HTML5 based Facet Browser for SPARQL Endpoints

Martina Janevska, Milos Jovanovik, Dimitar Trajanov
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje, Republic of Macedonia

***Abstract*—The Linked Data concept uses a collection of Semantic Web technologies in order to interconnect, publish, and share pieces of data on the Web, in a machine-readable format. It enables querying and combining data from different datasets together in order to retrieve specific information and enable use-case scenarios which are unavailable over isolated datasets. However, the process of querying linked data published on different places on the Web poses several challenges. Generally, each user should know the schema of the data, write a query and access a relevant linked data endpoint. According to statistics, endpoints are often unavailable and have significant downtime, so this presents a serious obstacle in application development and scenarios which rely on the data. In this paper, we present a facet browser for SPARQL endpoints, based on HTML5. It allows users to search and retrieve RDF triples based on a keyword, from public SPARQL endpoints. By using HTML5 Web Storage, the triples from the results can be saved in the browser, locally, for future use. The facet browser provides management functionalities over the stored data - capabilities to update, refresh, modify, delete and download the triples in various RDF formats: JSON-LD, Turtle, NTriples, RDF/XML, JSON, CSV. The locally stored RDF triples can also be shared with other users. We believe that these features of the facet browser will help overcome the endpoint downtime issues, by providing offline data accessibility for the user and his applications.**

Keywords—Facet Browser; Linked Data; SPARQL; HTML5; Semantic Web;

I. INTRODUCTION

The Linked Data principles propose using the Web to create typed links between data from different sources. These may be as diverse as databases maintained by two organizations in different geographical locations, or simply heterogeneous systems within one organization that, historically, have not easily interoperated at the data level [1]. W3C provides a palette of technologies (RDF, GRDDL, POWDER, RDFa, the upcoming R2RML, RIF, SPARQL) to get access to the data[1]. SPARQL is a query language that is flexible and is mostly used for interacting with RDF databases also known as triple stores. As a standardized query language it contains a lot of specifications, query operations, request and response formats.

Every dataset declared as Linked Data can be accessed using the SPARQL language and queries distributed over different SPARQL endpoints. For retrieving the data, a user has to access available endpoints and to be able to query them. The ability to query the data space of Linked Data provides benefits which have not been possible before. Data from different data sources can be gathered, and scattered information from multiple sources can be joined in order to achieve a more complete view of a given domain and support more complex use-case scenarios.

Executing queries over the Web of Linked Data poses several challenges that do not arise in traditional query processing. Due to the openness of the data space, it is not possible to know all data sources that might be relevant for answering a query in advance. People who need some data from the Linked Data cloud[2] sometimes do not know how the specified data is represented in datasets and they face the challenge to write complex queries in order to extract results.

Another issue which arises in the domain of querying data over the data space of Linked Data is the availability of the SPARQL endpoints. According to statistics[3], these endpoints are often unavailable and have significant downtime. This presents a serious obstacle in developing 'killer applications' over Linked Data datasets, and forces developers to focus their time and energy on other technologies and approaches.

In this paper, we present a facet browser for SPARQL endpoints, which tries to overcome these two main issues. Firstly, it offers keyword lookup in RDF datasets from SPARQL endpoints, mitigating the issue of schema knowledge beforehand, or knowing the SPARQL language at all. Secondly, it allows storage of the results received from a SPARQL endpoint, for future use. The results are being stored in HTML5 Web Storage, in the user browser, and can be manipulated by the user. The data can also be serialized into all common RDF formats, and shared with others.

II. RELATED WORK

Browsers can be a crucial tool when someone is dealing with data from the Linked Data cloud. They should provide different use-cases and options for users and their needs. That is the main reason why there are a number of browser applications for Linked Data and they all provide different approaches in dealing with data and presenting results to the end user.

The Tabulator project[4] represents a generic data browser, which provides ways for browsing RDF resources published on the Web, and follow RDF links from one resource to

[1] http://www.w3.org/standards/semanticweb/data

[2] http://lod-cloud.net/

[3] http://sparqles.okfn.org/availability

[4] http://www.w3.org/2005/ajar/tab

another [2]. The main goal of Tabulator is to increase the usage of Linked Data, explore the potentials and restrictions of the Semantic Web architecture and to increase development in the field of generic data interfaces.

