2013 IEEE 9th International Conference on e-Science

Lens: A Faceted Browser for Research Networking Platforms

Richard Whaling, Tanu Malik, Ian Foster
Computation Institute,
University of Chicago and Argonne National Laboratory
Chicago, USA
{rwhaling, tanum[1], foster}   @ci.uchicago.edu

***Abstract*— Research networking platforms, such as VIVO and Profiles Networking provide an information infrastructure for scholarship, representing information about research and researchers—their scholarly works, research interests, and organizational relationships. These platforms are open information infrastructures for scholarship, consisting of linked open data and open-source software tools for managing and visualizing scholarly information. Being RDF based, faceted browsing is a natural technique for navigating such data, partitioning the scholarly information space into orthogonal conceptual dimensions. However, this technique has so far been explored through limited queries in research networking platforms--not allowing for instance full graph based navigation on RDF data. In this paper we present Lens a client-side user interface for faceted navigation of scholarly RDF data. Lens is based on Exhibit, which is a lightweight structured data-publishing framework, but extends Exhibit for expressive SPARQL-like queries and scales it up for navigating amounts of RDF data. Lens consumes data in VIVO ontology, the de facto schema for researcher networking systems. We show how Lens provides better usability over current faceted browsers for research networking platforms.**

Keywords—researcher profiling systems, RDF databases; faceted search; ontologies; user interfaces (key words**)**

I. Introduction

Scientific activity for several decades was viewed as a purely intellectual exercise. Over the last decade, it is percieved as a complex adaptive system that requires various inputs elements, such as monetary resources and human resources, and output elements that focus on knowledge creation and its economic, social and human impact [2]. The understanding of this complex adaptive system is essential for policy makers, funding agencies, and researchers---Policy makers want to empirically quantify the impact of science dollars on job creation; funding agencies want to comprehend the science programs to support, and researchers want to identify and engage experts whose scholarly work is of value to one's own.

Research networking platforms such as VIVO [2] and Profiles Networking Software [10] and others [19] have emerged in response to this need of understanding complex research networks. The platforms provide an information infrastructure for scholarship, represent information about research and researchers, inlcuding but not limited to their scholarly works, research interests, and organizational relationships. To enable such diverse interests and research works, often obtained from heterogenous data sources, most platforms adopt Semantic Web principles to model the data and use tools for reasoning about researcher relationships and networks. To provide connections between people, organizations, works, and funding over time and place, the platforms provide a myriad of visualization interfaces. In our evaluation of profiling platforms, we have identified three kinds of interfaces: (1) a simple keyword search, (2) explicit SPARQL queries, and (3) graph visualisation. However, none of these interfaces provide the power of faceted search that makes available a number of co-existing dimensions that can be simultaeously browsed by the user. In keyword search, a user has to guess the right 'search' terms; SPARQL queries can lead to inconclusive searches; and graph visualization does not provide an intuitive feel for the data as a whole just by looking at the available or most relevant facets. This intuitive feel is important for policy makers and researchers to get an overall understanding of research investment of say a department, an institute or university, and explore areas of deep investment.

Several faceted browsers [5,6,12,13,14] have been investigated in the recent past, some of them are RDF-based. Amongst the RDF-based facet browsers, the degree of data exploration is based on the extent of graph navigation made possible by the user interface. Simple RDF-based browsers assume predicates as facets and determine elements to show based on basic facet selection and/or conjunctions or disjunctions of the basic selections. More advanced RDF browsers allow for exploring relationships between subjects based on properties of nodes that they are connected to. For instance, "determine all subjects who know somebody, who in turn knows somebody named 'Whaling'. However, exploring a large number of pathways or facets of facets (indirect facet chains of arbitrary length [17]) in an efficient manner is still a challenge for RDF-based facted browsers especially since paths may be of arbitrary length and relationships can be

This work is supported by NSF Grant SBE-1160899 and seed funding through the University of Chicago.
[1]: *Contact Author*

defined on arbitrary predicates. Full graph navigation can, however, aid in answering complex questions.

Profiling systems typically use a standard ontology (VIVO ontology [10]) for describing RDF data, thus making nodes and properties known apriori. This can help us in optimizing for efficient graph navigation. However, none of the current open-source RDF-based faceted browsers index data based on an underlying ontology thus preventing the user from visualizing the end-result with natural categorizations of the content.

In this paper, we present Lens, an open-source faceted browser for research profiling networks that uses RDF data based on the VIVO ontology. Lens can be used to answer a variety of questions about a research organizations, such as determining "who published in what area?", "which researchers received most funding from a federal program?", or "show all the faculty in areas of informatics".

Internally, Lens is based on Exhibit, which is a lightweight data publishing plaform and supports RDF data. Our choice of Exhibit is based on two current practices: First research profile servers are often based in IT departments and for security reasons do not support SPARQL end points, which make it hard to support a faceted browser based on server-side computation. Secondly, Exhbit offers a very intuitive user interface that can be used for basic RDF navigation. Exhbit, however, does not support graph navigation or provides support for ontologies. We have extended Exhibit in the following ways:

- We improve expressivity of faceted queries in Exhibit to support graph navigation, making it possible to support join queries, inverse join selection or a combination thereof.
- We improve the ability of Exhibit to accept ontology-based data and use generic methods to improve the efficiency of graph traversal. In particular, facet metrics, such as predicate and join frequency are used to optimize traversal.
- We make Exhibit scalable for large amounts of data by bypassing nodes and predicates in RDF data that are not used in faceted search.

