Spacetime: a Two Dimensions Search and Visualisation Engine Based on Linked Data

Fabio Valsecchi and Marco Ronchetti
DISI, Università degli Studi di Trento
Povo di Trento, Italy
fabiovalse@gmail.com, marco.ronchetti@unitn.it

Abstract— **DBpedia is one of the most interesting projects in the arena of the Semantic technologies. However, being able to extract useful information from it is not a trivial task for a user without a specific competence. We present the Spacetime, a two dimensions search engine that provides a simple visualization user interface for making it easy for a generic user to perform certain types of queries on the DBpedia body, and having results shown in a graphic and animated form. Spacetime has been equipped with various features, such as heat maps, time sliding animations, map aggregations, icon map customization and map saving and loading.**

Keywords-DBpedia; Wikipedia; Visualization; GUI.

I. INTRODUCTION

DBpedia is a community effort to extract structured information from Wikipedia, the well known collaboratively edited, multilingual, free Internet encyclopaedia supported by the non-profit Wikimedia Foundation. Wikipedia has over 25 million articles in various languages, written collaboratively by volunteers around the world. Over 4 million articles are present in the English Wikipedia alone. The DBpedia project extracts a subset of information from Wikipedia, and allows asking sophisticated queries on it [1]. DBpedia is also at the core of the Linked Data project [2]. The strength of DBpedia is the capability of answering complex user requests, such as, e.g., "Which European countries have a capital with more than 3 million people in which flows a river longer than 300 km?". The weakness is the difficulty in formulating such queries, due to the complexity of the huge schema that underlies the data. Even though DBpedia has built a "light" ontology for classifying data, the problem still remains extremely difficult. Several approaches to the problem have been developed, over the last few years, to provide user interfaces that attempt to deal (at least partially) with this issue. None of them is fully satisfying.

We identified a subset of the problem, which deals with space-temporal queries, and wrote a user interface, which enables users to perform queries in a simple way, and to get a response in graphical form. Our work is described in the present paper.

This paper is organized as follows: in Section II we shortly present more details of the problem; in Section III we review the attempts to solve it; in Section IV we present our "Spacetime" solution; in Section V we discuss our approach and we draw our conclusions.

II. FORMULATING QUERIES TO DBPEDIA

Which European countries have a capital with more than 3 million people in which flows a river longer than 300 km? Though Wikipedia does not have a page that directly describes this complex set, it contains all the data required for retrieving it. Wikipedia contains information written in natural language, that is very hard for a computer to extract meaning from, but it also contains some tables, called Infoboxes, which present structured information. It is exactly this information that is most valuable source of knowledge harvested by DBpedia, which stores it in a large database. The idea of using this source of information is due to Auer and Lehmann [3]. DBpedia also extracts data from other sources, such as the title of the Wikipedia articles, their categories, interlinks (i.e., the links that connect equivalent articles in different languages), geo-coordinates, redirects (strings used by Wikipedia to identify synonymous terms) and disambiguation (pages that explain the different meanings of homonyms), etc. Also a short and a long abstract are kept for each Wikipedia article.

The DBpedia Knowledge Base (DBKB) contains more than 2.6 million entities [4]. Each entity is defined by a Uniform Resource Identifier (URI), which is described by the common pattern http://dbpedia.org/page/Name, where *Name* is taken from the corresponding Wikipedia article URL. Each entity is composed of a set of Resource Description Framework (RDF) triples. DBKB is composed of around 274 million RDF triples which have been extracted from 35 different Wikipedia language versions. This information is represented using an OWL-based ontology, which was manually created by the members of the community, even though there have been attempts to automatically refine the ontology (see e.g., [5]). The ontology is composed of a large number of classes and properties. The need of having an ontology comes from the fact that the Wikipedia Infobox template system evolved without a central schema for describing entities and their properties. This leads to a situation in which, for instance, the entity Person has an attribute for describing his/her place of birth that can be either "birthplace" or "placeofbirth". The ontology allows centralizing the equivalent property names in a unique property label.