The Disco - Hyperdata Browser[5] is a browser which is used for handling the Semantic Web as an unbounded set of data sources. This tool renders all information related to a resource, which is specified by his URI entered into the navigation box. For this resource, the user gets a description which contains hyperlinks or facets allowing him to navigate between resources. When a user moves from one resource to other, the browser dynamically retrieves and displays information as a property-value table. Also, the browser stores all retrieved RDF graphs, whose hyperlinks have been clicked, in a session cache and provide an option for showing them in a list in a new browser window.

Swoogle[6] is a specialized web based data browser used for discovering, analyzing and indexing of data from datasets published on the Web with Semantic Web technologies. Swoogle explains and records significant metadata about data published in these datasets and their fundamental parts (e.g., terms, individuals, triples). As a browser it uses its search and navigation services in order to provide scalable service for accessing Semantic Web data and finding relevant documents [3]. On one hand, similar to our solution, Swoogle represents a keyword-based search engine and it does not require schema knowledge and query language expertise. That is an important reason why it is really appropriate for non-technical users. On the other hand, similarly as Tabulator and Disco – Hyperdata Browser, Swoogle does not provide any options for further management and wider usage of retrieved RDF results, which is one of main focuses used in our approach.

Longwell[7] is a web-based faceted browser, considered as a combination of the flexibility of the RDF data model and the effectiveness of the faceted browsing paradigm. It is a powerful tool, and just like our solution it enables visualization and browsing complex RDF datasets, allows the user to quickly get an overview of what data is present in the dataset, and offers specific information about resources.

Virtuoso Faceted Browser[8] is keyword-based faceted browser and as a solution it is the most advanced. The user can enter a text search pattern and get a results page containing a list of literal value snippets from property values associated with the searched pattern. With a click on some of the entities or relations, the user gets new results with description of that particular object. There is also an option for getting raw data from search result in several formats, such as CSV, JSON, XML, N-Triples, etc.

[5] http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/disco/
[6] http://swoogle.umbc.edu/
[7] http://simile.mit.edu/wiki/Longwell
[8] http://lod.openlinksw.com

The facet browser we have developed has several additional options which differentiate it from all of these examples. The user of our applications gets the results for the used keyword in a table, and has the additional option to save them for later use. The saved data can be managed by the user; it can be updated and refreshed, modified, removed, and serialized on the local machine in various RDF formats. The local storage of RDF data, in the browser, allows other future features, as well.

III. THE FACET BROWSER

The facet browser for SPARQL endpoints (Fig. 1) is a HTML5 web-based application which has a main focus of searching and retrieving RDF triples which contain a given keyword, from a specified SPARQL endpoint. The results are presented in a human-readable representation. The endpoint does not require any SPARQL or RDF knowledge from the user.

A. Data Retrieval

One of the main functionalities in our application is the effective search for data from a SPARQL endpoint. There are two basic things which should be provided by the user to the application: a search term and SPARQL endpoint URL. The application takes the search term of the user as a keyword, and sends a SPARQL query to the endpoint provided by the user.

In order to get these results, we use the following SPARQL query, in which we replace "pattern" with the keyword of the user:

```
SELECT ?s ?p ?o
WHERE
{
      ?s ?p ?o.
      FILTER ( regex(?o, "pattern", "i"))
}
LIMIT 1000
```

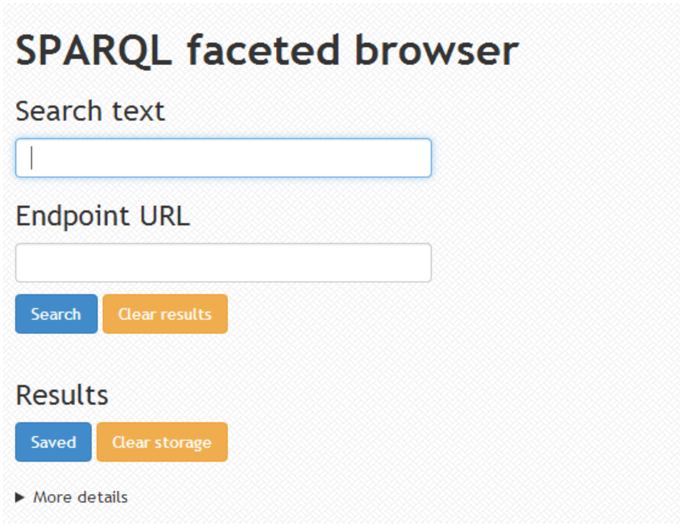

Fig. 1. The facet browser interface.