Our experiments are conducted on publicly available data about researchers at the University of Chicago. This data currently includes their directory, publication, grant and patent information. The experiments show how computing statistics on the data optimizes graph navigation and these statistics further help us in improving loading and querying time, thus improving the scalability of the data. Lens is being built as a prototype for the university profiling system (http://profiles.uchicago.edu) to help researchers find collaborators and find and explore departmental and institutional strengths.

The remainder of the paper is organized as follows. Section 2 presents related work in the areas of researcher profiling platforms and faceted browsing. Section 3 presents Lens, our faceted browser that supports graph navigation but is optimized for graph traversal. We show a prototype that can help a user explore research strengths in an organization with minimal effort. In Section 4, we experimentally evaluate the performance of Lens. We conclude in Section 5.

II. Related Work

A. Researcher Profiling System Interfaces

Exploring relationships is fundamental to research profiling platforms. Towards this, visualization interfaces such as temporal trends, geospatial maps, expertise profiles, and scholarly networks have been explored. These visualizations are useful for aggregating vast amounts of information, but not for classifying information as in faceted search.

VIVO supports a faceted search interface that allows a user to classify research products and affiliations of a given person using facets such as organizations, papers, grants, courses, and events. Their implementation constructs a syntactic Lucene text search on available research products and associates one or more products with a given facet. For e.g., if the word "NSF" appears in the same document as the word "Ian Foster" it is associated with the "Grant" facet, even if the document mentions that Ian Foster participated in a NSF workshop, i.e., the word "NSF" appeared in a different context. Thus they do not fully represent relationships of the person with the facet values.

B. Faceted Browsers

Several faceted browsers have been proposed based on the faceted browsing technique first proposed by Hearst *et. al* [11]. In the classic faceted model described by Hearst *et. al.*, the items to be searched constitute a single table in a relational database, and every user interaction can be mapped to adding or removing a condition from the WHERE clause of a SQL query. This conjunctive/disjunctive model is usually sufficient for homogeneous data sets, such as online merchandise, or a library catalog in which the tacit assumption is that only one set of items has facets viz. the books or products. However, semi-structured data containing inter-related items of heterogeneous types presents challenges in which facets themselves may be composed of facets.

Several RDF-based faceted browsers [5,6,12,13,14] have also been proposed varying on aspects such as browsing, filtering or query expressivity. Oren et. al developed a faceted browser that explores graph-based relationships by expressing joins over source and target RDF nodes [17]. Their implementation however, is based on non-standard ActiveRDF [16], an object-oriented API for arbitrary RDF data. More recently, W3C standard SPARQL-based faceted browsers have been proposed [15][20]. However, the expressivity of join realized in a faceted UI is limited [15] or unknown [20]. None of these RDF browsers are open-source for us to rapidly determine if they can work with scholarly data. Even if they were available, it is not clear that they can build facets by reading typed information from ontology-based RDF data. The ontology aspect is mostly considered outside the scope of the RDF browsers [17].

Exhibit, from MIT, is an open-source, lightweight client-side data-publishing framework that creates Web pages with rich, dynamic visualization of structured data, and with faceted browsing [13]. Exhibit supports RDF data, and being open-source is great for experimentation. It however has limited support for exploring graph-based relationships. In our work, we have used Exhibit but made our joins as expressive as proposed in [5], while exploiting the native VIVO ontology for RDF processing.

III. LENS: A Faceted Browser For Researcher Networking Platforms

The Lens faceted browser presents categories of the scholarly domain such as title, organization, publication journal, publication year, and grants program to a user. The browser assists the user in filtering information about the domain by selecting, combining, or joining categories and seeing the resulting information. By interacting with these categories users can query and manipulate scholarly information in an intuitive manner without having to construct logically sophisticated join queries. Thus users can relate topics, publications or grants to people, or vice versa, without a priori knowing the people in a particular area. In this section, we describe how queries are internally constructed, and relationships inferred based on user interactions. We first describe two background topics: (i) the nature of our data resulting from the underlying scholarly ontology, and (ii) basic faceted browsing as available through the open-source Exhibit system. We then describe graph operations on RDF data and show how they are efficiently implemented in Lens, using knowledge of the data. Finally, we show interaction with our prototype system.

A. Background

Scholarly Data: RDF schema describes classes or resources and their relationships. It defines the type of a resource and all relationship edges have a domain and range constrained to one or more type of resources. The VIVO Ontology [3], which includes classes and properties from external ontologies, such as BIBO [21], SKOS [22], FOAF [8], Event [18], GeoPolitical [1], and Eagle-I [23], is further customized to suit the incoming data from a variety of resources. For instance, we have expanded our own experimental data based on the VIVO ontology with additional data from a variety of sources, including NSF/NIH feeds and web scraping from our own institution. As a result, our data is a "mash-up" of a variety of semantic spaces, without and verifiable formal constraints. So we must empirically determine the domains and ranges of each class in our actual data. This type checking for domains and ranges is necessary for efficient graph processing.

