Harvesting Pertinent Resources from Linked Open Data

Atif Latif[1], Muhammad Tanvir Afzal[2] , Anwar Us Saeed[1], Patrick Hoefler[1], Klaus Tochtermann[1]
[1]Institute for Knowledge Management, Know-Center
Graz University of Technology, Austria
{atif.latif, anwar.ussaeed}@student.TUGraz.at, phoefler@know-center.at
klaus.tochtermann@TUGraz.at

[2]Institute for Information Systems and Computer Media
Graz University of Technology, Austria
mafzal@iicm.edu

Journal of Digital Information Management

ABSTRACT: *Linked Open Data (LOD) is becoming an essential part of the Semantic Web. Although LOD has amassed large quantities of structured data from diverse, openly available data sources, there is still a lack of user-friendly interfaces and mechanisms for exploring this huge resource. In this paper, we describe a methodology for harvesting relevant information from the gigantic LOD cloud. The methodology is based on combination of information: identification, extraction, integration and presentation. Relevant information is identified by using a set of heuristics. The identified information resource is extracted by employing an intelligent URI discovery technique. The extracted information is further integrated with the help of a Concept Aggregation Framework. Then the information is presented to end users in logical informational aspects. Thereby, the proposed system is capable of hiding complex underlying semantic mechanics from end users and reducing the users' cognitive load in locating relevant information. In this paper, we describe the methodology and its implementation in the CAF-SIAL system, and compare it with the state of the art.*

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval - Information filtering.

General Terms: Semantic web, Open data, Information extraction,

Keywords: Linked Open Data (LOD), Semantic web, Information retrieval, Information presentation, CAF-SIAL, URI, User Interfaces.

Received: 11 September 2009; Revised 18 November 2009; Accepted 29 November 2009

1. Introduction

The World Wide Web can be seen as a huge repository of networked resources. Due to its exponential growth, it is a challenging task for search engines to locate meaningful pieces of information from heavily redundant and unstructured resources. The semantic paradigm of information processing suggests a solution to the above problem: Semantic resources are structured, and related semantic metadata can be used to query and search the required piece of information in a very precise manner. On the other hand, the bulk of the data currently residing on the Web is unstructured or semi-structured at best.

Therefore, the W3C launched the Linking Open Data[1] (LOD) movement, a community effort that motivates people to publish their information in a structured way (RDF)[2]. LOD not only "semantifies" different kinds of open data sets, but it also provides a framework for interlinking. This framework is based on the rules described by Tim Berners-Lee [Berner-Lee 2006]. As of May 2009, the LOD cloud consists of over 4.7 billion RDF triples, which are interlinked by around 142 million RDF/OWL links [Auer et al 2009]. Although LOD has created huge volumes of data and has attracted the attention of many researchers, it still lacks broad recognition, especially in commercial domains. This is, amongst other reasons, because of complex semantic search and end user applications [Latif et al 2009].

In the absence of official standards, DBpedia[3] and Yago[4], amongst others, are considered de facto standards for classification. DBpedia is also a central interlinking hub for Linked Data. Facts about specific resources, extracted from the infoboxes of Wikipedia, are structured in the form of properties as defined by DBpedia's ontology [Auer et al 2007]. This ontology is associated with Yago's classification to identify the type (person, place, organization, etc.) of the resource. For instance, a query about Arnold Schwarzenegger returns about 260 distinct properties, encapsulating nearly 900 triples in the raw RDF form. Such semantic data is not (easily) graspable by end users. Representing this bulk of structured information in a simple and concise way is still a challenge.

Recently, a few applications have emerged, which provide user interfaces to explore LOD datasets [Berners-Lee et al 2006][Kobilarov and Dickinson 2008]. These applications use SPARQL endpoints to query LOD with Subject-Predicate-Object (SPO) logic. SPO logic represents a triple, which is a building block of RDF. A triple establishes a relationship between two resource types. One resource is called subject and the other one object. The relationship between subject and object is called predicate. For example, Arnold Schwarzenegger (subject) is governor of (predicate) California (object). Now, in order to exploit LOD resources using SPARQL endpoint with interfaces of recent applications, users have to understand the underlying semantic structures (triples, ontologies, properties). The same gap between semantic search and end user applications has also been identified by [Chakrabarti 2004].