 ISBN: 978-1-61208-355-1

Since the data in DBKB are stored as RDF triples, a natural way to extract information is to perform SPARQL queries. SPARQL is in fact a well-known query language designed specifically to query RDF databases [6]. Although this is in principle enough to solve the problem of extracting information from DBpedia, it is in no way a practical road. First of all, it requires the user to be familiar with SPARQL. This would be equivalent to saying that any Chief Executive Officer can know everything about the company he manages, because s/he only needs to run a SQL query against the company database(s). Although the statement is in principle true, it is not a viable solution.

In order to extract information from a database, one needs to be familiar with its schema. It is necessary to know which entities are represented, which attributes they have and which relations are stored. Hence, to be able to run a query onto DBpedia one needs to be familiar with its ontology, which is composed of 359 classes and 1775 properties. For instance, the class *Person* has properties like *first name, surname, age, birth date, death date, hometown,* etc., while the class *Organisation* has *name, foundation date, hometown, founders,* etc. A deep familiarity with the ontology is needed to be able to write queries. The ontology in itself presents shortcomings. In first place, it was not really "designed", but it is rather the outcome of choices taken by individuals or groups who collaborated in writing the corresponding Wikipedia pages. The ontology is hence partial. For instance, trade fairs do not have an Infobox in Wikipedia pages, hence DBpedia does not have *TradeFair* class in its ontology. Furthermore, it is unbalanced. In fact, some classes have a deeper structure than the others. For instance, the class *Event* has a number of subclasses (*Convention*, *Election*, *FilmFestival*, *MilitaryConflict*, *SpaceMission* and *SportsEvent*). Some are much more developed than others. For instance, *SpaceMission* has 40 attributes while *Election* and *FilmFestival* have respectively 9 and 15 properties. Moreover, some (like *Convention*) are much more general than other (like *SpaceMission*), while many other (such as *TradeFair*) are missing.

Another class of problem is related to the quality of the results rather than to complexity of formulating queries. One of them is related to the completeness of the data. When writing the Wikipedia page, some fields in the Infoboxes can be left blank. This makes sense from the point of view of Wikipedia, which allows progressive evolution of the pages, so that even stubs are allowed. Of course the lack of completeness of the data harms the quality of the query results. As an example, we mention that not all the movies belonging to the *Film* class have the attribute *releaseDate*. At the time of our work this property had a value (in Wikipedia) only on 30943 resources out of the 71715 belonging to the *Film* class (43%). Therefore, any query involving the *releaseDate* attribute will miss over half of the target population, simply because the data are not there.

In the same category also falls the misclassification problem. For example, sometimes a "thing" (represented by a Wikipedia page) is not classified in the correct class but rather in a superclass. For instance, the rock band Pink Floyd is generically classified as *Organisation* rather than as *Band* (*Band* is subclass of *Organisation*). It is worth remarking that when we speak of classification we mean the classification which is provided by the Infoboxes rather that the one given by Wikipedia Categories, which by the way have their own set of problems. For instance, in addition to sharing most of the DBpedia ontology shortcomings we just mentioned, Wikipedia Categories form a non-acyclic graph.

III. PROPOSED SOLUTIONS

Several attempts have been made to help end users extracting valuable information from DBpedia. Here, we do not intend to propose an exhaustive review of all available tools, but rather we arbitrarily choose some examples, as representative of the class of solutions they belong to.

Some are front-ends suitable for exploring the sea of RDF triples, and make it possible to interactively run SPARQL queries. An example is OpenLink Virtuoso. It allows performing research starting from keyword, URI or label. The text search requires the insertion of a text pattern to look for. Then a finder shows a list of entities with the text occurring in any literal property value or label. The entity URI lookup is used inserting entity URI that are recognised by the autocomplete feature of the tool.

Although such sort of tools is certainly useful, it is by no means suited for the end user. Apart the exposition of the technicalities of the query language, the "what can be asked" problem (which requires an understanding of at least a portion of the ontology) is far from being solved.

