A Visual Query Builder for DBpedia*

Dimitrios Soumis[1], George Stamoulis[1(✉)], and Manolis Koubarakis[1,2]

[1] National and Kapodistrian University of Athens, Greece
{cs2200018,gstam,koubarak}@di.uoa.gr
[2] Archimedes/Athena RC, Greece

Abstract. In the last years we have seen a huge effort to extract structured content from the information created in various Wikimedia projects. This wealth of information is acquired with the use of linked data. One such example is DBpedia, that allows to perform complex searches on Wikipedia datasets as well as linking them to data from other sources (e.g., data from a domain such as agrifood). While the data are available publicly, knowledge of SPARQL and ontologies is mandatory to search and analyse them. This paper presents an application for Android devices, aimed at familiarizing the average user to generate successful searches and extract desired data from DBpedia, without pre-existing knowledge in semantic web technologies.

Keywords: linked data · DBpedia · visual query builder · knowledge graph · Android.

1 Introduction

As the world moves further into the digital age, smartphones are becoming the primary access points to media and information for both advanced and novice users [3]. People need fast and abundant results from simple searches in the shortest possible time using quick access systems without having to use a computer to satisfy their needs and without prior sophisticated background in order to use a system [2]. In this paper we present an Android application that enables non-experts to create SPARQL queries over DBpedia [1], for simplified information retrieval. This is especially useful to non-expert users from other domains (such as agrifood) that are not familiar with semantic web technologies. The DBpedia Visualizer offers a graphical user interface and key word search over DBpedia resources, that allows the creation of a visual graph based on user feedback and automatically translates it to a SPARQL query in order to retrieve the information for the user.

* This work has received funding from the project STELAR (101070122), under the European Union's Horizon Europe research and innovation programme. This work has also been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

2 Related work

Traditional SPARQL querying can be challenging for users unfamiliar with the syntax or structure of RDF data. To address this, RDF Explorer [4] offers a visual interface that allows users to construct queries intuitively, lowering the barrier for querying RDF data. The tool aims to make the Semantic Web more accessible to a broader range of users, particularly those without extensive technical expertise. RDF Explorer enables users to query and retrieve specific information from RDF datasets using SPARQL, a query language tailored for this purpose. This makes it ideal for extracting insights from interconnected data sources, such as knowledge graphs or web-based linked data repositories. This is achieved by using a visual query language in which users can express queries on graphs through simple interactions. The proposed visual query language is formulated in terms of a visual query graph. Visual query graphs are designed as a visual metaphor for the basic SPARQL graph patterns, where the translation is therefore mostly direct and natural. This visual approach enables users to understand the structure of their data and build complex queries without needing to write SPARQL code manually. RDF Explorer also incorporates features that guide users through the process of query formulation, such as auto-completion, which further enhances its usability. This makes it a powerful tool for both novice and experienced users working with linked data and knowledge graphs.

3 The DBpedia Visualizer

DBpedia Visualizer is an Android application designed to support non-expert users in the creation and execution of queries over DBpedia utilizing a graphical interface. The system is based on the tool RDF Explorer, but runs as a native Android application to make it more accessible to the broaded audience, that could lead to more widespread adoption of RDF-based data analysis in various fields, including data science, artificial intelligence, and agrifood.

The application utilizes the public SPARQL endpoint of DBpedia [3] to extract information through HTTP requests, based on user input. The user first provides a search term used to pose an HTTP request to the DBpedia lookup service [4]. The results are presented as a list to the user to select an entity of interest as a starting point. User can expand on the properties of the entity by tapping on it and select a value. Both the property and its value are stored in application memory and drawn on the interface. The property is drawn in the box of the original entity and the value in a new one. The connection between them is represented with by an arrow. In order to introduce variables, users can double-tap on any number of entities to convert them into variables.

All HTTP requests are implemented with OkHttp client and are used to pose to DBpedia the automatically generated SPARQL queries based on the graph

[3] http://dbpedia.org/sparql
[4] https://www.dbpedia.org/resources/lookup/

the user has constructed. Results from DBpedia are retrieved in XML and JSON format and are integrated in the application using a table format.

4 Demonstration

In this section we demonstrate how to use the application in a scenario to retrieve notable ideas of people that influenced *Albert Einstein*.

The user begins by entering a search term into the designated field within the application. The application then automatically generates an HTTP request directed to the DBpedia Lookup service. The DBpedia Lookup Service is particularly useful for resolving entities, meaning it helps users find resources within the DBpedia database by offering keyword-based search. This service can be used to look up entities, terms and concepts based on the input, utilizing a semantic search mechanism. The service responds with search results in XML format, which the application displays in a list format using a ListView, ensuring a user-friendly presentation. The user selects an entity from the list, and this selection is saved in the application's memory and displayed in the initial user interface. In our demonstration we provide the name of the physicist *Albert Einstein* and select it in the results to create the starting node of our graph as shown in Figure 1.

