}$ which are connected to each other via the specified property in the path traversal operation.

Consider, for example, the query "objects `foaf:made` by people that `timbl:i foaf:knows`" (Query 6). That query yields three result sets R_0, R_1, R_2. We assume that the query was constructed in the following way: the user uses object focus operation to specify the object URI of Tim Berners-Lee R_0 = `timbl:i`, from there performs path traversal along the `foaf:knows` property R_1 = people that Tim knows, and from there performs another path traversal along the `foaf:made` property yielding R_2 = things made by people Tim knows. An example result tree is shown in Figure 3. An alternative representation of the result trees is in form of nested relations.

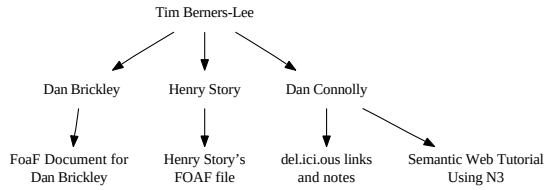

Fig. 3. Partial result tree for query "objects `foaf:made` by people that `timbl:i foaf:knows`" (Query 6). Labels displayed instead of URIs for clarity.

3.5. *Topical Subgraphs and Result Objects*

To be able to establish the connections between objects shown in Figure 3 from $R_{0...n}$, and to be able to render properties of objects in the result sets, the system requires more information than just object identifiers. The step involves a number of challenges:

- The dataset contains an RDF graph. For preprocessing and displaying the objects we require to manipulate the objects using a programming language.
- To minimise the amount of data fetched and increase response time, the system should only fetch additional information about objects that are ultimately rendered. The result sets may contain thousands of object identifiers, however, typically only a few are displayed at once.
- Depending on view or visualisation, only partial sections of the result object are required. For example, the list view only requires a few properties of an object.
- Since the data is very diverse and the amount of information available for objects varies enormously, we need to pick the best data for display. For example, displaying thousands of quads in the results view will likely overwhelm the users.
- To provide a meaningful display of information, the system requires data items to be ordered, showing the most important values first.

To solve the mismatch between RDF graph and objects that can be manipulated in a programming language, we use a generic implementation that stores properties of objects similar to ActiveRDF [31]. We cannot use mapping tools such as RDFReactor[11] since objects aggregated from many

[11] `http://rdfreactor.semweb4j.org/`

web sources do not strictly adhere to the vocabulary specifications. In addition, we require a way of generic processing of objects regardless of type.

We employ ranking to minimise the amount of data fetched and therefore reduce response time. We assume a global rank for each element in U. Full treatment of the ranking procedure is beyond the scope of this paper (see [12] for details), however, one can assume the occurrence count of a node as sufficiently useful approximation for the popularity of the node.

The system first orders the objects in $R_{0\ldots n}$ according to their global rank and then selects the objects in the range that should be displayed. Depending on the selected view, the system constructs a "topical subgraph" with all the information required to displaying the object. Given the topical subgraphs adorned with result nodes of level $0 \ldots n$ in conjunction with the initial query we are able to construct paths connecting the objects in various layers of $R_{0\ldots n}$.

Once the topical subgraph is parsed into a programming language object, the system orders the properties, literals and associated objects pertaining to the object according to the global rank. Thus, the algorithms selecting the pieces of data to display can retrieve elements of objects in order and more likely display meaningful information than with random order.

4. Discussion

In the following we discuss design decisions that directly follow from the properties of our web dataset. While existing work on faceted search covers datasets with little variation (e.g. [9] evaluates datasets with 14 to 815 facets), our dataset contains over 21k different vocabulary URIs (properties and classes) mandating a different approach to faceted navigation.

4.1. *Ranking*

Throughout the user interface the system has to decide how to order data items. For example, the set of result objects in the list view has to be ordered, the predicate/object combinations in the detail view have to be ordered, and the properties in the table view have to be ordered. In lieu of a fixed schema to code against, and given the lack of rendering descriptions in e.g. Fresnel[4], ranking becomes the method of choice for deciding on how to arrange items in the visualisations.