RelFinder [7] is an example of a different class of tools. It starts from instances instead of from queries. The user specifies two entities, and the system explores the RDF graph to find relations that associate the two instances, and shows the resulting graph to the user. Lodlive [8] is somehow similar. This system allows the the user to choose one instance as a starting point, and then to explore the RDF graph by navigating one of the relations the chosen item is involved in.

Somehow similar is gFacet [9]. It also provides the possibility of navigating the data, but the starting point is now a class instead of an instance. The starting class can be selected by writing a text (part of its name); all classes which contain as a substring in their name the text provided by the user will be shown, and the user will select one. At this point the list of instances is presented to the user. Through class relations, the user can then select a second class, which is linked to the one that was chosen in first place. Selecting an instance in the second class will put a restriction on the first, implicitly solving a query. As an example, if the first class is "Italian Actors" and the relation is "birth place", by choosing Rome in the second class, the box of first class will show all the Italian actors, which were born in Rome. Further restrictions can then be added.

DBpedia mobile [10] takes a different approach, since it starts from a query based on the context. Given the user location, it searches entities, which are nearby geo-located, and shows them on map to the (mobile) user. It is not meant

to explore the whole set of data, but it is rather aimed at providing a location-aware service.

Sgvizler [11] looks at the last part of the process of data retrieval. It provides a way to present the results of a query in graphical form.

Faceted Wikipedia Search (FWS) [12] adopted a faceted search paradigm. This approach enabled users to compose complex questions step by step using facets. A facet is a component shown in the user interface for refining user searches. Facets exploited the properties of an entity to refine the result of a user query. Unfortunately FWS, which had one of the most interesting approaches among the DBpedia-based applications, is now dead since Neofonie, where it was deployed, stopped maintaining the server.

Other (more specific) DBpedia-based applications are listed on a dedicated page on the DBpedia web site [13].

IV. SPACETIME

Spacetime is a tool that aims at making easy for the end user to run a certain subset of queries on the DBKB, and presents the results in graphical form. It considers all the resources in the DBpedia dataset that have at least one spatial and one temporal attribute. This approach allows overcoming the ontology complexity, even though it limits the set of queries that can be formulated. Our requirements for the application were quite simple. We wanted it to be graphically simple and pretty, to be intuitive and simple to use, and able to minimize the knowledge required to the user and its effort in dealing with the interface.

The interface is composed of a control panel for specifying the query parameters, and for saving and loading the resulting maps; a map for visualising the locations of the events found through a search; a timeline for showing the events on the temporal axis (see Figs. 1 and 2).

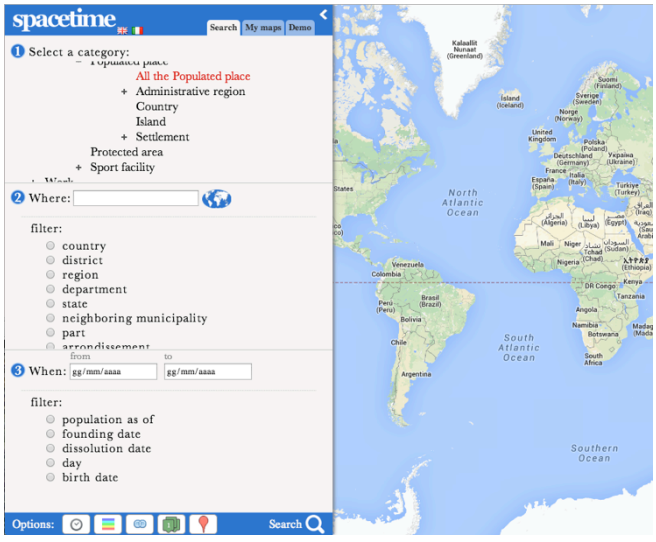

Figure 1. The Spacetime user interface.

To formulate the query, the user has to select "Where", "When" and "What". The "What" comes in the form of what we call "the Spacetime Categories" (SC), i.e., the set of classes that have among their attributes some spatial and temporal data. SC are a (hierarchical) subset of the DBpedia ontology. They are composed of six "top" categories: *Organisation*, *Person*, *Event*, *Place*, *Species* and *Work*. Each of them contains numerous subclasses.