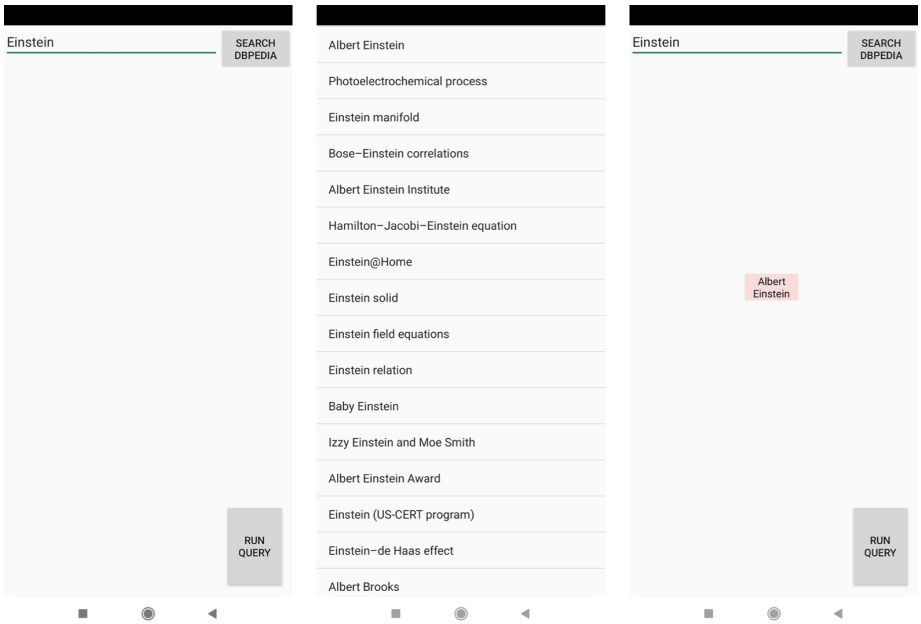

Fig. 1. Search DBpedia for Albert Einstein

The user then performs a long-click on the desired entity displayed in the interface. The application detects this long-click, generates a corresponding SPARQL query, and sends an HTTP request to the DBpedia API to retrieve the entity's properties and values. The results are returned in XML format and are presented by the application in an expandable list format. The user can then expand any property by tapping on it and select a value. Both the selected property and value are stored in the application's memory and displayed, with the property linked to the original entity and the value depicted as an independent entity. The connection between them is represented by an arrow, visually connecting the original entity to its value. In our scenario, we can view all the properties of the resource *Albert Einstein* and select the property *influencedBy* to choose one of the values to create a new node in the application. Now using the new node, we can view again its properties and select the *noteableIdea* attribute and one of the values as shown in Figure 2.

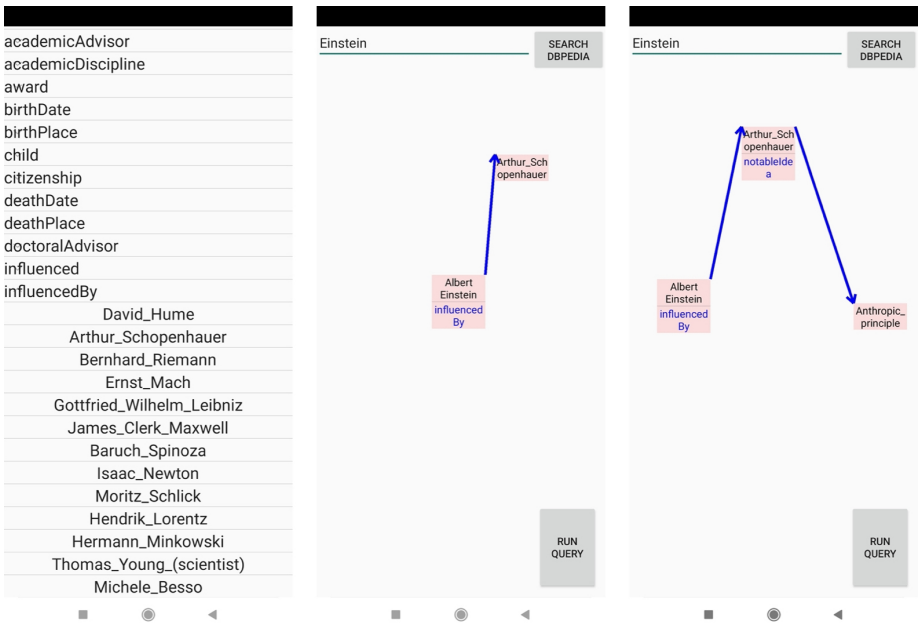

Fig. 2. Select properties and values to build our graph

In addition, the user can rearrange entities to create a desired visual layout and repeat the process as many times as needed. The user can also double-click on entities to turn them into variables, which will be used in the final SPARQL query. The application detects this action, replaces the entity text with a variable name like *?varX* and updates the diagram, with *X* representing a numbered sequence of variables. Utilizing this feature, we can tap on the