Each resource that is described by Linked Data can be uniquely identified by its URI [Sauermann et al 2008]. Relations and attributes of this URI can then be queried by use of SPARQL. However, regular Web users have never even heard of URIs or SPARQL. Therefore, when non-expert users interact with

[1]http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[2]http://www.w3.org/RDF/
[3]http://dbpedia.org/
[4]http://www.mpi-inf.mpg.de/yago-naga/yago/

the Semantic Web, the first step is to translate their queries into URIs. For example, when a user wants to know something about "Arnold Schwar-zenegger", it is necessary to find a URI that represents this person in the Semantic Web e.g. http://dbpedia.org/resource/Arnold_Schwarzenegger

To overcome the URI discovery, an intelligent Keyword–URI mapping technique has been introduced. Users don't need to remember a URI anymore to find resources from LOD. Users enter a keyword, and the system discovers the most relevant resources from LOD. The system employs a two-layered approach. In the first layer, users are automatically suggested with resources matching the entered keywords from a locally maintained LOD resource triple store. In the second layer, the user keyword is matched with metadata of resources indexed by a Semantic Web search engine (Sindice). The exploratory evaluations have shown that the system can reduce user's cognitive load in finding required URIs.

When the system has identified a correct resource URI, then it proactively picks up a set of properties related to the selected resource. The most relevant set of properties is grouped together by using the Concept Aggregation Framework. This property set is pre-computed for each resource type. This approach conceptualizes the most relevant information of a resource in an easily perceivable construct.

We also propose a two-step keyword search process in order to hide the underlying SPO logic. In the first step, users search for a keyword, and the system auto-suggests related entries to exactly specify the subject. Then, information related to that subject is structured using the aggregation framework. Furthermore, to avoid searching a specific property (predicate) of the selected subject by its name, a keyword based 'search within' facility is provided where the specified keyword is mapped to a certain property or set of properties.

The remainder of this paper is structured as follows: Section 2 discusses the state of the art and related work. Section 3 presents an intelligent Keyword-URI technique. Section 4 elaborates on the Concept Aggregation Framework. Section 5 describes the system architecture. Section 6 explains the overall use of the system with the help of a use case . Section 7 presents system evaluation and discussions. Conclusions and future work are discussed in section 8.

2. Related Work

2.1 URI Retrieval State of the Art

2.1.1 DBpedia

DBpedia is currently one of the most promising knowledge bases, having a complete ontology along with Yago [Suchanek et al 2007] classification. It currently describes more than 2.6 million things, including at least 213,000 persons, 328,000 places, 57,000 music albums, 36,000 films, and 20,000 companies [Auer et al 2009]. The knowledge base consists of 274 million pieces of information (RDF triples). The openly available RDF dumps make DBpedia an interesting subject of study. There has been valuable work done on studying the reliability of Wikipedia URI's[Hepp et al 2008] that are being used by DBpedia. This study suggests that the meaning of a URI stays stable approximately 93% of the time. Its heavy interlinking within the LOD cloud makes it a perfect resource to search URIs. For our current prototype, we concentrated on the part of DBpedia that encompasses data about people.

2.1.2 Sindice

Sindice [Tummarello et al 2007] provides indexing and search services for RDF documents. Its public API allows forming a query with triple patterns that the requested RDF documents should contain. Sindice results very often need to be analyzed and refined before they can be directly used for a particular use case. Similar kinds of services are provided by semantic search engines like Falcon [Cheng et al 2007] or Swoogle [Ding et al 2004]. We used Sindice in our work due its larger indexing pool and the ease provided in use of public API.

2.1.3 SameAs

SameAs[5] from RKB explorer provides a service to find equivalent URIs. It thereby makes it easier to find related data about a given resource from different sources.

2.2 Linked Data Consumption

2.2.1 Linked Data Browsers

The current state of the art with respect to the consumption of Linked Open Data for end users is RDF browsers [Berners-Lee et al 2006][Kobilarov and Dickinson 2008]. Some tools such as Tabulator [Berners-Lee et al 2006], Disco[6], Zitgist data viewer[7], Marbles[8], Object Viewer[9] and Open link RDF Browser[10] can explore the Semantic Web directly. All these tools have implemented a similar exploration strategy, allowing the user to visualize an RDF sub-graph in a tabular fashion. The sub-graph is obtained by dereferencing [Berrueta and Phipps 2009][Chimezie 2009] an URI, and each tool uses a distinct approach for this purpose. These tools provide useful navigational interfaces for the end users, but due to the abundance of data about a concept and the lack of filtering mechanisms, navigation becomes laborious and bothersome. In these applications, it is a tough task for a user to sort out important pieces of information without having the knowledge of underlying ontologies and basic RDF facts. Keeping in mind these issues, we suggest a keyword search mechanism to reduce the cognitive load of the users.

2.2.2 SPARQL Query Tool

Regarding the problem of searching and filtering in the Web of Data, a number of approaches and tools exist. One approach is to query a SPARQL endpoint that returns a set of RDF resources. There are a few tools that allow to explore a SPARQL Endpoint. NITELIGHT [Russell et al 2008], iSparql [Kiefer et al 2007], Explorator [Samur and Daniel 2009] are Visual Query Systems (VQS) [Catarci et al 1997] allow visual construction of SPARQL queries and differ mainly in the visual notation employed. However, in order to use these tools, the user must have comprehensive knowledge of the underlying RDF schemata and the semantic query languages (e.g. SPARQL). In summary, current tools allow users to manipulate the raw RDF data and do not provide user-friendly interfaces.