Exhibit [13] is a popular interface, which provides a faceted filtering view on graph-based data. Exhibit's interface presents different facets for a given set of resources. To 'exhibit', we need to create the dataset that could be consumed by Exhibit, and the HTML page to present these data. Exhibit supports RDF data by harvesting RDF metadata locally and then translating into JSON format [4] using Babel web service [7].

The simplest realization of a faceted browser in Exhibit is by using resources as RDF subjects, facets as RDF predicates and restriction-values as RDF objects. The key user interaction is faceted filtering. For example, a list of persons might be filtered by selecting 'Director' in an 'Organizational Position' facet. This action will filter down the list of persons who hold the position of director in an organization. However, if another facet, such as 'Journal' is selected along with 'Organizational Position', an Exhibit based faceted browser still returns only the list of persons that satisfy both the criteria. In particular, it does not return the list of publications, which satisfied the 'Journal' criteria and are related to each of the Director's obtained through earlier selection criteria.

This loss of relationship information in Exhibit is due to its limited declarative language, which only allows for a single collection of items, in this case "persons", to be rendered. To render publications belonging to persons, Exhibit's query language needs to support graph navigation on RDF data. The minimalist graph navigation that is possible in Exhibit is if the resource "Publication" is a priori joined to the resource "Person". However, this is manual work and the underlying RDF schema rarely shows a direct join relationship--several intermediate nodes belonging to different semantic spaces such as BIBO, Event, may connect the Person and Publication resources, thus making it difficult to describe in Exhibit as to how to traverse the graph a priori.

B. Graph Operations for Faceted Browsing

In this subsection, we explain how graph browsing can be achieved for arbitrary RDF data obtained from scholarly systems. In the next subsection, we demonstrate how this browsing can be efficiently implemented in Exhibit to give a usable, relationship-preserving faceted browser.

Given a graph $G = (V,E)$, and source S and target T node, graph navigation implies determining the path that leads from S to T. In facet-based graph navigation, S represents the root node or primary information resource about which information is being sought. For instance, in scholarly domains, nodes of type "Person" form the root nodes---all information desired is in the context of root nodes. T represents the facet nodes, which the user wishes to select and constrain and determine what relevant root nodes are linked to the selected T nodes. Given the RDF schema graph shown in Figure 1(a) and its instantiation in Figure 1(b), central nodes of type "Person" represents the root nodes and other nodes represent possible facets and their facet values. There are two challenges in performing navigation on such graphs:

(1) The RDF schema is not known a priori from the RDF data but must be inferred; and

(2) It is not evident immediately which target facet nodes are more useful and more important than others in navigating the information space.

We address the first challenge by performing type checking on the RDF data to determine some structure of the underlying schema. We perform analytics on nodes to determine which nodes are more useful than others. We explain these operations and then describe our algorithms for graph navigation.

Type Checking helps to determine if there is a path between source and facet nodes using the domain and range constraints present in RDF schema. For example, if the user selects a

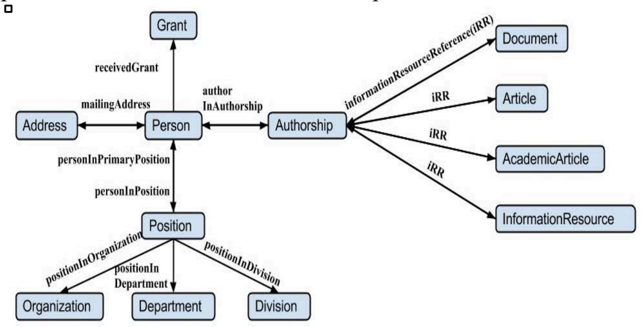


Figure 1(a): RDF schema showing types and their relationships

value V of a facet node with the name "Publication-Journal" then we know the type of this selection. However, the relationship of the publication item to the primary records of type Person is still unknown. A naive implementation would find "any Person reachable by *any* path in the graph from a Publication with Journal value V". This would be inefficient and may probably return the entire graph for most queries. Instead, we can use properties on source and target nodes to determine the most efficient path. For any selection, we can infer the type and the set of objects resulting from the selection. We can then search for a reliable path, i.e., a sequence of non-recurring edge labels that always contain exactly one item of each type in the set (this is only feasible to compute for edge labels in which both source and target nodes are typed).

Analytics help us determine relevant target nodes. One of the most predictable analytic is frequency of a potential facet node. A suitable facet occurs frequently inside the collection: the more distinct resources covered by the facet, the more useful it is in dividing the information space [6]. If a facet occurs infrequently, selecting a restriction value for that predicate would result in too many items being returned or affect a small subset of the resources. In [17], Oren, et. al. attempt to compute a metric for facet frequency, to aid in selecting facets that divide the information space in the most useful form. However, we have observed that it is simply not the facet frequency that is crucial in determining the relevance of a facet, but how many of those facets are reliably linked to the root node. In other words, it is the aggregate join frequency of the path from the facet to the root, than simply the frequency of the facet.

