Thanks to Open Data initiatives the amount of data available on the Web is rapidly increasing. Unfortunately, most of these initiatives only publish raw tabular data, what makes its analysis and reuse very difficult. Linked Data principles allow for a more sophisticated approach by making explicit both the structure and semantics of the data. However, from the user experience viewpoint, published datasets continue to be monolithic files completely opaque or difficult to explore by making complex semantic queries. Our objective is to facilitate the user to grasp what kind of entities are in the dataset, how they are interrelated, which are their main properties and values, etc. Rhizomer is a data publishing tool whose interface provides a set of components borrowed from Information Architecture (IA) that facilitate getting an insight of the dataset at hand. Rhizomer automatically generates navigation menus and facets based on the kinds of things in the dataset and how they are described through metadata properties and values. This tool is currently being evaluated with end-users that discover a whole new perspective of the Web of Data.

Keywords: Semantic Web; Linked Data; Data Publishing; Information Architecture; Human-Computer Interaction; Usability.

1. Introduction
Data is being accumulated in the Web at an increasing pace, with relevant contributions from the Open Data initiatives from government agencies from countries like USA, UK or Spain. Unfortunately, the process is mainly of accumulation because most of these initiatives are limited to publishing raw data, which is very difficult to analyse and reuse, especially for users without a great knowledge of the data domain and about tools capable of processing it.

A way to facilitate reuse, though requiring a greater effort from the publisher, is to use Semantic Web and Linked Data principles when publishing data. They make the structure and semantics of the data explicit so data is more easily integrable and queryable.

However, the benefits of Linked Data are difficult to see from the user perspective, and thus demotivate its adoption. The problem continues to be that datasets are monolithic files completely opaque, or that can just be browsed using complex semantic queries. The objective is to make all this data more usable so users facing a dataset can easily grasp what kind of entities are in there, how they are interrelated, what are their main properties and values, etc.

This way, users can search, browse and analyse the data. This will increase the awareness of the data currently available in the Web and also facilitate the development of new and innovative applications on top of it. The overall outcome is that available data increases its impact and the society as a whole benefits more from data openness.

The simplest and more common approach to make a dataset accessible to a wider range of users is to publish it using Web tools. These tools usually provide HTML views of each resource in the dataset as a Web page showing all the properties and values describing them. These pages are linked based on the connections among resources and the user can follow HTML links to browse through them.

Unfortunately, this approach just provides a very narrow view of the dataset, just suitable for users with some a priori knowledge about the dataset, especially the URI identifier of the resource for which they want the details. On the contrary, for first comers or users looking for more than the description of a particular resource, there is no way to get at least an overview of the kind of resources in the dataset.

The objective, an the proposal of this paper, is to provide mechanism that facilitate getting a quick impression of what a dataset is about and its exploration, thus facilitating also the reuse of the dataset. More concretely, the proposal is to draw from the experience accumulated by the Human-Computer Interaction (HCI) and the Information Architecture (IA) disciplines.

From the former, interaction patterns specialised in data exploration are adapted, and from the later, we adapt existing IA components well known to Web users, as they are present in most web pages (navigation bars, navigation facets, sitemaps, breadcrumbs, etc.). Altogether, the proposal is implemented in the Rhizomer publishing tool also presented in this paper.

The rest of this paper is organised as follows. First, the related work is presented in Section 1.1 and the approach in Section 2. Then, Rhizomer is introduced in Section 3 and its Information Architecture components are detailed in Section 4. The tool has been evaluated with end-users and the results are presented in Section 5. Finally, conclusions and future plans are presented in Section 6.

1.1. Related Work

The simplest tools providing access to Linked Data are Semantic Web browsers, like Disco [5]. However, they do not provide publishing mechanisms, they just provide a more user-friendly view on previously published data. Moreover, they just provide detailed views for a particular subject or set of subjects, showing all the properties and values for them. At most, they can leverage the search mechanism of the underlying publishing tools, like SPARQL endpoints, so the user can perform keyword based queries to retrieve a set of subjects that can be browsed to get their descriptions.

Consequently, Semantic Web browsers do not provide support for more general views on the dataset being browsed. They provide just a narrow view of the resource corresponding to the current URI or the resources selected by a query, and at most of the steps followed so far using information architecture components like breadcrumbs\(^b\).