2.2.3 Faceted Search Tools

Contrary to VQS applications, Freebase Parallax [Hildebrand et al 2006], the winner of Semantic Web challenge 2006, is based on the idea of faceted search. Freebase Parallax is a browser for exploring and presenting the structured data in a centralized infrastructure. Similar faceted search application YARS2 [Harth et al 2006] explores distributed datasets using SPO constructs. To the best of the authors' knowledge, the approach presented in this paper is the first one that uses arbitrary data accessible

[5] http://www.sameas.org
[6] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/
[7] http://dataviewer.zitgist.com/
[8] http://beckr.org/marbles
[9] http://objectviewer.semwebcentral.org/
[10] http://demo.openlinksw.com/rdfbrowser/index.html

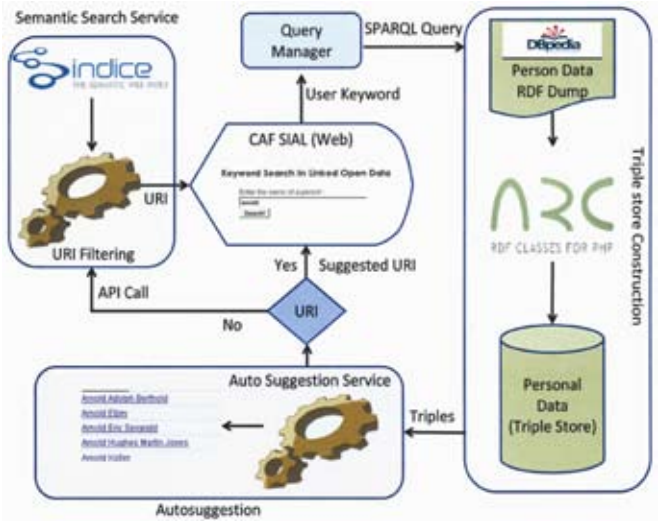

Figure 1. Keyword-URI mapping design

via SPARQL and aggregates important facts on the basis of informational aspects.

3. Keyword-URI Mapping Technique

The design of the Keyword-URI mapping is depicted in Figure 1. The proposed technique is divided into three parts called Triple Construction, Auto-Suggestion, and Semantic Search Service. The triple construction technique discusses the data acquisition and the process of converting it to triples. The auto-suggestion technique discusses how the suggestions are derived from the local data store and highlights the added value of providing seamless URI mapping. In the semantic search service, the querying and filtering of retrieved results is discussed.

3.1 Triple Store Construction

The DBpedia data is maintained locally for guaranteed response and to avoid the negative consequences of a sudden downtime of DBpedia. The "Persondata" dump[11] was downloaded from DBpedia. This dump contains information about persons extracted from the English and German Wikipedia, represented using the FOAF vocabulary. ARC[12], a flexible system for RDF data, is then used to import this dump into a local triple store. This triple store provides an interface for the SPARQL queries. At the moment, there are 62,313 URIs of persons stored in the CAF-SIAL triple store.

3.2 Auto-Suggestion

When a user starts entering a keyword for a search, a SPARQL query is constructed on the fly on every key press and an AJAX-enabled autosuggestion module is invoked. The autosuggestion module is responsible for finding all possible occurrence of an entered person name in the local triple store and returning a list of suggestions. These suggestions help users in the following aspects:

- With auto-complete, users need to type less
- Give user leverage about searching possibilities within dataset
- On-the-fly disambiguation of concepts having similar or the same names
- Selecting the correct concept

[11]http://downloads.dbpedia.org/3.3/en/persondata_en.nt.bz2
[12]http://arc.semsol.org/

Figure 2. Autosuggestion

On user selection from any of the suggested option, the underlying URI of the selected keyword is passed on for further processing. The presentation of the list of suggestions is shown in Figure 2.

3.3 Semantic Search Service

In case the keyword is not mapped to any concept in the local triple store, the semantic search service is invoked. The public API of Sindice is used for this operation. It returns an RDF file containing number of URIs belonging to different data sources. This file is then parsed into the local triple store by using ARC. Further on, a URI filtering service is called, and the URI matching our set description, i.e. a DBpedia person type resource, is filtered out. If more than one URI belonging to DBpedia person type exist, the first one in the list is picked.

The SINDICE service provides a faster crawling procedure as compared to other semantic search engines. DBpedia releases new data dumps approximately every six months. Hence the newly entered or updated resources will not be part of the older DBpedia dump as well as our local triple store. Meanwhile Sindice, due to its fast crawling procedure, will be having an index of these newly added resources, which may be very useful to locate new resources. This will increase our system's performance and ensure up-to-date URI supply to users.

4. Concept Aggregation Framework

The Concept Aggregation Framework aggregates relevant concepts from DBpedia and organizes the most important informational aspects related to a resource.