Our analysis shows that the most frequently occurring predicates are either unique descriptors, which are unsuitable for faceting, or internal data that is uninformative for the general user, such as types and numeric identifiers. On the other hand, we found that the predicates that are informative for the user are those in which relationship edges are strongly correlated with a specific type of node. We can identify more

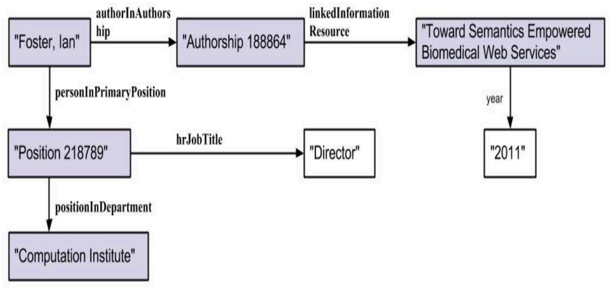


Figure 1(b): An instance of the RDF schema

useful facets, then, by computing predicate frequency not on individual types, but for each distinct relationship type.

Having determined the underlying schema to an extent and the target facet nodes, and reliable paths, we are ready to describe the graph navigation algorithms.

We define a facet F as <*name*, <*path*, *property*>> tuple, where *name* is the name of the facet and a resource class in the RDF schema, and <*path*, *property*> tuple represents the path from the root node to the set of *property* nodes whose type is literal. The literal values of *property* nodes are available for user selection in the faceted UI. An empty path implies the property on the root node directly. Thus the facet "Department" in the graph of Figure 1(a) can be expressed as:

<"Department",<personInPosition, positionInDepartment>,label>

Given this definition of a facet we define three kinds of operations on facets: select, exists and join, which are also described in abstract in [17].

Select operation selects nodes that have a direct restriction value. The basic selection allows for example in Figure 1(b) to select CI.

Exists operation selects all nodes when a property is true or false for those nodes. However, its exact value is not important.

Join operation selects resources based on the properties of the nodes that they are connected to. A typical join will select *source* resources and not *target* resources. Target resources are obtained by performing an **inverse-join**.

Path-Join By recursively composing join and select operations, we can traverse the path, and either lists all extant values, or applies a filter corresponding to the user's selection in the facet, and return the root node. *path-join* is a recursive function that traverses a path, and a set of target nodes, and returns a set of matching source nodes; i.e., the source nodes which can reach any target node via the path. The target nodes are the list of values selected by the user in the facet's UI. The path is the path between source and target, obtained a priori through type checking.

```
//P[n]: path of length $n$; $\mathcal{T}$: set of target nodes
$i = 1$
path-join(P[n], $\mathcal{T}$)
$l_i$ = P[$i$]
if ($i$ == n) then return $\mathcal{T}$
else join($l_i$, path-join(P[$i$+1], $\mathcal{T}$)
```

Inverse-Path Join Because our join operation returns the source node, and not the target. To reconstruct the graph fragments corresponding to the structure of the users query, we must construct one or more inverse joins, beginning with the root node and terminating with the property selection, for each constrained type, i.e., each unique path in F. We also suppress return of items of types with no constraints at all, so as to eliminate unnecessary nodes, but this is an optional design detail.

```
//P[n]: path of length $n$; $\mathcal{S}$: set of source nodes
$i = 1$
inverse-path-join(P[n], $\mathcal{S}$)
$l_i$ = P[$i$]
if ($i$ == n) then return $\mathcal{S}$
else join($l_i$, inverse-path-join(P[$i$+1], $\mathcal{S}$)
```

C. Implementation

We now describe how basic UI interactions such as facet values, facet selection and facet search are enabled through path joins and inverse path joins.

Facet Values builds a list of potential facet selections that are reachable from the root-type. For this, we traverse the path over the entire collection as follows:

*facet-values(*path, property*) = path-join(*path*, exists(*property*))*

Facet Selection If the user makes a selection, we can then use the same path traversal function to apply a suitable filter. An inverse-selection can similarly be defined:

facet-selection(path, property, selection) =
$\{v \in V \mid \exists\, s \in$ selection:
$v \in$ *path-join*(path, *exists*(*select*(property, s)) $\}$

inverse-selection(item, path, property, selection) =
$\{v \in V \mid \exists\, s \in$ selection: $v \in$ *intersect*(*inverse-path-join*(path, item), *select*(property, s)) $\}$

Facet Search To execute the search in its entirety, we intersect the results of each facet, which has a selection (a facet without selections is understood to be unconstrained). If no facets whatsoever are constrained, then the search returns all elements of the root type. Here, *n* is the number of facet values.

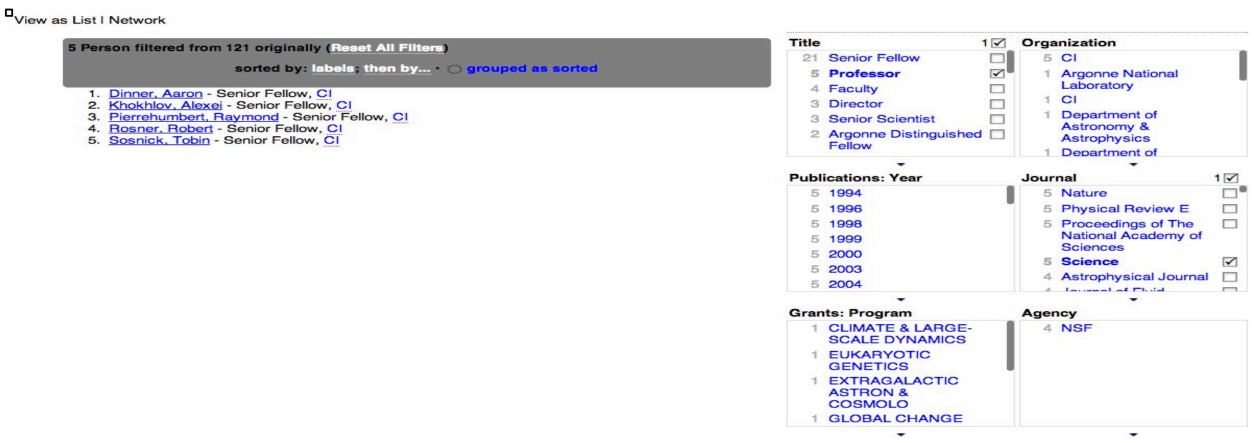
```
facet-search(F, root-type) =
 if (F = [])
        select("rdf:type", root-type)
 else
        intersect(
      facet-selection(path(F[n]), property(F[n]), select(F[n])),
     facet-search( (F[1] ... F[n - 1]), root-type) )
```