With the use of this specific query, we get a number of triples, composed of literal value snippets from property values related with the searched keyword.

In the query we place a limit of maximum 1000 triples, in order to comply with the fair use policy of Linked Data. The results are shown in a table with three columns: entity, relation and value, sorted by entity values, and with the possibility of displaying a particular number (10, 50, 100 or 1000) of result rows per page (Fig. 2).

Faceted navigation is an examining technique which uses several different ways for navigating through a collection of components, instead of using a specific and determined order. Facet browser interfaces provide navigation through collections of data in a user-friendly way [4]. When users are facing Linked Data datasets, they often are not sure what exactly they are looking for, so they may not always be familiar with the domain and its schema. Or, the users may just want to learn something new about a particular matter.

With our faceted browser we allow users to search data through multiple successive iterations without having excessive (or any) knowledge of the datasets. In the results table, the entities and the relations are displayed as facets, providing a multi resource scheme. The users can click on the any of the facets, and get a description of the resource or relation. This is done by the application by sending a new query to the endpoint, asking for the description of that resource or relation. As a result from the 'describe' query, the browser displays a table with two columns: relation and value, displaying all of the relations and object values related to the entity (Fig. 3).

There are also options for the users to see more details about the displayed result data, and get all of the results from the endpoint, not just the first 1000. This supports use-cases for more advanced users, who can be interested in more technical details about the query, or would like to get all possible RDF triples for their initial query.

The application also has the capability of saving data. We use HTML5 for the web application, which provides us with the Web Storage functionality. This functionality supports persistent data storage with great capacity. It offers two different types of storage: local and session. For our application we use the former, because it stores data with no expiration date, which means the data will not be deleted and will persist when the user closes the browser and his session ends. The application allows the users to define details about the RDF triples before saving them in Web Storage (Fig. 4).

This Web Storage functionality allows our application to provide offline access to the RDF results from the saved queries. This is the feature which helps Linked Data users mitigate the issue of having the needed SPARQL endpoint available at any given moment. This client-side storage is a programming model which does not require a server infrastructure and single user can only access his local files, which means that our web application becomes personalized.