4.2. *Facets in Content*

In faceted navigation the facets are aggregated from all available items in the current result set. Various heuristics are applied to cut down the amount of processing and the size of facets presented to the user [9]. For web data on a large scale, these heuristics are currently missing. Without such heuristics, computing an aggregated view of all facets in a large result set (such as for query 8, `rdf:type foaf:Person`) becomes very expensive. In addition, it is not clear how to visualise the aggregated view consisting of potentially hundreds of facets.

Given these issues, we exploit the fact that the detail view contains all items required to specify additional facets restriction or path traversal operations. Hence, the detail view is also used to specify parameters for additional query operations. By using in-content facets in conjunction with drag-and-drop actions reduces visual clutter and thus increases the ink-to-data ratio, one of the tenets of good visual design [35].

That use of in-content facets is also the reason why the interaction always starts with a keyword query or object focus operation: that users can pick up a facet or path traversal from the initial result set as input to both the current query or a new query. The alternative of just using point-and-click actions would require a large number of buttons cluttering the user interface (for example, the user interface would require three buttons next to each object to allow for the object focus, add facet to current result, and use facet to construct new query operations).

4.3. *Evaluation*

The system underwent several user tests and expert reviews during design and development. We used the "thinking aloud" method [20] on prototypes using both the Mondial dataset [26] and a Web dataset, which led to several changes and adaptations of the system. Complete user tests are subject to future work.

5. Related Work

A number of systems exist that operate over RDF data. Broadly speaking, most of the generic data

browsing systems follow the "naked objects" approach [7]: objects are displayed directly without type-specific styling; as a result, user interfaces can be generated automatically and do not need domain-specific adaptation. Ideally, systems would provide means for styling the display of objects based on templates (if there exists a template for a specific type of objects) similar to Fresnel [4] or Tal4Rdf[12], however, declarative templating languages for RDF is still a developing area of research.

Most of the current RDF browsers implement a series of design steps that can be described in terms of the Semantic Hypermedia Design Method [10]. The method describes techniques to perform requirement analysis, data modelling tasks, defining user interactions and the final display of objects. In a sense, our approach is a generalised version of the method, where, rather than specifying data and interaction model in concrete terms pertaining to a schema, we describe our abstract interface – the information exchange between users and system – in terms of general query operations, and result pages as generic rendering of trees of objects. Our concrete interface – the look and feel – is implemented using a multi-layered rendering pipeline spanning server (RDF, queries and XML) and client (XSLT and CSS), with only minimal assumptions on the schema used to describe the objects.

Rather than operating on objects described in RDF documents, search engines such as Swoogle [11] and Sindice [36] assume a document-centric model. The main operation of the systems is to return RDF documents which contain specified keywords. As a result, the systems do not allow to integrate data about the same object originating from different RDF documents; the systems also lack functionality such as facet selection ond path traversal which require an object-oriented perspective rather than a document-centric one.

Systems for browsing and displaying semi-structured data range from quite basic browsing facilities (allowing only to navigate from one object to another) to systems including constructs such as negation [30] or nested facets [37]. Semantic MediaWiki [23] allows to embed complex queries into wiki pages, however, these queries typically represent a once-off query and do not allow for iteratively refining results. Typical linked data browsers fetch data on demand and thus lack features such as keyword search or facet selection which require query processing capabilities over the entire collected and integrated dataset.

The systems most closely related to our system in terms of features are GRQL [2], Humboldt [22] and Parallax [18]. GRQL relies on schema information rather than automatically deriving the schema from the data itself, a feature required for web data which does not necessarily adhere to the vocabulary definitions. GRQL lacks keyword search, a useful feature when operating on arbitrary data, since keywords are independent of any schema. Rather than allowing arbitrary facets, GRQL allows to restrict based on the `rdf:type` predicate. GRQL is, to our knowledge, the earliest system that provides functionality to perform set-based navigation. Parallax [18] is a recent system which exhibits browsing features similar to ours. However, Parallax operates over the Freebase dataset which is manually curated and requires the user to select one of the many disparate datasets contained in Freebase; our system operates over a fully integrated RDF dataset collected from the web. In contrast to Parallax which lacks ranking, VisiNav prioritises the display of data based on global ranks. Although Parallax uses multiple result sets, the connections between the result sets are not propagated to the level of the user interface; our system maintains result paths in the results trees. Finally, we provide a set of export plug-ins which allows to directly load result sets into application programs and online services for display or further processing.