Inverse join can only be implemented after faceted search. This is done through an inverse query function that performs *inverse-selection* and *inverse-path-join*.

inverse-query(F, item, unique-path) =
$\{v \in V \mid \forall\, f \in F$: $v \in$ *inverse-selection*(source, path(f), property(f), selection(f)) *iff* |selection(f)| > 0 and path(f) = unique-path$\}$

Query Implementation We initiated the project by using Exhibit's existing query functionality and UI widgets. Each of the facet selection, facet values and facet search were implemented via a pair of Exhibit callback API's in JavaScript. The *onItemsChanged* event fires after a user has made a facet selection and the current search results change, and the *itemOnShow* callback fires anytime a single result item is displayed to the user. By catching both of these, and interrogating the defined path for each facet, we can reconstruct the information necessary for our *inverse-path-join*() function. The implementation consists of two major components: approximately 200 lines of Ruby to perform graph analytics and optimization, and about 400 lines of JavaScript to enhance Exhibit's functionality to display inverse path queries. Because Exhibit actually stores the unique label for each outgoing link in an object, we evaluate join's in much the same manner as a relational database, maintaining a list of results as we evaluate each edge of the path. Unlike a relational database, property lookup in JavaScript is a constant-time operation in Google Chrome. We can then generate a representation of the result set in any form; our implementation provides HTML and SVG output. Our prototype is available from the following URL http://profiles.uchicago.edu/facetedbrowser. In order to implement these interactions efficiently, we had to further make modifications to the RDF semantics and make some optimizations on internal nodes.

Semantics We found two major mismatches between RDF's semantic model and Exhibit's data model. RDF requires the rdf:about URI of an item to be globally unique and identifying, while the rdfs:label property is merely descriptive. However, Exhibit merges items with identical values of rdfs:label, whether they have unique URI's or not. Thus, for several item types, such as Position, we experienced label collisions--which resulted, in this case, in all identical titles, such as "Professor", to be merged, even though they belonged to different Departments. We corrected this behavior by generating globally unique numeric labels for each node.

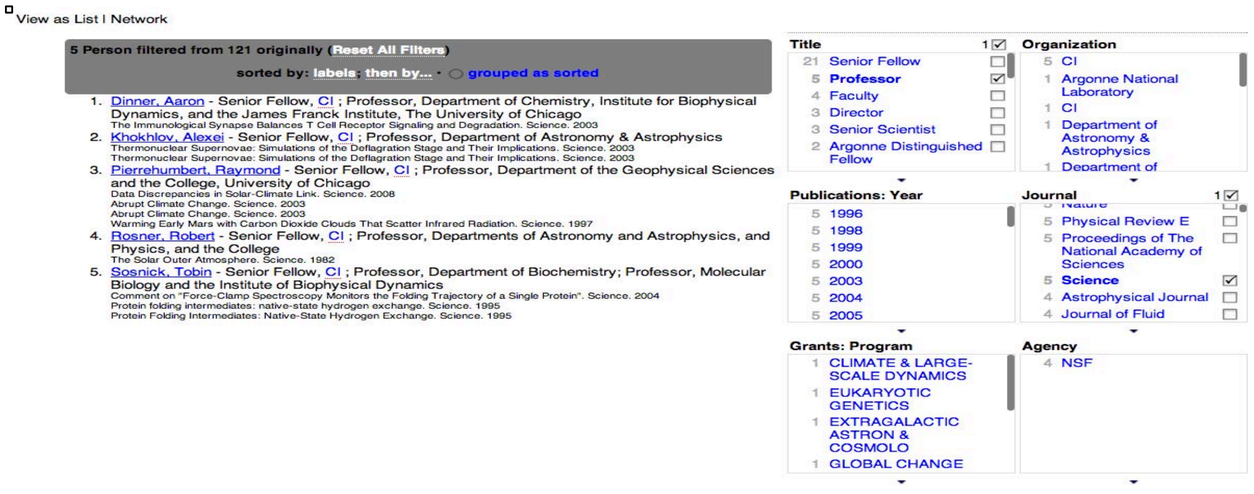


5 Person filtered from 121 originally (Reset All Filters)

sorted by: labels; then by... • grouped as sorted

1. Dinner, Aaron - Senior Fellow, CI
2. Khokhlov, Alexei - Senior Fellow, CI
3. Pierrehumbert, Raymond - Senior Fellow, CI
4. Rosner, Robert - Senior Fellow, CI
5. Sosnick, Tobin - Senior Fellow, CI

Figure 2(a): Lens before graph navigation. The selected facets of 'Title:Professor' and 'Journal:Science' retrieves only the list of persons that qualify the facet characteristics. It does not retrieve the associated publications or allow the user to classify publications further based on available facets such as "Publications:Year" or "Grants:Program".