There are also tools that provide more informative interface components like facets, which allow getting a quick idea about the main properties and values used to describe a set of resources, but they are tied to particular datasets like the DBPedia Faceted Browser [6] or facet [7] and not easily adaptable to other datasets.

Explorator [8], a tool that makes it possible to browse a dataset available through a SPARQL endpoint, facilitates exploration by combining search, facets or operations on sets of resources. However, it does not provide any mechanism that provides a broader view on the dataset other than a list of all the classes or properties used.

Like Explorator, there are other tools that allow interactively building semantic queries without requiring knowledge of languages like SPARQL. In some case, they even facilitate the process by taking into account the ontologies that structure the dataset, like OWLPath [9]. However, they do not provide either a generic view of the dataset that facilitates knowing which words to use to build the query.

Other alternatives are specialised tools like Google Refine\(^c\), though it just accepts as input comma-separated values files and it is not available for semantic data. Other options are Content Management Systems (CMS) with semantic capabilities\(^d\). However, semantic CMSs are intended for content creation than for the importation and publication of existing one. Consequently, they do not provide features for facilitating access to the imported data by automatically generating an overview of the imported data. The same applies to semantic wikis, such as the semantic extension for MediaWiki called Semantic MediaWiki [10].

---

\(^b\) Breadcrumbs is a navigation aid used in user interfaces that allows users to keep track of their location within an information set and the path followed.

\(^c\) Google Refine, http://code.google.com/p/google-refine

\(^d\) Drupal 7, http://drupal.org/drupal-7.0
Finally, there are some tools that are specialised, to some extent, in publishing Linked Data. The aspect that we consider here is the kind of support they provide to users when accessing the dataset and try to get an idea about what it is about. Paget\(^6\) is a framework for building linked data applications. At the moment it is focussed on publishing data but the intention is that it is capable of managing updates too. It is resource-centric and data driven. From the semantic data describing subjects identified by their URI it generates different representations (RDF, HTML, JSON and Turtle) using content negotiation.

Pubby\(^7\) is similar to Paget. It builds a Linked Data frontend with dereferenceable URIs for the subjects in the served dataset and content negotiation. It also features a metadata extension that provides provenance information. However, in both cases, the frontends provided are just like those Semantic Web browsers have.

To conclude, it is also possible to consider platforms for semantic data storage and publishing like Talis Platform\(^8\) or OpenLink Virtuoso\(^9\). In both cases, in addition to the data stores, there is an HTML frontend for the datasets. However, like with previous tools, the support for broader awareness of the dataset structure is very limited.

The most significant contribution is in OpenLink Virtuoso, which provides a faceted view on a specific subset of the data, but in order to get it, it is necessary to provide an URI or some keywords for textual search. Consequently, the facets view is limited to the resources retrieved from a previous search and there is no way to previously get an overview of the kinds of resources in the dataset.

### 2. Approach

The amount of data in different forms, from CSVs to Linked Data, is rapidly increasing. The more sophisticated the way of publishing it, the more computers can help us dealing with it. However, at last, it is our responsibility to make sense of all this data in order to discover unforeseen patterns, make decisions, etc.

Linked Data is just one example, and the potential of this huge amount of data is enormous but it is not being fully realised as end-users find a great barrier when facing it. The barrier is that most of this data is available as data dumps or SPARQL\(^11\) endpoints.

For data dumps, it is really complicated to realise what data does one have at hand, what it refers to and what kind of terms are used. And it requires some experience in Semantic Web tools in order to do so. For SPARQL endpoints, the amount of work required for grasping the internalities of the data set might be reduced. However, a good knowledge of SPARQL is required in order to generate and understand a set of queries that allow realising how big the data set is, which are the main kinds of things, how are they interrelated, etc. And in any case, the results from the queries are not very usable, list of URIs and appearances counts.

\(^6\) http://code.google.com/p/paget/
\(^7\) http://www4.wiwiss.fu-berlin.de/pubby/
\(^8\) http://www.talis.com/platform
\(^9\) http://virtuoso.openlinksw.com
Consequently, computers need a powerful way to communicate with us when such amounts of data are into play. Humans process best great amounts of information using their fastest sense, sight [12]. It has the same bandwidth as a computer network, comparatively touch is about the speed of a USB key and hearing and smell the throughput of a hard disk. Finally, taste is like barely the throughput of a pocket calculator.

As our sense best prepared to deal with these amounts of data is sight, the way to establish this communication among computers and us seems to be quite dependent on disciplines like Visualisation, Interaction Patterns and Information Architecture.