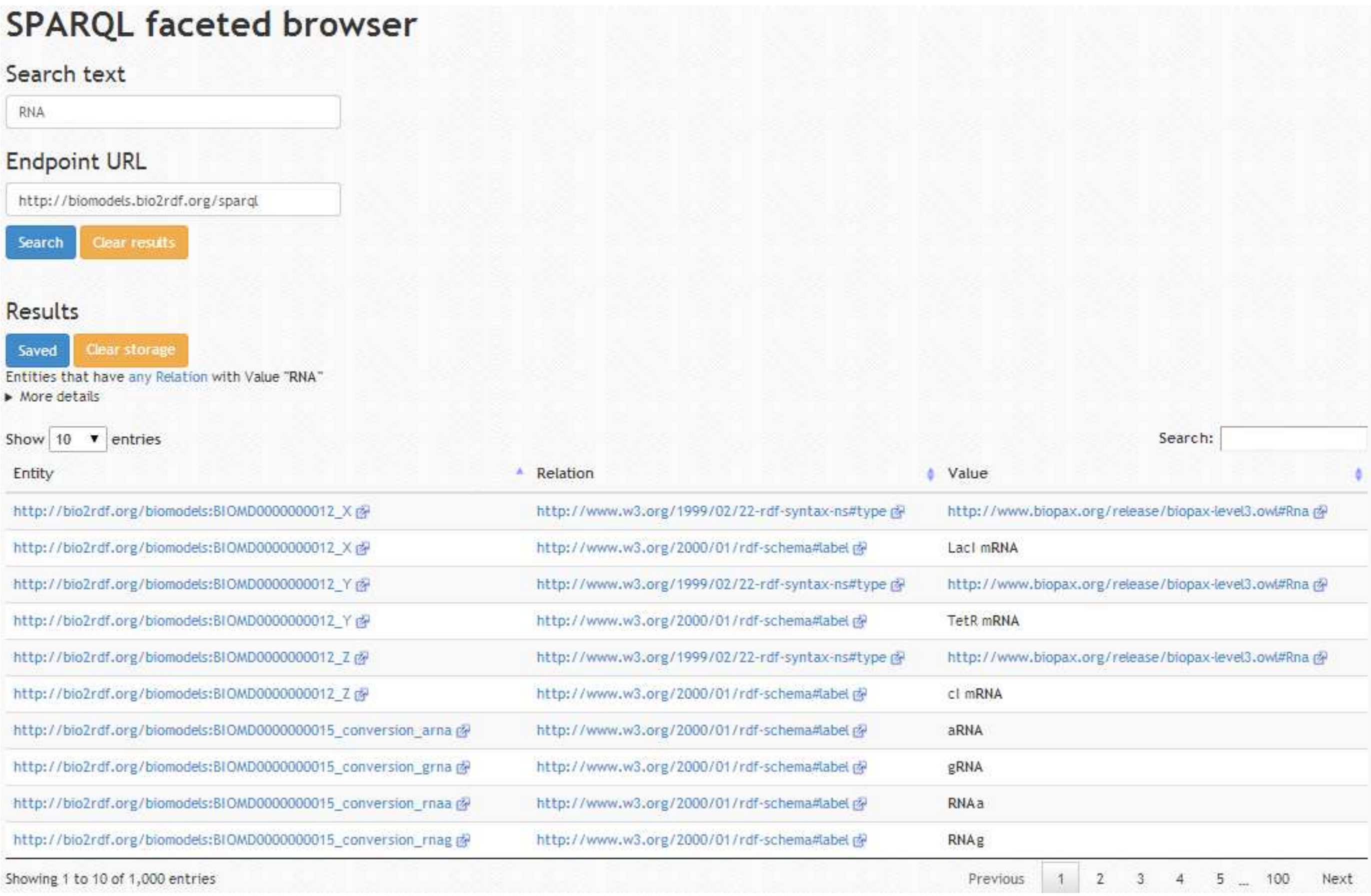

Fig. 2. Results from searching the keyword 'Aspirin'.

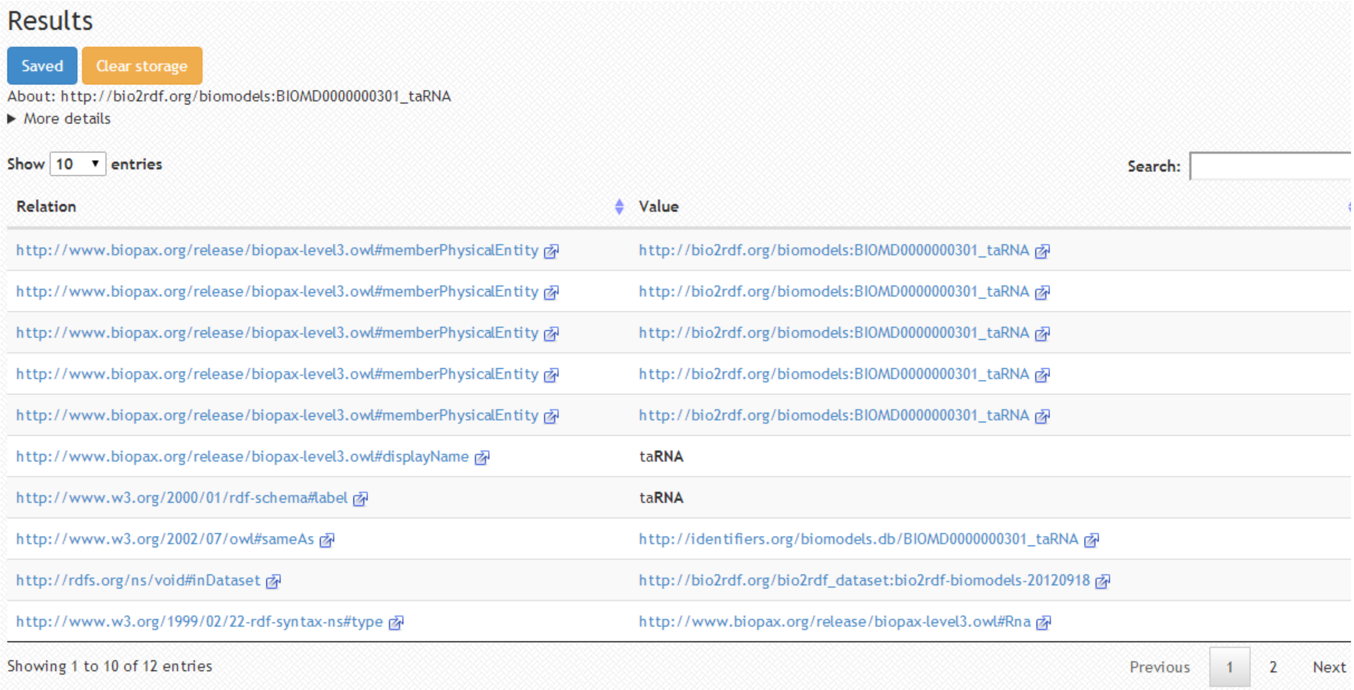

Fig. 3. A resource description, showing its relations and values.