5 Person filtered from 121 originally (Reset All Filters)

sorted by: labels; then by... • grouped as sorted

1. Dinner, Aaron - Senior Fellow, CI ; Professor, Department of Chemistry, Institute for Biophysical Dynamics, and the James Franck Institute, The University of Chicago
   The Immunological Synapse Balances T Cell Receptor Signaling and Degradation. Science. 2003
2. Khokhlov, Alexei - Senior Fellow, CI ; Professor, Department of Astronomy & Astrophysics
   Thermonuclear Supernovae: Simulations of the Deflagration Stage and Their Implications. Science. 2003
   Thermonuclear Supernovae: Simulations of the Deflagration Stage and Their Implications. Science. 2003
3. Pierrehumbert, Raymond - Senior Fellow, CI ; Professor, Department of the Geophysical Sciences and the College, University of Chicago
   Data Discrepancies in Solar-Climate Link. Science. 2008
   Abrupt Climate Change. Science. 2003
   Abrupt Climate Change. Science. 2003
   Warming Early Mars with Carbon Dioxide Clouds That Scatter Infrared Radiation. Science. 1997
4. Rosner, Robert - Senior Fellow, CI ; Professor, Departments of Astronomy and Astrophysics, and Physics, and the College
   The Solar Outer Atmosphere. Science. 1982
5. Sosnick, Tobin - Senior Fellow, CI ; Professor, Department of Biochemistry; Professor, Molecular Biology and the Institute of Biophysical Dynamics
   Comment on "Force-Clamp Spectroscopy Monitors the Folding Trajectory of a Single Protein". Science. 2004
   Protein folding intermediates: native-state hydrogen exchange. Science. 1995
   Protein Folding Intermediates: Native-State Hydrogen Exchange. Science. 1995

Figure 2(b): Lens after graph navigation shows the list of persons with the list of publications that match the selected facets

Optimization Beyond purely corrective and analytical processing, we also found several ways to optimize our graph for performance. Since it is very difficult to mechanically distinguish descriptive properties from genuinely uninformative nodes; however, our derived graph of type relationships allowed us to identify a large variety of node types that were never used in a query or a result description: these could be subdivided into two subsets, (1) those nodes that are present for ontological coherence but otherwise contain no data, such as Class and DatatypeProperty, which we simply omitted, and (2) nodes that only occurred in the intermediate steps of a multi-step path, such as Authorship, and could safely be consolidated into a single edge. These optimizations reduced the total number of nodes in our graph by approximately 50%.

D. LENS Faceted Browser

Figure 2(a) shows the default Lens faceted browser for scholarly data. The browser returns only the list of persons, which is not meaningful for a user even though both person and journal facet is selected. Figure 2(b) shows the faceted browser with graph navigation, and shows how by selecting person details (Professor) and publication (in "Science" journal) details, one can render the relationships of publications and positions of persons on the same UI. In addition users can browse the dataset by constraining one or several of these facets.

IV. Experiments

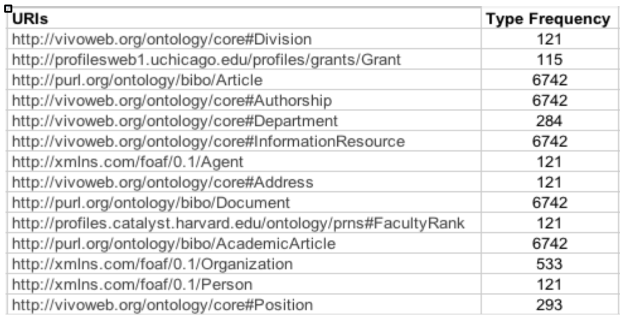
Profiles Research Networking Tool: Our experiments are conducted on publicly available data about researchers at the University of Chicago (UoC). This data currently includes their directory, publication, grant and patent information. The data contains approximately 6000 nodes, containing 121 Fellows, Senior Fellows, and Faculty of a research institute at UoC. The data is resident in the Profiles Catalyst system, which provides an RDF export function. Even though our dataset is moderate in size it is sufficient for performance characterization.

Experimental setup: We implemented this design in Simile Exhibit, client-side faceted browsing framework written in Javascript, due to the fact that it implemented the join, selection, and existential functions we need, while supporting a wide variety of data formats, including RDF. We used Exhibit's implementations of the facet-search and facet-values functions without modification; the inverse query mechanism was implemented as a Javascript function and injected via Exhibit's runtime callback API. We performed all testing on a 8-core 3GHx Intel Xeon Mac Pro with 16 GB of RAM. We used version 26.0.1410.65 of Google Chrome for all tests due to its strong JavaScript VM as well as the availability of performance profiling tools. We used a modified version of Exhibit 2.0 -- Exhibit 3.0 was deemed insufficiently stable for these modifications at the time the work began. Exhibit uses property lookup on a single large database object to perform joins, so we expected Chrome's constant-time property lookup to speed up performance substantially.