From the Interaction Patterns point of view, we have started from the fundamental set of tasks for data analysis proposed by Shneiderman [13]. Below, there are these tasks associated to the Interaction Patterns that we propose to apply in Linked Data scenarios. This is just a preliminary proposal based on simple Interaction Patterns and future work now concentrates on exploring richer ones [14]:

- **Overview**: get the full picture of the data set at hand. At this stage we propose to apply the Global Navigation interaction pattern. It corresponds to the navigation bars users are used to see at the top or on the right of web sites.

- **Zoom & Filter**: zoom in on items of interest and filter out uninteresting items. Here the proposal is Faceted Navigation. Once we have zoomed by selecting the kind of things we are interested in from the navigation bar, facets for that set of things help us filtering out those we are not interested in.

- **Details**: after zooming and filtering the user arrives to the concrete resources of interest. At this point, the user can get the details for those resources, which in the case of Linked Data is to get the properties for the resources plus those properties pointing to them. This is related to the Details on Demand interaction pattern and can be implemented as a simple list of properties and values of the resource of interest or as a specific visualisation tailored to the kind of resource at hand, e.g. a map for geo-located resources.

Our proposal is to elaborate these interaction patterns in the context of Linked Data. We have chosen them because they are simple and very common so users are very comfortable with them. They are currently part of the “culture” about how information is structured in the Web so they have been deeply studied in Information Architecture (IA) domain [15].


The drawback of all these IA systems is that they are quite expensive to develop and maintain. Fortunately, when they are built on top of the structured data typical in the

---

Semantic Web and Linked Data, it is possible to automate most of the development and maintenance work.

We are currently testing all these interaction patterns in a Linked Data publishing tool called Rhizomer\(^1\). It features navigation bars automatically generated and maintained starting from the underlying thesaurus and ontologies, and how they are structured and instantiated. A similar approach is followed for generating facets for each kind of entity in the data set and also for prioritising them taking into account their “utility”, i.e. how well they perform when discriminating a set of resources.

3. Rhizomer

First of all, the basic functionality of Rhizomer is to make semantic data available as both Linked Data for machines and HTML views of that data for human users. To make this distinction, like other Linked Data publishing tools like Pubby, Rhizomer implements content negotiation taking into account the requested content type thus providing the requested data in the desired format.

However, this basic mechanism is extended following simple HTTP mechanisms and a REST approach [16] to make it also possible to manage the published data. Each resource is managed through the URI referencing where it is published, thus following a Resource Oriented Approach. The basic HTTP commands allow managing each resource: GET retrieves the semantic data associated with the resource in the requested format, PUT updates the data for the resource with the submitted one, POST creates a new resource with the submitted semantic description and DELETE removes the specified resource and the corresponding data.

The GET command is used to get the metadata for a particular resource but it is also used to send semantic queries based on the SPARQL query language [11]. Consequently, it is also possible to use this approach to retrieve metadata for sets of resources and to benefit from SPARQL Update, which might be more convenient when various resources are updated simultaneously.

All the previous HTTP commands (and the SPARQL queries) go through the server part of Rhizomer and reach at last the underlying data store, as shown in Fig. 1. Currently, Rhizomer integrates connectors for Jena, Sesame and Virtuoso. It is also possible to directly integrate any SPARQL endpoint, though in this case all the Rhizomer functionality is not available.

On the other hand, there are the client-side functionalities that constitute the focus of this paper. They have been developed with the aim of improving the usability of the published datasets. They are deployed in the user’s browser and implemented using JavaScript and asynchronous HTTP calls (AJAX [17]), thought most of the functionality is also available without JavaScript in order to improve accessibility [18].

The first feature for the client side is that all the RDF syntax of the semantic data is completely hidden. Like many Semantic Web browsers, Rhizomer provides an HTML

\(^{1}\) http://rhizomik.net/rhizomer/
view on the data that also facilitates the navigation across the data graph, as detailed in Section 4.1.

However, as it has been shown in the related work section, this approach is just appropriate for detailed views on resource descriptions and it does not contribute towards an awareness of the overall structure of a dataset. In order to provide such functionality, Rhizomer features a set of components inspired by those common in Information Architecture and combined following Shneiderman’s Data Interaction Patterns [13].