The user starts browsing the SC tree. The user interface is shown in Fig. 1. As it can be seen, there are three sections: Category, Where and When. Once the user selects the suitable class in the upper part, the following sections auto-adapt showing those properties, which have a spatial and temporal dimension. The user can select among those, and put a restriction indicating a geographical place and a time window. When the user fires the request, a SPARQL query is generated and run against the DBKB. The results are rendered on the map in different ways, according to the specific user request, as we shall discuss later.

Behind the scenes, the system performs three types of queries: filter queries, search queries and resources queries.

Filter queries provide the data for the space and time filters in the user interface of the application. These filters are the ones that allow presenting to the user the selectable properties along the space and time dimensions. The time filter selects those attributes which are of data type *xsd:date*. The space filter selects DBpedia properties having a correspondence in latitude and longitude in the Basic Geo (WGS84 lat./long.) Vocabulary. Filter queries are performed as soon as the user selects a category.

Search queries compose the information provided by the user (selected category and properties, and filtering values). An approximation is done in this phase. In fact, the user selects a geographical region by specifying its name, being helped by an auto completion feature. However, geographical data are identified by their coordinates, rather then by logically (or politically) belonging to a geographical entity. Hence, what we do is to use the bounding box of the geographical region specified by the user as a matching filter for the location-dependent properties. This has an obvious problem in terms of precision of the supplied results, as it may include some data belonging to (logically or politically) neighbouring regions.

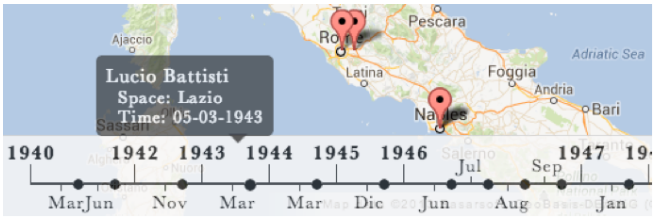

Figure 2. A detail of the shown result set, with a pop-up.

Resource queries are composed in the last phase. Results are graphically shown in a variety of ways. The simplest one (shown in Fig. 2) is in the form of "pins" appearing on the

map, and of dots shown on a timeline. At this point the user can investigate the details; selecting a pin or a dot, information about that specific element of the result set is shown in a pop up.

Typically, the name of the corresponding Wikipedia page is shown, together with the relevant space and time information. Optionally also a short abstract from the corresponding Wikipedia page can be shown. Also links of the resource to Wikipedia and DBpedia are provided, in case the user wishes to obtain additional and more specific information. All this is retrieved by a "resource query", which is a simple SPARQL query asking the DBKB to provide the needed information and the matching abstract, which can be obtained in multiple languages.

Apart of running a query and inspecting the results, the user can save, load, and modify a map. Maps can be saved locally so as to be able to later import them in external resources such as, e.g., a multimedia presentation. Saved maps can be later reimported – e.g., to modify them so as to create a join between the results of two different queries, as we will discuss later.

As we mentioned, Spacetime provides also other types of visualizations. It is possible to create time-sliding animations. Instead of showing all the results at the same time, it is possible to have them ordered along the time axis, and to let them appear in a temporal sequence. As an example, this might show how civilization spread by showing a set of cities in the order in which they were founded.

A second type of presentation shows the events in form of "heat-maps" instead as individual pins (see Fig.3). This is useful in the case one has to show many events: they appear as a density map rather than individually.

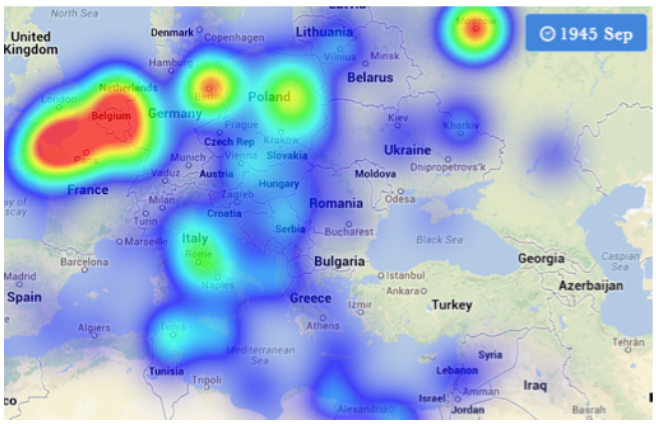

Figure 3. The Europe density map of the military conflict during the Second World War.