After a result set is saved, the user can then manage it, or continue querying a SPARQL endpoint.

B. *Storage Management*

As we already mentioned, the application allows the users to access and manage their stored data. An example stored dataset is shown on Fig. 6. Each saved dataset, aside from the RDF triples, contains a mnemonic name, graph name and a description, which are used to distinguish the datasets.

The application provides management options to the users for manipulating the RDF triples from a stored dataset (Fig. 6). The 'Details' option allows a user to see all of triples which are parts of that dataset, instead of only the first 10. A user can also edit the mnemonic name, the graph name and the description (Fig. 5).

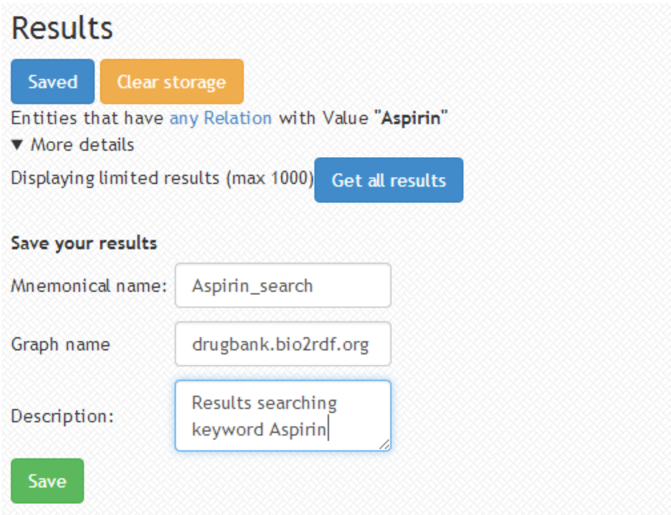

Fig. 4. Form for saving data in local storage.

However, one of the main functionalities over the stored datasets is the ability to edit the RDF triples themselves (Fig. 5). This is also a feature available in the application, and its intention is to provide the user with the ability to modify the datasets in order to correct, change or delete some of the RDF triples which he intends to use for other purposes.

In order to provide the capabilities of storing the dataset triples as into HTML5 Web Storage, as well as review and manage them, we use the JavaScript wrapper library triplestore.js[9]. The library offers several functionalities which provide RDF data administration for the user. Additionally, it is a library which is very simple to use and develop with.

When a user has an RDF dataset stored in his browser, it can only be viewed and manipulated through the application. However, if the user needs a dataset for other purposes, such as using it as a data layer for an application, or using it in data analysis, the user may need to serialize the data and save it locally on his machine.

Our application offers this functionality to the users, by allowing serialization of a saved dataset in various RDF formats: JSON-LD, Turtle, NTriples, RDF/XML and CSV. The serialized data is showed in the web application, for supporting use-cases when the user would like to copy and paste the dataset content, but the application also allows for the serialized RDF dataset to be downloaded as a file, on the local machine (Fig. 7).

[9] http://www.w3.org/2013/04/semweb-html5/triplestoreJS/

Mnemonical name	Aspirin_search	
Graph name	drugbank.bio2rdf.org	
Description	Result search for key	
Entity	**Relation**	**Value**
http://bio2rdf.org/drugbank_resource:DB00055_I	http://www.w3.org/2000/01/rdf-schema#label	DDI between Drotrecogin alfa and Vilazodone - Increa
http://bio2rdf.org/drugbank_resource:224b519e4dc3d785b30acb492b685e77	http://www.w3.org/2000/01/rdf-schema#label	CVS Pharmacy non-aspirin 500 mg caplet [drugbank_resource:224b519e4dc3d785b30acb492b685e77]
http://bio2rdf.org/drugbank_resource:a4e1740f0560bfd10771498f0034d70f	http://www.w3.org/2000/01/rdf-schema#label	CVS Pharmacy non-aspirin 500 mg tablet [drugbank_resource:a4e1740f0560bfd10771498f0034d70f]
http://bio2rdf.org/drugbank_resource:36d56140405a7e4b513a94eed0d8b684	http://www.w3.org/2000/01/rdf-schema#label	Non-aspirin 500 mg tablet [drugbank_resource:36d56140405a7e4b513a94eed0d8b684]

Fig. 5. Local storage data management.