The added value is, in comparison with traditional Information Architecture approaches, that in our case these components are automatically generated and updated from the data, thesaurus and ontologies in the published dataset, as described in Section 4.2 and 4.3.

The previous mechanisms constitute the features offered by Rhizomer to publish data in a way that facilitates potential users realise what are the main kinds of entities in the dataset, what properties are more commonly used to describe them and the more common values. Due to this focus on the data, the intended users are mainly developers of applications driven by open data, who can more easily identify the potential value of the datasets published using Rhizomer.
Additional features of Rhizomer are specialised interaction services or semantic forms that facilitate semantic data edition [19] in the user interface. Examples of available interaction services are a map view for geolocated resources or a timeline for those with a date or start and end time-points. These services are automatically associated with the resources that satisfy the view requirements, e.g. latitude and longitude properties in the case of the map. More details about these additional features are available from [20,21].

4. Information Architecture Components

The more direct way to interact with a semantic dataset and benefit from its semantic structures is to use query forms. However, this is not the more convenient way of making users interact with a dataset, specially when it is the first time they face it. In this case, and even when they have interacted previously with the dataset, it supposes a quite big cognitive load to guess or try to remember the actual words to use to build the required queries to browse the dataset.

Users require mechanisms that show upfront the underlying structure of the dataset. This helps them detect which are the appropriate words to use in order to get the data they need but it also provides some guidelines when there is not a particular need and the users just want to explore the dataset. In all this cases, Shneiderman’s Data Interaction Patterns become specially relevant because the natural path to follow is overview, zoom & filter and finally to get to the details.

These tasks are not specific to Linked Data and have been considered for many different kinds of information systems. They became especially relevant when considering how to make users aware of the structure of the information they are facing through web pages. This is the focus of the Information Architecture (IA) discipline [4], which has accumulated a lot of experience about how to structure a web site to make it easier for users to have access to the information it contains.

Information Architecture identifies four kinds of systems:

- **Organisation systems**: they allow presenting information in different ways, following different schemas that make it possible to group or differentiate information using different criteria, like chronological, geographical or alphabetic order.
- **Navigation systems**: they help users move across the available information. For instance, there are navigation bars, facets or site maps.
- **Labelling systems**: they describe categories, options and links using terms that are meaningful for users. They are all around the information architecture of a site, even as part of other systems, e.g. navigation bars labels.
- **Search systems**: they allow users to search specific information chunks based on some sort of keywords. They also offer mechanisms to restrict the search space.

When looking at information architecture rooted from a semantic data perspective, it becomes quite natural to develop organisation systems that exploit the underlying thesaurus and ontologies. Each resource is thus described using terms from these thesaurus and ontologies, which can be organised in different ways and combined to
provide complementary perspectives on the resources, called facets. Our aim is to provide these facets as an interactive navigation system that is automatically generated from the available data and ontologies, as detailed in Section 4.3.

However, facets are just useful when the user has already focused on some particular kind of resources so it is possible to organise them using the properties they have in common and are shown as facets. A faceted view on the whole dataset becomes easily unusable because it would accumulate all the properties used to describe resources in the dataset, and most of them would just be used by a subset of them. Therefore, before this faceted navigation is performed, a more general view on the dataset is required. The best candidates for this are, following the IA tradition, global navigation systems.

Global navigation systems typically take the form of navigation bars that, in the case of web sites, are called global because they are present in all pages. They provide a view of the main kinds of things in the whole information system, so it is easier to move from any point of the information space to another following a limited number of steps.

Unfortunately, these IA systems are quite expensive to develop and maintain, especially if they deal with big and complex datasets like those typical in the Linked Data domain. Fortunately, these datasets are highly structured using ontologies and schemas that can be exploited to automate most of the development and maintenance work.

Section 4.2 details how global navigation menus are automatically created and maintained in Rhizomer starting from the underlying thesaurus and ontologies, and how they are structured and instantiated. A similar approach is taken for generating facets for each kind of entity in the dataset and also for prioritising them taking into account their “utility”, as detailed in Section 4.3.

Other IA components have also been considered. The labelling systems that usually accompany ontologies and thesaurus are exploited by the generic data-browsing component presented in Section 4.1. Other options are sitemaps, which can be generated following a similar approach to that for building navigation menus, or breadcrumbs, that keep track of the navigation steps followed by the user so far.