NLMenu [34] is an early system advocating the use of multi-step query construction based on menus. Faceted browsing [38], while less expressive in terms of the complexity of queries, has become popular and is used on e-commerce sites such as Ebay.com. Polaris [33] provides complex query and aggregation operations, however, operates over relational data and thus requires a priori knowledge about the schema used. Cammarano et al. [5] describe a method to match data with visualisation specifications based on schema matching algorithms. However, their method does not include means for users to graphically construct the visualisation specification.

While natural language question answering interfaces are judged preferable to other interfaces by users [21], they are not in common use today. Despite user training with regards to the capabilities and limitations of a natural language system, users quickly develop negative expectations about the system due to the relatively high error rates in parsing

[12] `http://champin.net/t4r/`

and interpreting natural language [34]. Users are unable to understand the limitations of such systems, that is, to distinguish between conceptual coverage (i.e. does the dataset contain the answer?) and linguistic coverage (i.e. is the system capable of parsing the query?). One way of reducing the error rate in parsing queries could be restricting the type of questions a user may ask to the system.

6. Conclusion

The ability to integrate hitherto disparate pieces of data and thus enable applications to re-purpose data in unanticipated ways enables novel applications but at the same time introduces new challenges to the design of user interfaces. In this paper, we introduce a general, formal model for searching and browsing objects, and present a prototype system implementing these ideas. The interaction model provides operations that allow users to explore and visualise data without requiring knowledge about the schema of the data. Users are able to learn the structure of the domain of interest while interacting with VisiNav.

The system provides a universal way of interacting with any RDF dataset, without manual curation of the data or domain-specific adaptation of the interface, and thus can directly applied to data from the open web. In addition, the system allows to rapidly develop and employ data integration systems in more confined environments such as intranets where the data is typically domain-specific and the number of data sources confined, which opens the door for cost-effective manual curation of the data. In such environments, providing domain-specific widgets for visualising popular types of objects (similar to the current timeline and map visualisations) is a way to further increase the overall utility of the system.

Potential areas of future work include to investigate the possibility of mapping the generic query operations to other modalities, such as natural language or speech input. Also, we would like to test our system on new hardware such as touch screens, and investigate how groups of users can use VisiNav to collectively explore and analyse data integrated from vast amounts of sources.

Acknowledgements

I gratefully acknowledge the comments of the participants of the user studies, and discussions with Paul Buitelaar, Brian Davis, Stefan Decker, Renaud Delbru, Armin Haller, Aidan Hogan, Sheila Kinsella, Axel Polleres, Rene Schubotz, and Jürgen Umbrich.

A large portion of this work has been carried out at the Digital Enterprise Research Institute (DERI) in Galway and has been partially supported by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] N. Athanasis, V. Christophides, and D. Kotzinos. Generating on the fly queries for the semantic web: The ics-forth graphical rql interface (grql). In *3rd International Semantic Web Conference*, pages 486–501, Nov 2004.

[3] H. Bast, A. Chitea, F. Suchanek, and I. Weber. Ester: efficient search on text, entities, and relations. In *30th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 671–678, 2007.

[4] C. Bizer, E. Pietriga, D. Karger, and R. Lee. Fresnel: A browser-independent presentation vocabulary for rdf. In *5th International Semantic Web Conference*, November 2006.

[5] M. Cammarano, X. L. Dong, J. Talbot, B. Chan, J. Klingner, A. Halevey, and P. Hanrahan. Visualization of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1200–1207, 2007.

[6] G. Cheng, W. Ge, and Y. Qu. Falcons: searching and browsing entities on the semantic web. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1101–1102. ACM, 2008.

[7] L. L. Constantine. The emperor has no clothes: Naked objects meet the interface, 2002.

[8] R. Cyganiak, M. Catasta, and G. Tummarello. Towards ecsse: live web of data search and integration. In *Semantic Search 2009 Workshop*, 2009.

[9] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 3–12. ACM, 2008.

[10] S. S. de Moura and D. Schwabe. Interface development for hypermedia applications in the semantic web. In *Joint Conference 10th Brazilian Symposium on Multimedia and the Web & 2nd Latin American Web Congress*, pages 106–113, 2004.