Presentations of these two types can be combined, having a density map evolving in time in an animation. One such example is among the demos of the system, which can be seen on the Spacetime web site [13]. It is a query about the battles, which took place during the Second World War. It asks about "the military conflicts occurring in the world between 1939 and 1945 (inclusive)". The density map allows following the evolution of the Second World War, clearly showing how the conflict spread and moved in the world, or, by zooming, in a particular geographical area.

Pin customization allows creating easy-to-understand maps. Pins can be customized by numbering them (in the order they are generated) or by choosing icons among eight categories: colours, numbers, letters, people, culture, events, transportation and sports. Icon colours can also be selected.

Map aggregation allows the creation of more complex maps that include different DBpedia categories. In fact, it is possible to render a map containing resources from categories such as Writer and Book, or different historical events like Election, Military Conflict, and Convention, or to define the historical context when a certain person was born just by aggregating on the map of person birth also a set of historical events. Moreover, the possibility of modifying the icon markers of the resources, allows making the map better in term of meaning and clearness.

An example of pin customization in an aggregated map is shown in Fig.4, which displays the career of the soccer player Zinedine Zidane. Different icons mark the place where he was born, the towns hosting the soccer teams he was playing for, and the places where he won cups. The map is an aggregation of the results of multiple queries.

Figure 4. A map, built as aggregation of the results of multiple queries, shows the career of the soccer player Zinedine Zidane.

All these functionalities were introduced in view of the actual use of the search results: we thought of Spacetime as a tool, which could be used, e.g., when teaching or studying. Hence, it was necessary to think how the user might need to clarify some points, for instance when using those (probably precompiled) results during a lecture, or to incorporate them in a homework.

From a technical point of view, the application is based on five pillars:

• DBpedia: the repository where the knowledge base is kept;

• SPARQL: the query language used for interrogating DBpedia and retrieving the data through a SPARQL endpoint;

 ISBN: 978-1-61208-355-1

• Google Maps: the rendering engine for showing the retrieved data;

• JavaScript: it is at the core of Spacetime, and it contains its application logic. It is responsible of all the interactions between the application components;

• HTML: defines the graphical structure of the Spacetime and the dynamic content of the application.

In more detail, the used technologies are:

•SPARQL Query Language for RDF. Queries are composed by the Javascript engine, and are executed through the SPARQL endpoint;

• JavaScript Object Notation (JSON): the results of the SPARQL queries are produced in this format, which is used for managing the results of the query and for the saving and loading maps;

• Google Maps JavaScript API v.3: the Google Maps API are used for populating a map with the data extracted in the JSON file returned by the SPARQL endpoint;

• JavaScript and JQuery library: the scripting language and its library define a set of functions that are the core of the application. In particular the JQuery library allows the creation of animations inside Spacetime;

•Asynchronous JavaScript and XML (AJAX): this technology is used to have a responsive user interface compliant with the Rich Internet Application paradigm;

• Cascading Style Sheets (CSS): the style sheets language is used for designing the graphical aspect of Spacetime;

• HyperText Markup Language 5 (HTML5): the markup language is used for developing certain part of the application, such as the map saving operation, implemented via the Blob object, and some graphical feature, such as the rounded corners.

V. CONCLUSION

We presented Spacetime, a Rich Internet Application, which deploys the power of Linked Data, and in particular those data, which the DBpedia project gathers from Wikipedia. Our solution does not fully solve the difficult problem of allowing a non-technical user to perform generic queries on the data. However, it provides an easy-to-use interface for a subset of the possible queries. It has been designed to make it possible for a generic user to obtain results that can be embedded in a presentation, or to prepare catching animations.