4.1. Generic Data Browsing

This component is aimed at the Details Data Interaction Pattern. It is based on a generic HTML view of the data retrieved when performing a GET operation on a subject. This is a typical feature of Semantic Web browsers and many Linked Data publishing tools that facilitates the interaction with users. However, in Rhizomer, this view introduces some particularities.

First of all, the HTML rendering for the data associated to a resource in Rhizomer includes both HTML and RDFa [22]. The rendering also tries to facilitate things for human users so to improve the usability of the HTML rendering all URIs are replaced with their labels, whenever they are available. Moreover, the language tag associated to the labels is taken into account so it is easy to implement a multilanguage interface.

In order to generate HTML from RDF, fragments are serialised as RDF/XML and transformed using an XSLT. The XSL transformation, which is part of the Rhizomer
platform, guarantees consistent results whenever the input RDF/XML has been generated from fragments based on the Concise Bounded Descriptions [23] approach.

XSL transformation from RDF/XML to HTML is invoked from the client using AJAX, which is also responsible for sending the SPARQL queries and making the whole process go smoothly behind the scenes, making the user experience even more comfortable. Finally, the AJAX part of Rhizomer at the client also keeps track of the browsing steps so it is possible to use the "back" and "forward" browser buttons. Moreover, the browsing steps are cached at the browser in order to improve responsiveness.

4.2. Navigation Menus

As previously introduced, navigation menus in the context of websites let users navigate through different sections and pages of the site. Traditionally, techniques like Card Sorting [24] are used to develop the navigation menus of web sites. This technique requires a lot of time and effort from developers and most of this effort is lost as soon as the structure of the menu is established and fixed in a menu that becomes something static. If new kinds of items are introduced or a part of the content becomes more relevant, the Card Sorting should be repeated, at least in part.

Consequently, the added value in the case of web sites build on top of semantic data is to automate as much as possible the process of generation and maintenance of the navigation menus. This is possible because semantic data is structured by thesaurus and ontologies, that hierarchically organise the kinds of things described in the dataset. They specify all the classes or categories but also which subjects belong to each class or category.

Consequently, if there are few members of some kind, or there are not at all, it should be less relevant in the menu bar. On the contrary, those that do have a lot of members should be shown prominently in the menu bar. To do this, we obtain the hierarchy of all kinds of entities and apply inference rules to get their members. Then, the hierarchy is flattened to the amount of levels required because this component can generate both global and local menus, i.e. a menu for the whole dataset or for a subset of it. The site administrator can also configure some parameters:

• The number of levels in the hierarchical menu.
• The number of items in each level of the menu.
• The order of items: alphabetical or by number of members.
• A list of classes or categories to omit.

According to these parameters, this component generates the menu applying a recursive algorithm that mainly performs two operations:

• Split the categories or classes with a large amount of members in their subcategories or subclasses.
• Group those with few members into a supercategory or superclass.

This approach makes it possible to show the user the navigation bar that best fits the data in the dataset at that particular moment. For instance, if the dataset changes from containing mainly data about projects to mainly about publications, the menu would
change accordingly to show more prominently the part of the dataset structure about publications.

On the other hand, one possible drawback of this approach, as it has been pointed by some usability expert evaluations [18], is that users might find it very disturbing that the navigation menus change from visit to visit due to changes in the underlying data. This is an inconvenient effect of navigation menus dynamism, as users see them as a static part of the site and, as they get used to them, they rely on them as a handful guide to the site.

In any case, our experiments show that these changes are only systematic if there is very few data. Under those circumstances, the navigation menu undergoes changes quite often when adding new resources. However, as more resources are introduced, changes in the navigation menu tend to be minimal and as soon as the amount of data is statistically significant to keep the natural tendency in the dataset evolution, the changes in the menu bar are practically inexistente or not significant from the point of view of the user as they only affect to particular options in the submenus that are added or removed in the context of more general options in the menu, that keep users in the track to the information they need.

4.3. Facets

Most times, when users face a semantic dataset, they do not know exactly what they are looking for or they do not know how to name it. In these cases, exploratory search is a strategy that allows users to refine their search by successive iterations. An exploratory interface such as faceted browsing allows users to find information without a priori knowledge of its schema.

Navigation menus make the user aware of the hierarchical structure of a dataset but, once users choose the class of things they are interested in, they face the barrier of not knowing how they are described. In other words, what are the main properties that describe them, which ones are the more relevant for that particular kind of things, the range of values they have in that particular case, etc.