[11] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM international*

conference on Information and knowledge management, pages 652–659. ACM, 2004.

[12] A. Harth, S. Kinsella, and S. Decker. Using naming authority to rank data and ontologies for web search. In *8th International Semantic Web Conference (ISWC2009)*, October 2009.

[13] M. A. Hearst. Uis for faceted navigation: Recent advances and remaining open problems. In *HCIR08 Second Workshop on Human-Computer Interaction and Information Retrieval*, October 2008.

[14] M. Henzinger. Search Technologies for the Internet. *Science*, 317(5837):468–471, 2007.

[15] M. Hildebrand, J. van Ossenbruggen, and L. Hardman. /facet: A browser for heterogeneous semantic web repositories. In *5th International Semantic Web Conference*, pages 272–285, Nov 2006.

[16] A. Hogan, A. Harth, and A. Polleres. Saor: Authoritative reasoning for the web. In *3rd Asian Semantic Web Conference*, pages 76–90, 2008.

[17] A. Hogan, A. Harth, J. Umrich, and S. Decker. Towards a scalable search and query engine for the web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1301–1302. ACM, 2007.

[18] D. F. Huynh and D. Karger. Parallax and companion: Set-based browsing for the data web. Available online (2008-12-15) http://davidhuynh.net/media/papers/2009/www2009-parallax.pdf.

[19] E. Hyvnen, 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.

[20] A. H. Jørgensen. Using the thinking-aloud method in system development. In *Proceedings of the third international conference on human-computer interaction on Designing and using human-computer interfaces and knowledge based systems (2nd ed.)*, pages 743–750. Elsevier Science Inc., 1989.

[21] E. Kaufmann and A. Bernstein. How useful are natural language interfaces to the semantic web for casual end-users? In *6th International Semantic Web Conference*, pages 281–294, Nov 2007.

[22] G. Kobilarov and I. Dickinson. Humboldt: Exploring linked data. In *Linked Data on the Web Workshop*, 2008.

[23] M. Kroetzsch, D. Vrandecic, and M. Voelkel. Semantic mediawiki. In *ISWC 2006*, pages 935–942, 2006.

[24] T. B. 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 *In Procedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06*, page 06, 2006.

[25] D. E. Lipkie, S. R. Evans, J. K. Newlin, and R. L. Weissman. Star graphics: An object-oriented implementation. In *SIGGRAPH '82: Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, pages 115–124. ACM, 1982.

[26] W. May. Information extraction and integration with FLORID: The MONDIAL case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999. Available from `http://dbis.informatik.uni-goettingen.de/Mondial`.

[27] P. Mendes, B. McKnight, A. Sheth, and J. Kissinger. Tcruzikb: Enabling complex queries for genomic data exploration. *IEEE International Conference on Semantic Computing*, pages 432–439, Aug. 2008.

[28] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

[29] D. A. Norman. *The Design of Everyday Things*. Basic Books, September 2002.

[30] E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for rdf data. In *5th International Semantic Web Conference*, Nov 2006.

[31] E. Oren, R. Delbru, S. Gerke, A. Haller, and S. Decker. Activerdf: object-oriented semantic web programming. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 817–824. ACM, 2007.

[32] V. Sinha and D. R. Karger. Magnet: supporting navigation in semistructured data environments. In *ACM SIGMOD International Conference on Management of Data*, pages 97–106, 2005.

[33] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[34] C. W. Thompson, K. M. Ross, H. R. Tennant, and R. M. Saenz. Building usable menu-based natural language interfaces to databases. In *9th International Conference on Very Large Data Bases*, pages 43–55, 1983.

[35] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2001.

[36] G. Tummarello, E. Oren, and R. Delbru. Sindice.com: Weaving the open linked data. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, volume 4825 of *LNCS*, pages 547–560. Springer Verlag, November 2007.

[37] M. Tvarozek and M. Bielikova. Adaptive faceted browser for navigation in open information spaces. In *16th International Conference on World Wide Web*, pages 1311–1312, 2007.

[38] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *SIGCHI Conference on Human factors in Computing Systems*, pages 401–408, 2003.