A. Data Types

Table 1 contains the total frequency of all the RDF types in our data set. Because of Profiles's subclassing, each node may have one or more types. We've omitted the standard RDF Ontology, DataType, and ObjectProperty classes--although vital for semantic clarity, they are never accessed during query execution.

Table 1: Frequency of Types

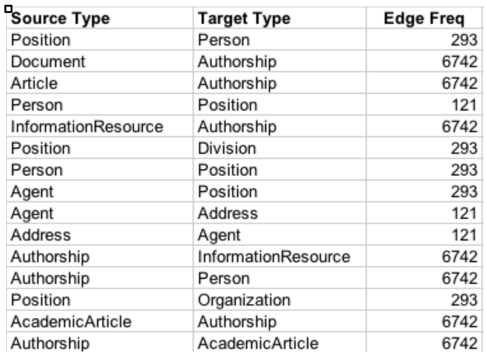


URIs	Type Frequency
http://vivoweb.org/ontology/core#Division	121
http://profilesweb1.uchicago.edu/profiles/grants/Grant	115
http://purl.org/ontology/bibo/Article	6742
http://vivoweb.org/ontology/core#Authorship	6742
http://vivoweb.org/ontology/core#Department	284
http://vivoweb.org/ontology/core#InformationResource	6742
http://xmlns.com/foaf/0.1/Agent	121
http://vivoweb.org/ontology/core#Address	121
http://purl.org/ontology/bibo/Document	6742
http://profiles.catalyst.harvard.edu/ontology/prns#FacultyRank	121
http://purl.org/ontology/bibo/AcademicArticle	6742
http://xmlns.com/foaf/0.1/Organization	533
http://xmlns.com/foaf/0.1/Person	121
http://vivoweb.org/ontology/core#Position	293

B. Edge-Type Correlations

Table 2 shows a snapshot of our measurements of the type relationships of each edge label in our graph--for clarity, we've omitted literal datatypes and the edges that point at them. The major complicating factor in this table is the duplicate types--since both the source and the target of a single edge can have more than one type.

Table 2: Type Relationship Analytics



Source Type	Target Type	Edge Freq
Position	Person	293
Document	Authorship	6742
Article	Authorship	6742
Person	Position	121
InformationResource	Authorship	6742
Position	Division	293
Person	Position	293
Agent	Position	293
Agent	Address	121
Address	Agent	121
Authorship	InformationResource	6742
Authorship	Person	6742
Position	Organization	293
AcademicArticle	Authorship	6742
Authorship	AcademicArticle	6742

C. Loading

Figure 3 demonstrates the start-up performance of Lens, not including graph analytics and optimization. Exhibit has RDF import capability built-in; however, it actually performs the conversion by a remote HTTP request to the Simile Babel service, with round-trip latency of a few hundred milliseconds--since this must be performed once per RDF file, it slowed our loading time down by a factor of ten. However, by consolidating the RDF files into a single JSON file in advance, we were able to avoid performing the conversion at load time altogether, taking our total load time from about 20 seconds to 2.

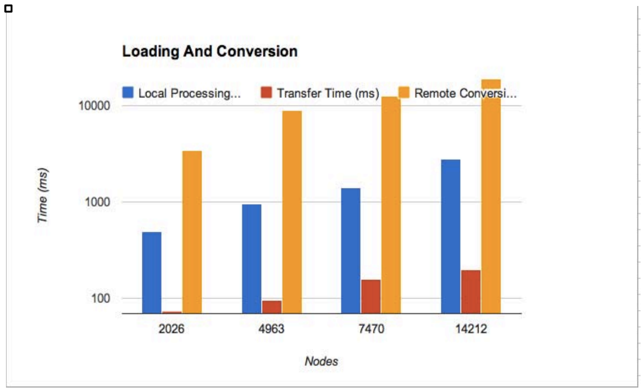


Figure 3: Load Times

D. Query Performance

We inserted timestamps at three critical places in our code -- (1) right before Lens begins processing the query, (2) at the beginning of the inverse-query function, called immediately after the main search routine returns, (3) at the end of the last rendering method after the inverse-query returns, when processing is completed--this allowed us to separate the execution time of Exhibit's Query mechanism from our Inverse-Query function.. We devised two sample queries to compare performance on: a) *selective query*, a sparse query with very few results, and b) *scan query*, a very large set of publications. We initially set out to compare performance on data sets of different size--however, our preliminary results showed a minuscule correlation between dataset size and performance--query time was effectively constant; we believe that the inverse-query was slowed somewhat because of the increased number of nodes it traversed while computing the inverse-joins. However, close examination of the performance profiler indicated that HTML rendering and garbage collection far outweighed the impact of doubled collection size.

Based on these results, we devised an additional set of measurements to study the correlation of performance to query complexity. Rather than measuring the total number of nodes in the database, in Figure 4 we inserted a counter in the join function to track the number of nodes traversed throughout the inverse-query method, then tested it on 14 queries of varying sizes. As the figure demonstrates, the relative performance of the query and inverse-query functions was somewhat unpredictable; however, a trend is clearly visible in the overall times. Close examination of the logs showed that Chrome's garbage collector was firing at unpredictable times, and taking up to 70 ms to retrieve up to 5MB of discarded object per pass--most likely a consequence of the large arrays generated and discarded at each step in our inverse-join function.