nodes of the influencer and his notable idea to change them into variables. After designing the diagram, the user clicks the *Run Query* button, and the application automatically formats the final SPARQL query based on the diagram as shown in Figure 3. The query is sent to the DBpedia API, and the application receives the results in XML format, displaying the SPARQL query and the retrieved variable values in a table for clear and user-friendly presentation.

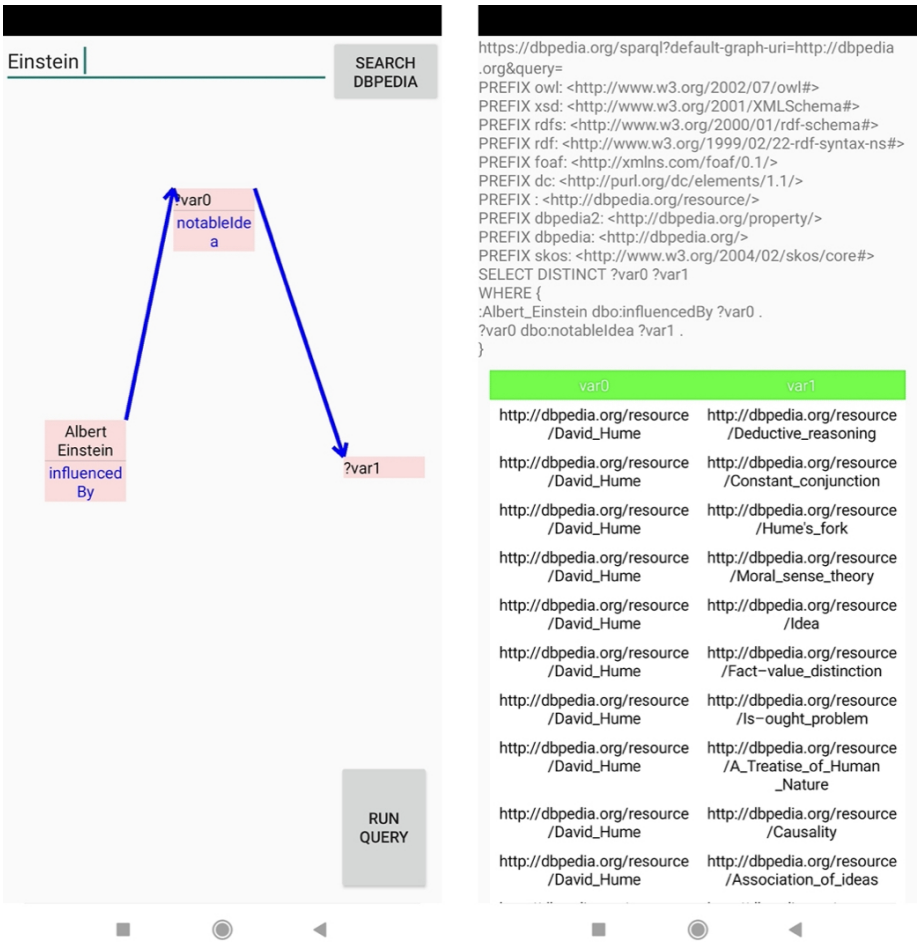

Fig. 3. Change nodes to variables and run the query

5 Conclusion and Future work

In this paper we demonstrated the DBpedia Visualizer, that offers a graphical interface for non-expert users to build SPARQL searches for streamlined information retrieval over DBpedia. For future work we plan to enhance both the functionality and interface design of the application. We will add an autocomplete service to the search field based on the available ontology of DBpedia. Since search results contain URIs we would like to offer access to these resources from DBpedia by redirecting to the device's browser. Furthermore, we plan to support more SPARQL functionality such as filters and aggregates to enable the creation of more complex search queries.

References

1. DBpedia: About dbpedia. https://www.dbpedia.org/about/
2. Hoelzle, U.: The google gospel of speed. Think with Google, https://www.thinkwithgoogle.com/future-of-marketing/digital-transformation/the-google-gospel-of-speed-urs-hoelzle/ (2012)
3. PECB: How smartphones are 'killing' pcs. https://insights.pecb.com/smartphones-killing-pcs/ (2018)
4. Vargas, H., Buil-Aranda, C., Hogan, A., López, C.: Rdf explorer: A visual sparql query builder. In: The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I. p. 647–663. Springer-Verlag, Berlin, Heidelberg (2019)