Spacetime has some limitations. As we mentioned, the geographic selection is made through a bounding box, which might end up in retrieving some data, which are not pertinent to the query. It obviously reflects the weaknesses of Wikipedia and DBpedia in terms of missing information and misclassifications, as discussed in section IV.

The SPARQL endpoint that is used constitutes a single point of failure. If the endpoint has a problem, users cannot perform a search, but they can only load their own maps and work on them. Sometimes errors are generated by the endpoint, as it runs out of its memory pool size. We try to catch these anomalies and to warn the user, but it is not always possible. Unfortunately, the class of problems related to the SPARQL endpoint is out of our control possibilities.

In summary, we think that Spacetime shows in practice the potential of Linked Data, and provides an original solution to part of the problem of building a good and simple interface for the user. Unfortunately, we did not have the time (yet) to run a validation study to support our claims about ease of use and user friendliness.

REFERENCES

[1] S. Auer et al. "Dbpedia: A nucleus for a web of open data." In The semantic web, Springer Berlin Heidelberg, 2007, pp. 722-735.

[2] G. Eason, C. Bizer, T. Heath and T. Berners-Lee, “Linked Data - the story so far.” International Journal on Semantic Web and Information Systems, 5, (3), 1-22, 2009. doi:10.4018/jswis.2009081901 http://dx.doi.org/10.4018/jswis.2009081901 Retrieved Apr, 2014

[3] S. Auer, and J. Lehmann, "What have Innsbruck and Leipzig in common? Extracting semantics from wiki content." In The Semantic Web: Research and Applications, Springer Berlin Heidelberg, 2007, pp. 503-517.

[4] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, U. Becker, R. Cyganiak and S. Hellmann, "DBpedia-A crystallization point for the Web of Data." Web Semantics: Science, Services and Agents on the World Wide Web 7, no. 3, 2009, pp.154-165.

[5] Fei Wu and D. S. Weld, “Automatically refining the wikipedia infobox ontology”. In Proceedings of the 17th international conference on World Wide Web (WWW '08). ACM, New York, NY, USA, 2008, pp. 635-644 DOI=10.1145/1367497.1367583 http://doi.acm.org/10.1145/1367497.1367583 Retrieved Apr, 2014

[6] S. Harris and A. Seaborne, “SPARQL 1.1 Query Language - W3C Recommendation 21 March 2013”. Retrieved April, 2014 from http://www.w3.org/TR/sparql11-query/

[7] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann and T. Stegemann, "RelFinder: Revealing relationships in RDF knowledge bases." In Semantic Multimedia, Springer Berlin Heidelberg, 2009, pp. 182-187.

[8] D. V. Camarda, S. Mazzini and A. Antonuccio, "LodLive, exploring the web of data." In Proceedings of the 8th International Conference on Semantic Systems, ACM, 2012, pp. 197-200.

[9] P. Heim, J. Ziegler and S. Lohmann, "gFacet: A Browser for the Web of Data." In Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW’08), vol. 417, 2008, pp. 49-58.

[10] B. Becker and C. Bizer, “DBpedia Mobile: A Location-Enabled Linked Data Browser”. 1st Workshop about Linked Data on the Web (LDOW2008), Beijing, China, April 2008.

[11] M. J. Skjæveland, “Sgvizler: A JavaScript Wrapper for Easy Visualization of SPARQL Result Sets”. In: 9th Extended Semantic Web Conference (ESWC 2012), workshop and demo proceedings. Heraklion, Crete, Greece, 2012.

[12] R. Hahn et al. "Faceted wikipedia search." In Business Information Systems, Springer Berlin Heidelberg, 2010, pp. 1-11.

[13] Spacetime. Retrieved Apr, 2014 from http://latemar.science.unitn.it/spacetime/spacetime.html, also reachable from the “DBpedia applications” web site, http://wiki.dbpedia.org/Applications, demos visible from http://latemar.science.unitn.it/spacetime/usecase.htm

 ISBN: 978-1-61208-355-1