The scope of this application is limited to DBpedia and Yago. DBpedia covers 23 types of resources (places, people, organizations, etc), we selected the resource type person for our initial proof of concept application.

The Concept Aggregation Framework is shown in Figure 3. The aggregation classification layer is responsible for aggregating

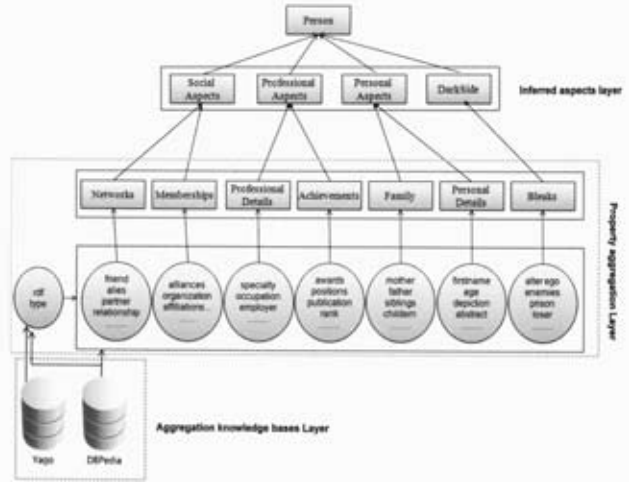

Figure 3. Concept Aggregation Framework

the most relevant information related to the person in question. This information is collected based on the list of related properties compiled at the property aggregation layer. The properties are extracted from knowledge bases shown in the aggregation knowledge bases layer.

4.1 Aggregation Knowledge Bases Layer

DBpedia, Yago and Umbel ontologies mainly contribute in the identification and classification of the resources. Two of them (DBpedia and Yago) are considered complete knowledge bases [Suchanek et al 2007]. The underlying mechanism in our system is as follows:

We have generated two knowledge bases, a DBpedia Property Dump and a Yago Classification Dump. The DBpedia Property Dump is built by querying each type of a person (Artist, Journalist, etc.) from SNORQL query explorer[13] (SPARQL endpoint of DBpedia). Then we aggregate all the distinct property sets for each person. Out of 21 queried person types in total, we were able to collect distinct properties of 18, which are presented in Table 1. It shows the number of distinct properties in total that we collected for a specific person as well as the number of properties picked by a set of experts, which will be mapped to defined aspects. The formulated query for this operation is given below:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?p
WHERE {
?s ?p ?o .
?s rdf:type <http://dbpedia.org/ontology/Artist> .
}
```

The Yago Classification Dump is built by querying subclasses of Person class from SNORQL query explorer. The query looks like this:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?s
WHERE {
?s rdfs:subClassOf <http://dbpedia.org/class/yago/Person100007
}
```