Because of the dynamic nature of Linked and Open Data datasets, a user may want to update a locally saved resulting RDF dataset, once the source SPARQL endpoint is available.

In order to provide support for this scenario, we added the functionality of refreshing the data from a stored RDF dataset. In order to do this, we store the initial query used for obtaining the dataset, along with the source SPARQL endpoint, so when a user needs to refresh the data, we send a new query to the SPARQL endpoint and retrieve the updated results. After a dataset refresh, we prompt the user in order to see whether he wants to do an update refresh or a clean refresh. If he chooses the former, only the newly retrieved triples which did not exist in the previously stored results are added to the RDF dataset, and if he chooses the latter, the newly retrieved results will replace the previously stored ones.

Since HTML5 Web Storage has different storage capacity for local storage depending on the user web browser and RDF results retrieved from an endpoint can reach the size of thousands of triples, we also provide the option for deleting an RDF dataset from the web browser storage (Fig. 6).

C. *About the application*

The Web of Linked Data is open, so applications which work with Linked Data can follow relations between data and learn about new data sources. As a result of that, with increase of published data, these applications will provide to the users more complete information and knowledge [5]. This also means that anyone can publish data structured with these technologies on the Web, and become part of the increasingly growing LOD cloud. With this, every Linked Data publisher adds more possible use-cases over the interconnected datasets on the Web, and with this, indirectly adds more value to the LOD cloud datasets.

Therefore, we based our HTML5 facet browser on the same principles of openness and collaborative value leverage. We developed and published it as an open-source application[10] where other can also participate in further development, and deployed a public instance[11].

IV. Conclusion and Future Work

In general, Semantic Web technologies are used for publishing and managing Open Data, and interconnecting it with other data in the Linked Data cloud; using already

Mnemonical name	Aspirin_search	
Graph name	drugbank.bio2rdf.org	
Description	Result search for keyword Aspirin	
Entity	**Relation**	**Value**
http://bio2rdf.org/drugbank_resource:DB00055_DB00758	http://www.w3.org/2000/01/rdf-schema#label	DDI between Drotrecogin alfa and Clopidogrel - Antiplatelet agents such as clopidogrel may enhance the adverse/toxic effect of Drotrecogin Alfa. Bleeding may occur. Incre
http://bio2rdf.org/drugbank_resource:224b519e4dc3d785b30acb492b685e77	http://www.w3.org/2000/01/rdf-schema#label	CVS Pharmacy non-aspirin 500 mg caplet [drugbank_resource:224b519e4dc3d785b30acb492b685e77
http://bio2rdf.org/drugbank_resource:a4e1740f0560bfd10771498f0034d70f	http://www.w3.org/2000/01/rdf-schema#label	CVS Pharmacy non-aspirin 500 mg tablet [drugbank_resource:a4e1740f0560bfd10771498f0034d70f]
http://bio2rdf.org/drugbank_resource:36d56140405a7e4b513a94eed0d8b684	http://www.w3.org/2000/01/rdf-schema#label	Non-aspirin 500 mg tablet [drugbank_resource:36d56140405a7e4b513a94eed0d8b684]
http://bio2rdf.org/drugbank_resource:0161e8c5a8871f72f9d96ecf4b7d74bb	http://www.w3.org/2000/01/rdf-schema#label	Non-aspirin 500 mg geltab [drugbank_resource:0161e8c5a8871f72f9d96ecf4b7d74bb]
http://bio2rdf.org/drugbank_resource:calculated_property_DB00945_12	http://www.w3.org/2000/01/rdf-schema#label	Traditional IUPAC Name: aspirin from ChemAxon [drugbank_resource:calculated_property_DB00945_12] [drugbank_resource:calculated_property_DB00945_12]
http://bio2rdf.org/drugbank_resource:Aspirin-with-Stomach-Guard	http://www.w3.org/2000/01/rdf-schema#label	Aspirin with Stomach Guard [drugbank_resource:Aspirin-with-Stomach-Guard]
http://bio2rdf.org/drugbank_resource:Extra-Strength-Aspirin-Backache	http://www.w3.org/2000/01/rdf-schema#label	Extra Strength Aspirin Backache [drugbank_resource:Extra-Strength-Aspirin-Backache]
http://bio2rdf.org/drugbank:DB00945	http://www.w3.org/2000/01/rdf-schema#seeAlso	http://www.drugs.com/aspirin.html
Details Update Refresh Clean Refresh Delete		

Fig. 6. Table of data saved in local storage.