Faceted navigation is an exploratory technique for navigating a collection of elements in multiple ways, rather than a single and pre-determined order. Facet browser interfaces provide a user-friendly way to navigate through a wide range of data collections. For example, a collection of books can be classified using an author facet, a subject facet, a date facet, etc.

Traditional facet browsers relied on manual identification of the facets and on previous knowledge of the target domain. When dealing with semantic data, it is possible to automate this process and a semantic faceted browser should be able to handle any RDF dataset without any configuration. Since Linked Data facilitates integrating data from different sources, we can’t assume a single fixed schema for all data. Consequently, a Linked Data faceted browser should be scalable and generic, not depending on a particular dataset structure.

In traditional Web, facet browsers are developed to navigate through homogeneous data and facets are fixed. This conflicts with Linked Data, where data is too diverse to use
a single set of facets: facets that make sense for one type of entity could be inappropriate for other types. Moreover, when new data is added the system should be able to add new facets at run time.

One of the most important aspects of a facet browser is that, when constraining the dataset, all properties and values that would lead to an empty set of results need to be automatically removed from the interface, protecting the user against dead ends.

To build the facets, and to keep them updated, what Rhizomer does is to perform queries for each class in the dataset ontologies that retrieve all the properties their members have, the different values each property has and the cardinality for each value, i.e. how many times that property for that class takes that value.

Facets are calculated and stored in a data structure. They are then updated whenever the dataset is edited through Rhizomer. They are also updated, but just a local copy associated to a user session, when the user starts browsing and selecting values for different facets. In this case, the set of instances used for facets generation is constrained by the choices made so far and the facets are recalculated for that constrained set of results obtained so far. Those facets that are no longer relevant, i.e. no result uses them, are removed from the facets set.

When a dataset is very large and heterogeneous, the number of facets will also be very large. Therefore, an automated method is needed to choose which facets are the most useful for the user. We need to find those facets that best represent the dataset and those that are best to navigate the dataset. Choosing the right facets is very important, a suitable facet should allow efficient navigation through the dataset and be representative for those objects.

To measure the quality of a facet, and therefore showing it more prominently to the user, we use three metrics:

- **Property frequency**: we are interested in those properties that occur frequently for the subjects being browsed. The more described by a property, the more useful it is in dividing the information space. If the property is not frequent it will only affect a small subset of the collection. We compute the property frequency (1) as the number of resources for which the property \( p \) is defined. We normalise this value dividing it by the total number of subjects.

  \[
  \text{freq}(p) = \frac{n_r(p)}{n_r} \tag{1}
  \]

- **Property balance**: a facet helps the user better discriminate the set of items being browsed when it takes a well-balanced range of values for the facet property. On the contrary, a facet whose property takes always or mainly a particular value is less useful. The same happens if each item has a different value for the facet property. Consequently, we will favour facets that show behaviours in between these worst cases. To compute the property balance we use the Shannon’s entropy formula (2).

  \[
  H(S) = -\sum_{i=1}^{n} p(v_i) \log_{10} p(v_i) \tag{2}
  \]

- **Value cardinality**: a suitable property should have a small amount of values to choose from. If there are too many choices it is difficult to display all the options
and it might confuse the user. We compute the value cardinality as the number of different values for a property in a given set of subjects. This metric is normalized using a function based on the Gaussian density that can be regulated through the $\mu$ and $\sigma$ parameters to the top and bottom values of the range of different values we are interested in \((3)\). This range is still to be fixed experimentally but existing work recommends values around $\mu=2$ and $\sigma=20$ \([25]\). 

\[
\text{card}(p) = \begin{cases} 
0 & \text{if } n_0(p) \leq 1 \\
\frac{1}{e^{-\frac{(n_0(p) - \mu)^2}{2\sigma^2}}} & \text{otherwise}
\end{cases}
\] \hspace{1cm} (3)

However, as users test detailed in Section 5 have shown, the number of different values for a property will be also too big for showing it to the user. This is especially true when dealing with large datasets, something common in Linked Data. Consequently, we decided to show just the five most used values for each property and we added a text search box, which suggests possible matches. There is also the possibility to see the rest of values and choose from them. If the user wants to choose from the rest of values there is also a link, which can be used to see more values.

The previous three metrics are combined using a weighted function that produces a unique usefulness value for each facet. Currently, the function is equally balanced among the three metrics but we are conducting tests to determine the best arrangement of weights for most Linked Data datasets.