To decide which of these properties should be presented to the user, a query is formulated to get the count of every distinct property used for person type. After getting the count, the rank is assigned to each property. The higher the rank, the more prominently the property will be displayed. For example, some of the properties of person type Athlete like “Position” (70939 times), “clubs” (46101 times) and “debutyear” (9247 times) provide interesting stats to organize properties in a more conceivable fashion. The formulated query to get the count of each distinct property is given below.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?p count(DISTINCT ?s)
WHERE {
?s ?p ?o .
?s rdf:type <http://dbpedia.org/ontology/Athlete> .
}
```

[13]http://dbpedia.org/snorql/

Person Type	Total Properties	Picked Properties
Artist	2111	409
Journalist	186	55
Cleric	419	76
BritishRoyalty	252	47
Athlete	2064	496
Monarch	337	50
Scientist	421	126
Architect	132	41
PlayboyPlaymate	125	37
Politician	36	18
MilitaryPerson	725	158
FictionalCharacter	599	273
Criminal	287	74
CollegeCoach	282	124
OfficeHolder	1460	634
Philosopher	226	71
Astronaut	168	62
Model	211	99

Table 1. Person’s property list

4.2 Property Aggregation Layer

This layer first identifies the profession type. This works in two steps. In the first step, the resource type (RDF type) is identified by using DBpedia. In case that in the retrieved set of properties, there is no property mapped within DBpedia knowledge base, the system tries to map the retrieved property to a Yago class. For example if the retrieved property is “AustrianComputerScientist” which is not listed in DBpedia knowledge base, then the system maps it to the Yago hierarchy and can infer that the person belongs to the profession of “Scientist” because “AustrianComputerScientist” is a subclass of “Scientist”.

Based on a resource type, we have extracted all the possible properties from the DBpedia Property Dump. We then have manually identified sets of properties indicating an informational concept (networks, memberships, family, achievements etc.) related to a person. These concepts are aggregated and mapped to the related informational aspect identified in the inferred aspects layer. More than one concept may be mapped to a single informational concept defined at the inferred aspects layer.

4.3 Inferred Aspects Layer

The information for a resource such as person may be organized and viewed in different informational aspects like personal, professional, social etc. The most popular search engine like Google also tries to present such informational aspects related to a topic in its top results. It has been shown in [Brin and Page 2008] how Google rank its results to provide the most relevant contents. For example, in a response to a user query of “Bill Clinton”, Google top ten results are based, amongst other things, on personal information (biography) and his professional career (president, writer). These results, however, depend on the complex link analysis of Web pages (citations to Web pages from

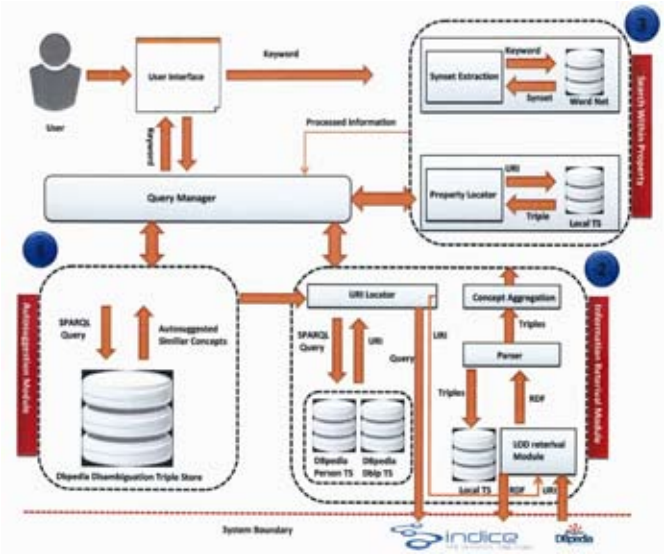

Figure 4. System Architecture

different sources) along weight mechanisms assigned to different factors [Feldstein 2009][Boykin 2005]. Google is considered as the most popular search engine having 64.2 % share in U.S search market [Lipsman 2009]. Inspired from Google's success in calculating and presenting the results in diverse and important informational aspects related to a query, we developed a concept aggregating framework.

5. System Architecture

The system architecture is depicted in Figure 4. The implemented system is divided into four modules called query manager, auto-suggestion module, information retrieval module and search within property module. The query manager is a controlling module of the application. It is responsible in translating the

keyword search query into SPARQL queries. The auto-suggestion module helps users to disambiguate entered search term. The information retrieval module is responsible for locating the URIs and extracting related information. The search within property module provides the facility of searching within all retrieved properties of a resource.

5.1 Auto-Suggestion Module

The query manager triggers the auto suggestion module by converting the searched keyword of a user into a SPARQL query. This module interacts with the DBpedia person and the DBpedia disambiguation triple store to autosuggest persons with names that match the entered keyword. This module has been discussed in detail in section 3.2. If the user does not select any of the suggested terms, or in case of a distinct query (no auto-suggestions yielded), the searched term will be passed on to the information retrieval module for further processing.

5.2 Information Retrieval Module

This module is further divided into four processes:

1. URI locator
2. LOD retrieval
3. Parser
4. Concept aggregation

The searched term is passed to the URI locator process which will query the locally maintained data sets (i.e. DBpedia Title TS, DBpedia Person Data TS, and DBLP TS) to get a URI. If this fails, a new query is formulated for the SINDICE[14] Web service to locate the URI. After locating the URI of a resource, the LOD retrieval process dereferences that URI at the DBpedia server to get the respective resource RDF description. This RDF description is further passed to the Parser process. This process will parse RDF description into triples and stored them locally. Then, the concept aggregation process is called to sort out the most important information aspect of the resource and in the end, the output is presented to the user.

5.3 Search Within Property Module

This module lets the user search within all properties of a resource retrieved from the information retrieval module. When a user enters a keyword to search some information about a resource, the synset extraction process queries wordnet[15] to retrieve the synset of searched keyword. This synset will be passed to the query manager and for each word in the synset it will query the local triple store through the property locator process. The property locator process matches the keyword as substring in the retrieved property set. All matched properties are then extracted and presented to the user.

6. Case Study

The working of the system is described with the help of a use case scenario. We have selected "Arnold Schwarzenegger" affiliated with four interesting and diverse professions along with multiple awards and achievements.

These capabilities make him a distinct person and a suitable choice for the use case. The application flow is explained as follows: User starts typing the search term "Arnold". The persons' names starting with the keyword "Arnold" are auto-suggested. For example "Arnold Bax", "Arnold Bennett", "Arnold Schwarzenegger" etc. as depicted in Figure 2.

The user selects "Arnold Schwarzenegger" to see his details as shown in Figure 5. The output is comprised of different informational aspects such as social, personal, and professional. Important properties are shown on the top for each informational aspect. The important property list was prepared manually and the weight to each property is assigned on the basis of its count automatically.

The screenshot shows how his important professional details have been displayed concisely and in easily graspable manner.

7. System Evaluations and Discussions

The system was evaluated with the help of user interviews. We collected data with the help of combining focus groups [Kitzinger 1995] and post-search interviews. We held two focus group sessions having 4-6 participants each (10 participants in total). All participants of both groups were Web users and had knowledge of basic Web search. One focus group comprised of users having experience in Semantic technologies while the other user group was naive Web users. Each group session was conducted by a skilled representative.

The selected application for evaluation in comparison with CAF-SIAL were Marble, Snorql, and Freebase. We conducted semi-structured interviews. Users were asked about comments, feedbacks, overall satisfaction, problems faced etc.

[14] http://sindice.com/
[15] http://wordnet.princeton.edu/wordnet/

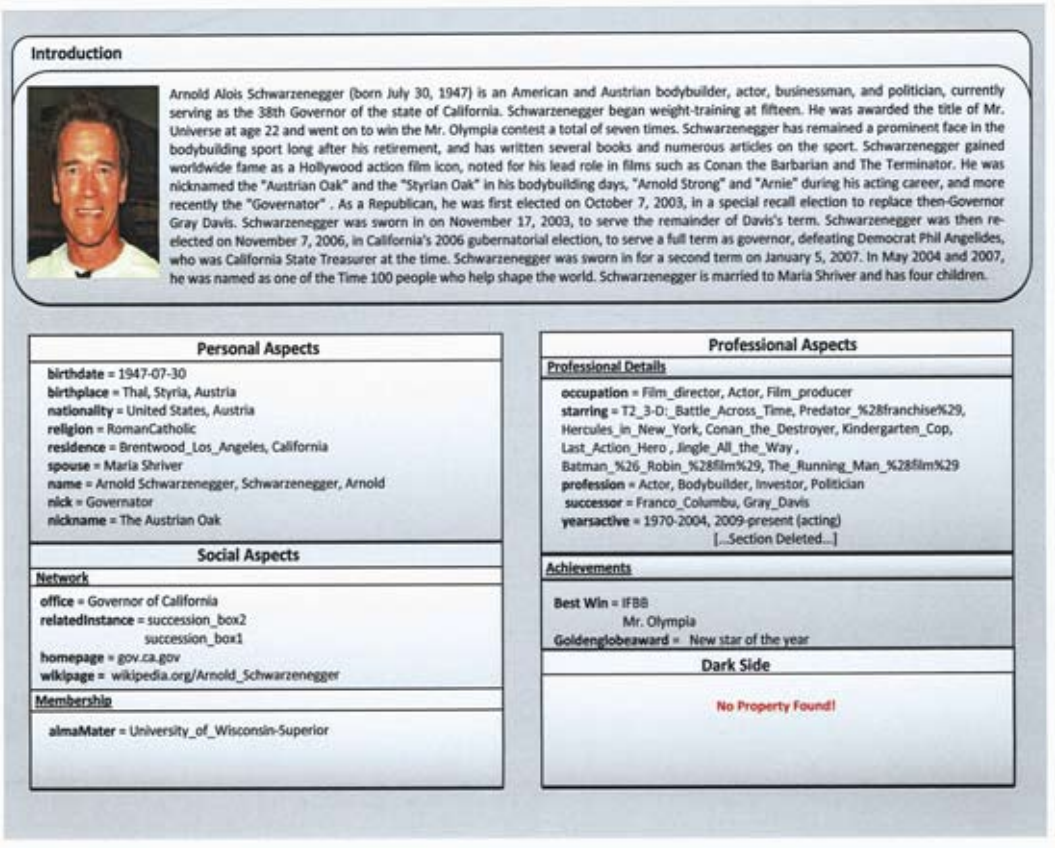

Figure 5. Case Study

The interviews showed that it was very difficult to get the sought-after information from Marble and Snorql, because the users did not know the exact URI of the resource, and due to the difficulty of formulating Subject Predicate Object logic. On the other hand, Freebase and CAF-SIAL systems were easier to use. Although Freebase was comparatively better in terms of providing rich information and content organization, CAF-SIAL was useful most of the time to get the sought-after information. One negative aspect that was mentioned about CAF-SIAL was the fact that when the users searched for a person and then clicked on a particular property within the retrieved result set, they were again redirected to complex systems like DBpedia. Then again it became difficult to find required information from a long list of properties.

When comparing CAF-SIAL to Freebase, there are some noteworthy differences:

1. Dbpedia, which provides that basis for CAF-SIAL, is built around a controlled vocabulary (an ontology, actually), whereas freebase adopts the folksonomy approach in which people can add new categories much like tags [O'Reilly 2007].