[10] https://github.com/mjanevska/html5-sparql-browser
[11] http://fct.linkeddata.finki.ukim.mk

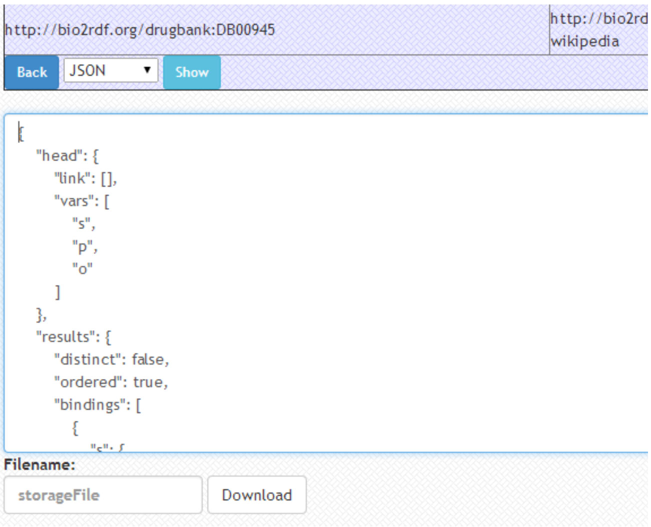

Fig. 7. Data serialization and download.

published Open and Linked Data from public SPARQL endpoints; developing applications based on Open and Linked data, either stored locally, or available on the Web.

Our HTML5 facet browser is intended to be used as a search engine for datasets published as Linked Data, available through SPARQL endpoints. It provides a flat browsing capability through the dataset by combining search, facets and operations on sets of resources.

Using HTML5 Web Storage, we enable the option to store the RDF results for future use, locally, in the browser storage. The facet browser provides a broader view on the stored data and management functionalities over it. The locally stored RDF triples can be serialized in several RDF formats, downloaded as separated files and then used in applications, analysis or shared with other users. We believe these features help the users to overcome the existing SPARQL endpoint availability issues.

The original goal of the project was to simplify the process of searching the Web of Open and Linked Data by using facets, and to provide more services and options to their users. With our application, we would also like to encourage development of innovative applications and services over Linked and Open Data.

The future development of the project will include a feature for providing more transparent sharing of the stored RDF datasets. We plan to add an option for publishing a dataset in a chosen RDF format on a permanent URL, in order to support use-cases in which a user would like to share an RDF dataset with other, via the Web.

We also plan to add more management options for the stored data. By using other functionalities from triplestore.js, we can enable scenarios in which the users would filter a stored RDF dataset and obtain smaller subsets of it. These filtered subsets will contain RDF triples with subjects which are matched with an optional property and value, or triples which contain values that match with a specified subject and optional property.

ACKNOWLEDGEMENT

The work in this paper was partially financed by the Faculty of Computer Science and Engineering, at the Ss. Cyril and Methodius University in Skopje, as part of the research project “Semantic Sky 2.0: Enterprise Knowledge Management”.

REFERENCES

[1] Bizer, C., Heath, T., Berners-Lee, T.: “Linked Data - the story so far”. Journal on Semantic Web and Information Systems, 2009.

[2] Berners-Lee, T.: “Tabulator: Exploring and analyzing linked data on the semantic web”. Proceedings of the 3rd International Semantic Web User Interaction Workshop, 2006.

[3] Finin, T.: "Swoogle: Searching for knowledge on the Semantic Web" Proceedings, AAAI 05, 2005.

[4] Brunetti J. M., Gil R., Garcia R.: “Facets and Pivoting for Flexible and Usable Linked Data Exploration”. Interacting with Linked Data, 2012.

[5] Bizer, Christian. "The emerging Web of Linkedata." Intelligent Systems, IEEE 24.5, 2009.