5. Evaluation

Rhizomer, though it is still at a prototype stage, has already been tested with end-users in order to evaluate its functionality and usability. The goal of the test conducted so far was to preliminary evaluation of the Information Architecture components, if they are understood and if they improve the awareness of the structure of a particular dataset by improving user performance when looking for a specific piece of information.

We worked in the UsабилиLAB\(^a\) facilities at Universitat de Lleida. To register sessions we used Morae Recorder and Morae Observer to analyse user test data. For the usability test metrics we chose effectiveness (percentage of tasks completed, workload) and efficiency (time to complete a task).

We have used a real test dataset called the Linked Movie Data Base (LinkedMDB)\(^b\). We chose the movies domain because it is well known for most people and quite appealing. LinkedMDB is generated from the Internet Movie Database\(^c\) (IMDB), data is extracted from the IMDB site, represented as Linked Data and enriched with an ontology.

---


\(c\) IMDB database http://www.imdb.com/
Therefore, we considered interesting to compare the evaluation results with those for IMDB and thus be able to test if the same data as Linked Data can become more usable than from the original web site. Consequently, we established one scenario with one task to be performed with IMDB and another one with one task for Rhizomer.

Six participants were selected, with a unique profile characterized by good knowledge of information technology, limited knowledge about Semantic Web technologies and interest in movies. The reason we chose this profile is because Rhizomer is intended for users interested in working with data, used to information technologies but not necessarily aware of Semantic Web and Linked Data.

Users signed a confidentiality document and received a post-task and a post-test questionnaire. The test facilitator proposed users the two scenarios and tasks, but not necessarily in the same order:

- **Task A**: “Find three films where Woody Allen is director and actor at the same time” using IMdB.
- **Task B**: “Find three films where Clint Eastwood is director and actor at the same time” using Rhizomer.

Fig. 2 shows a screenshot showing how Task A can be resolved using the Rhizomer interface. The main findings from the test are:

- Only one participant was able to complete the first task without assistance.
- 100% of participants needed in at least one occasion the guidance of the facilitator to successfully complete task B.
- In task B, 100% of the participants began the navigation from actors instead than from movies. This was the reason why users required assistance but as soon as they realized they were able to start it from movies, the tasks was easily solved.
- Both tasks required a lot of interaction steps to be completed. The first task was completed with an average of 8.67 steps and 9.83 in the second task.
- Efficiency based on the degree of completeness is relatively low for both tasks. 32% on average in the first task, and 54% in the second task. Only one participant approached 100%, giving an efficiency of 95% for the second task.
- 83% of participants completed the second task in less time than the first. Just one user completed the first task in less time than the second.

From test results and their analysis, we have elaborated these proposals for Rhizomer improvement:

- Navigation must be better contextualised. The interface should provide more mechanisms to inform the user where she/he is, where she/he can go and where she/he has been. For that, the proposal is to integrate some kind of breadcrumbs that summarise the navigation steps though navigation menus and facets.
- All items should be labelled so URIs or URI fragments are not shown to the user. For resources that have no label, this would required a tool that detects unlabelled items and assists during the labelling process.
- To add a results pagination mechanism that makes it clear the total number of results and allows browsing them.
- To improve how facets are presented to the user, especially when there are a lot of values. For that, the proposal is to use values indexes or graphical representations for numeric values, e.g. histograms.
• To mark the external links, using some sort of image, text or colour, so in case the user leaves the application she/he is aware in advance.

• To hide some advanced features, like data edition, that are not useful for non-advanced users. Different user profiles will be defined and we will determine which options are displayed to each user profile.

• One of the main issues detected is that the user interaction is currently too constrained by how the underlying data is structured. In this test, the result was that the task was performed differently from it was expected and this confused all users. They were looking for movies where actor and director were the same but, instead of initiating their interaction from the “Movies” menu option, all users started from “Actor”. From there, as the underlying data just modelled actors per film but not the reverse, it was impossible to filter those films where the same person was the director. The easy way was to look for movies and to filter by director and actor using the corresponding facets, as the underlying data has these two properties associated to every film. The impression is that users tend to think first about persons and consider films a secondary entity. The idea here is to exploit the possibilities of the underlying conceptual model and derive implicit properties, for instance reverse properties, in order to provide users with alternative paths. In this particular case, there will be reverse properties from actors to films. Moreover, it will be necessary then to focus on the set of films for an actor and filter it by director.