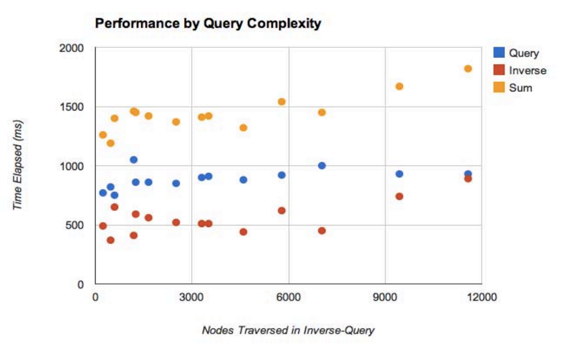


Figure 4: Query Performance

V. Conclusion

Universities and organizations are increasingly adopting researcher-networking platforms to inform policy decisions. Visualization is key to explore and absorb dense information. This paper demonstrated an open source faceted browser for researcher profiling systems that enables deep exploration and classification of relationships of researchers with their research products. Developed with data of our own institution, the browser is open source and applicable to all scholarly data. We also plan to demonstrate our system in the upcoming VIVO conference for research profiling systems.

References

[1] Bonura Jr, C. J., Situating political culture within the construction of geopolitical ontologies. *Rethinking Geopolitics*, 86, 1998.

[2] Conlon, M., Scholarly Networking Needs and Desires, In VIVO: A Semantic Approach to Scholarly Networking and Discovery: Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan Claypool, 2:1, 2012.

[3] Corson-Rikert, J., Mitchell, S., et. al., The VIVO Ontology, In VIVO: A Semantic Approach to Scholarly Networking and Discovery: Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan Claypool, 2:1, 2012.

[4] Crockford, D. The application/json media type for javascript object notation (JSON), 2006.

[5] Dadzie, A. S., & Rowe, M., Approaches to visualising linked data: A survey. *Semantic Web*, *2*(2), 89-124, 2011.

[6] W. Dakka, P. Ipeirotis, and K. Wood. Automatic construction of multifaceted browsing interfaces. In CIKM. 2005.

[7] Epperly, T. G., Kumfert, G., Dahlgren, T., Ebner, D., Leek, J., Prantl, A., & Kohn, S., High-performance language interoperability for scientific computing through Babel. *International Journal of High Performance Computing Applications*, *26*(3), 260-274, 2012.

[8] FOAF Vocabulary Specification, http://xmlns.com/foaf/spec/

[9] Gewin, V., Collaboration: Social networking seeks critical mass. *Nature*, *468*(7326), 993-994, 2010.

[10] Harvard Catalyst Profiles. http://profiles.catalyst.harvard.edu/.

[11] Hearst, M.. Clustering versus faceted categories for information exploration. Comm. of the ACM, 46(4), 2006.

[12] Heim, P., Ertl, T., & Ziegler, J., Facet graphs: Complex semantic querying made easy. In *The Semantic Web: Research and Applications*, 288-302. Springer Berlin Heidelberg, 2010.

[13] Huynh, D.F., Karger, D.R., Miller, R.C., Exhibit: Lightweight structured data publishing. In: Proc. of the 16th International Conference on World Wide Web, Banff, Canada, 737–746, 2007.

[14] Kobilarov, G., & Dickinson, I. Humboldt: Exploring linked data. *context*, *6*, 7, 2008.

[15] Maali, F., & Loutas, N., SPARQL 1.1 AND RDF Faceted Browsing, 2012.

[16] E. Oren and R. Delbru. ActiveRDF: Object-oriented RDF in Ruby. In Scripting for Semantic Web (ESWC), 2006.

[17] Oren, E., Delbru, R., & Decker, S.. Extending faceted navigation for RDF data. In *The Semantic Web-ISWC 2006*, 559-572, Springer Berlin Heidelberg, 2006.

[18] Raimond, Y., & Abdallah, S., *The event ontology*. Technical Report, 2007. http://motools. sourceforge. net/event, 2007.

[19] Research Profiling Systems. http://en.wikipedia.org/wiki/Comparison_of_Research_Networking_Tools_and_Research_Profiling_Systems.

[20] Rozell, E., Fox, P., Zheng, J., & Hendler, J. S2S Architecture and Faceted Browsing Applications. In *Proceedings of the 21st international conference companion on World Wide Web*, 413-416, 2012.

[21] Shotton, D. Cito: The citation typing ontology. *Journal of Biomedical Semantics*, *1*(Suppl 1), S6, 2010.

[22] SKOS Core Guide, http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102.

[23] Tenenbaum, J. D., *et. al*.. The Biomedical Resource Ontology (BRO) to enable resource discovery in clinical and translational research. *Journal of biomedical informatics*, *44*(1), 137-145, 2011.

[24] Weber, G. M., Barnett, W., Conlon, M., Eichmann, D., Kibbe, W., Falk-Krzesinski, H. and Kahlon, M. Direct2Experts: A pilot national network to demonstrate interoperability among research-networking platforms. *Journal of the American Medical Informatics Association*, *18* (Suppl 1), 157-160, 2011.