2. Along with the semi-automatic approach of Freebase to collect and organize data in to their knowledge base, a group of editors is responsible to pre-check the organization and add new knowledge in a structured way manually. On the other hand, CAF-SIAL works on a set of heuristics. These heuristics were defined manually once by experts and can then be applied by the system to organize knowledge in an entirely automated way. The system makes a transition to exploit LOD resources in an autonomous way, which could provide significant help in navigating the ever-growing LOD cloud.

The system has been made publicly available at http://cafsial.opendatahub.org/. For continuous evaluations, users can give feedback at any time online. The submitted information is saved and we plan to extend the system by incorporating users' comments and feedback.

8. Conclusion and Future Work

This work tries to bridge the gap between semantic search and the end user. The proposed keyword-based search mechanism has simplified the process of finding information from LOD by hiding underlying semantic logic. With the help of Concept Aggregation Framework, the information related to a resource (consisting of hundreds of properties) was structured in major and most relevant categories of informational aspects. This reduced the users' cognitive load to find the required information. Keyword-URI locating technique was very helpful to identifying a particular resource from huge LOD repository. The evaluation of the systems has shown promising results and we plan to extend this application with the help of further users' online evaluations.

9. Acknowledgements

This contribution is partly funded by the Know-Center and the Higher Education Commission of Pakistan. The Know-Center is funded within the Austrian COMET program — Competence Centers for Excellent Technologies — under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

[1] Berners-Lee, T. (July 2006). Linked Data Design Issues. http://www.w3.org/DesignIssues/LinkedData.html (accessed 22, Sep.2009).

[2] Latif, A., Hoefler, P., Stocker, A., Us-saeed, A., Wagner, C. (2009). The Linked Data Value Chain: A Lightweight Model for Business Engineers. *In:* Proceedings of I-Semantic. Graz, Austria.

[3] Chakrabarti, S. (2004). Breaking through the Syntax Barrier: Searching with Entities and Relations. *In:* Proceedings of Principles and Practice of Knowledge Discovery in Databases *(PKDD).* Springer, p. 9–16, Berlin, Heidelberg, Germany.

[4] Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D. (2006). Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. *In:* Proceedings of 3rd International Semantic Web User Interaction Workshop. Athens, Georgia, USA.

[5] Kobilarov, G., Dickinson, I. (2008). Humboldt: Exploring Linked Data. *In:* Proceedings of Linked Data on the Web Workshop *(LDOW)*. Beijing, China.

[6] Berrueta, D., Phipps, J. (2009). Best Practice Recipes for Publishing RDF Vocabularies. http://www.w3.org/TR/swbpvocab-pub/ (accessed 22, Sep 2009).

[7] Chimezie. (2009). Dereferencing a URI to RDF. http://esw.w3.org/topic/DereferenceURI (accessed 22, Sep.2009).

[8] Russell, A. R., Smart, P., Braines, D., R. Shadbolt, N. (2008). NITELIGHT: A Graphical Tool for Semantic Query Construction. *In:* Proceedings of Semantic Web User Interaction Workshop *(SWUI).* Florence, Italy.

[9] Kiefer, C., Bernstein, A., Stocker, M. (2007). The fundamentals of iSparql a virtual triple approach for similarity-based Semantic Web tasks. *In:* Proceedings of International Semantic Web Conference *(ISWC)*. Busan, Korea.

[10] Samur, C. F., Daniel, S. (2009). Explorator: a tool for exploring RDF data through direct manipulation. *In:* Proceedings of Linked Data on the Web Workshop *(LDOW)*. Madrid, Spain.

[11] Catarci, T., Levialdi, F. M., Batini, S. C. (1997). Visual Query Systems for Databases: A Survey. *Journal of Visual Languages and Computing*. 8 (2) 215-260.

[12] Hildebrand, M., Ossenbruggen, V., Hardman, J. (2006). Facet: A Browser for Heterogeneous Semantic Web Repositories. *In:* Proceedings of International Semantic Web Conference *(ISWC)*. Athens, Georgia, USA.

[13] Harth, A., Umbrich, J., Hogan, A., Decker. S. (2007). YARS2: A Federated Repository for Querying Graph Structured Data from the Web. *In:* Proceedings of International Semantic Web Conference *(ISWC)*. Springer, Busan, Korea.

[14] Suchanek, M., Kasneci, F., Weikum, G. (2007). Yago: A Core of Semantic Knowledge - Unifying WordNet and Wikipedia. *In:* Proceedings of 16th International World Wide Web Conference *(WWW)*. Banff, Alberta, Canada.

[15] Boykin, Jim. (2005). Google Top 10 choices for search results. http://www.jimboykin.com/googles-top-10-choices-forsearch-results/ (accessed 22, Sep.2009).

[16] Feldstein, A. (2009) . Search ranking factors. http://www.seomoz.org/article/search-ranking-factors (accessed 22, Sep.2009).

[17] Brin, S., Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*. 30, 107-117.

[18] Lipsman, A. (2009). ComScore Releases April 2009 U.S Search Engine Rankings. http://www.comscore.com/Press_Events/Press_Releases/2009/5/comScore_Releases_April_2009_U.S._Search_Engine_Rankings (accessed 22, Sep.2009).