6. Conclusions and Future Work

As it has been shown, Rhizomer implements a set of features that make it possible to deploy it on top of any dataset based on Semantic Web technologies and Linked Data principles and publish it, while facilitating user awareness of what is in there. This awareness is accomplished by components borrowed from the Information Architecture discipline.
Currently, these components are navigation bars, which show the main kinds of items in the dataset, and facets, which show the more significant properties for different kinds of items and their values. And what is more important, these components are automatically generated and maintained. Moreover, it is possible to deploy Semantic Web services that provide specialised ways to interact with the data and analyse it.

Our preliminary tests with users show that Rhizomer facilitates publishing and browsing a dataset, like many other similar tools, but also allows that users realise what is the value of the dataset in the context of their particular needs. It has also shown its scalability from small datasets like the one for the Rhizomik initiative\(^9\) to really big ones like LinkedMDB\(^8\). Both datasets can be accessed online through Rhizomer at the provided URIs.

The future work focuses now on implementing the improvements detected during the user tests and detailed in Section 5 together with the proposed way to face them. Here, in addition to included facets for inverse properties when they are not explicit in the data, the objective is to flexibilise navigation making it possible for users to move from a set of resources they have filtered using facets, to a related set connected through a particular property and continue filtering using the facets for that related set.

This way, it will be possible to complete the tasks proposed in Section 5 independently from the starting point chosen by the author, i.e. films, actors or directors. If the user starts from actors, he can restrict by name to a particular one and then move to the set of films related to the actor through the corresponding property. Once there, it is possible to use the “directed by” facet to get the desired result.

The mechanism is quite obvious in this case because we are starting from just a particular actor and it is possible to solve the task performing “classical” faceted navigation from one particular kind of resources, in this case films. However, this capability of moving from one set of resources to another one, which is called pivot navigation [26], shows its whole potential when considering sets of things with particular properties related to other sets of things further filtered using their own properties.

This way it is possible to build really complex queries just by navigating the dataset using the navigation bars and facets. For instance, consider the task to retrieve all the 2010 films directed by Chinese that feature actors born in the USA. In this case, it is not possible to complete the task just by faceting on films, actors or directors. It is necessary to start faceting on one of then and then move to the other kinds of resources, following the appropriate properties and “inherit” the restrictions done so far, and continue faceting them using their particular properties.

We are currently implementing this feature so it will be possible, for instance, to choose actors from the navigation menu, restrict them using the nationality facet to just Chinese, pivot to the films directed by Chinese directors, use the date facet to restrict them to those for 2010 and finally pivot to the set of actors participating in 2010 films.

\(^8\) http://rhizomik.net
\(^9\) http://rhizomik.net/linkedmdb
directed by Chinese directors. There, the last step is to use the nationality facet to get just the USA actors so the set of films they are connected to at this stage is the desired set.

To our knowledge, the only tool capable of providing a faceted view and pivoting on facets is Parallax\textsuperscript{1}, though it is build on top of the Freebase dataset and proprietary technologies. There is an attempt called SParallax\textsuperscript{2} to make it capable to work on top of SPARQL endpoints. However, in both cases, they lack a mechanism to provide an overview of the dataset and some performance issues, specially in the case of SParallax.

Other areas of future work are related with more generic aspects of facets. One objective is to generate facets customised to the kind of values being managed, i.e. numerical values, alphabetical values, dates, geographical points, etc. Another objective is to improve the functionality of this component by adding other operators for restricting the results: multiple value selection, inverse selection, existential selection, join selection, selection between ranges, etc.

We are also experimenting with different ways of combining the three metrics used to rank the facets. And, it would be also interesting to consider not only the statistical value of each facet but also their descriptive value.

To conclude, the evaluation with end-users has been particularly fruitful. Though the results for Rhizomer have been better than those for the non-semantic version of the data and user interfaces, the main contribution of the test has been qualitative, i.e. the amount of improvements spotted by the way real users interact with the tool.

Acknowledgments

The work described in this paper has been partially supported by Spanish Ministry of Science and Innovation through the Open Platform for Multichannel Content Distribution Management (OMediaDis) research project (TIN2008-06228).

References


\textsuperscript{1} http://www.freebase.com/labs/parallax/
\textsuperscript{2} http://sparallax.deri.ie