[19] Auer, S., Bizer, C., Idehen, K. (2009). DBpedia Knowledge Base. http://dbpedia.org (accessed 22, Sep.2009).

[20] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. *In:* Proceedings of the 6th International Semantic Web Conference *(ISWC)*. Springer, Busan, Korea .

[21] Sauermann, L., Cyganiak, R., Ayers, D., Vlkel, M. (2008). Cool URIs for the Semantic Web. W3C Interest Group Note, http://www.w3.org/TR/2008/NOTE-cooluris-20081203/ (accessed 22, Sep.2009).

[22] Hepp, M., Siorpaes, K., Bachlechner, D. (2007). Harvesting Wiki Consensus Using Wikipedia Entries as Vocabulary for Knowledge Management. *IEEE Internet Computing.* 11 (5) 54-65.

[23] Tummarello, G., Delbru, R., Oren, E. (2007). Sindice.com: Weaving the open linked data. *In:* Proceedings of International Semantic Web Conference *(ISWC)*. Busan, South Korea.

[24] Cheng, G., Ge, W., Qu, Y. (2008). Falcons: Searching and Browsing Entities on the Semantic Web. *In:* Proceedings of 17th International World Wide Web Conference *(WWW)*. p. 1101-1102, Beijing, China.

[25] Ding, L., Finin, T., Joshi, A., Pan, R. S., Cost, R., Peng, Y., Reddivari, P. C., Doshi, V., Sachs, J. (2004). Swoogle: A Search and Metadata Engine for the Semantic Web. *In:* Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management. P. 652 - 659, Washington, D.C., USA.

[26] Kitzinger, J. (1995). Qualitative research. Introducing focus groups. *British Medical Journal*. 311, 299-302.

[27] O'Reilly, Tim. (2007). Freebase Will Prove Addictive. O'Reilly Radar. http://radar.oreilly.com/archives/2007/03/freebase_will_p_1.html (accessed 22, Sep.2009).

Authors Biographies

Atif Latif is a PhD student at Graz University of Technology, Austria since 2008. He is affiliated with "Institute of Knowledge Management and Know-Center", Austria's COMET Competence Center for Knowledge Management. He is/was involved in scientific services (Committee-member/session-chair) for number of international conferences. He received his master's degree in Computer Science from Quaid-i-Azam University, Islamabad in 2007 and awarded with excellent Grade in his final thesis. His research interest includes practical aspects of Semantic Web, Linked Data in various paradigms (Social Web, Digital Libraries, and Information Supply).

Muhammad Tanvir Afzal studied Computer Science at Graz University of Technology, Austria and was awarded Ph.D. with distinction in 2010. He received his master's degree in Computer Science from Quaid-i-Azam University, Islamabad, Pakistan and secured Gold Medal in 2004. During his Ph.D., he spent one month each in Technical University Braunschweig and Universiti Malaysia Sarawak for research activities. He worked in software houses, R&D institutes, and universities at various levels. He worked on Context-aware systems for Journal of Universal Computer Science (J. UCS). He authored more than 20 publications in international journals and conferences. He is/was serving as editor/reviewer/session-chair for various reputable international journal and conferences. His research areas include personalized services, Semantic Web and web/text mining.

Anwar us Saeed is registered as Doctoral student in the (IWM) Knowledge Management Institute at Graz University of Technology since 2007. He studied Systems Engineering and earned his Masters degree in 2002. From 2002 to 2006 he was senior engineer of the directorate of safety Pakistan. Since 2007 he has been the researcher of the Know-Center, a Research Institute for Knowledge Management in Graz, Austria. His main research areas include Knowledge Management, Knowledge diffusion in new participatory web and social software.

Patrick Hoefler received his master's degree in information management from the University of Applied Sciences FH Joanneum Graz in 2005. He currently works at the Know-Center Graz and studies for his Ph.D. degree at the Knowledge Management Institute at Graz University of Technology. His research focuses on practical aspects of the Social Semantic Web. For the last three years, he has also been involved in organizing the I-KNOW, the International Conference on Knowledge Management, held every September in Graz, Austria.

Klaus Tochtermann studied Computer Science and earned his Dr. degree in 1995. In 1996 he spent his post-doc at the A&M University in Texas. From 1998 to 2000 he was head of the department for Environmental Information Systems at the Research Institute for application oriented Knowledge Processing, Ulm, Germany. Since 2000 he has been the scientific director of the Know-Center, a Research Institute for Knowledge Management in Graz, Austria. Since 2004 K. Tochtermann has been head of the Institute for Knowledge Management at Graz University of Technology. His main research areas include Knowledge Management, Knowledge Technologies and Corporate Web 2.0.