ForestQB: An Adaptive Query Builder to Support Wildlife Research

Omar Mussa[1,3,*], Omer Rana[1], Benoît Goossens[2], Pablo Orozco-terWengel[2] and Charith Perera[1]

[1]*School of Computer Science and Informatics, Cardiff University, United Kingdom*
[2]*School of Biosciences, Cardiff University, United Kingdom*
[3]*College of Computing and Informatics, Saudi Electronic University, Saudi Arabia*

Abstract
This paper presents ForestQB, a SPARQL query builder, to assist Bioscience and Wildlife Researchers in accessing Linked-Data. As they are unfamiliar with the Semantic Web and the data ontologies, ForestQB aims to empower them to benefit from using Linked-Data to extract valuable information without having to grasp the nature of the data and its underlying technologies. ForestQB is integrating Form-Based Query builders with Natural Language to simplify query construction to match the user requirements. (Demo is available at https://iotgarage.net/demo/forestQB)

Keywords
Linked-Data, Visual Querying, SPARQL, Query Builders

1. Introduction

Publishing the data as a Linked-Data using Semantic Web technologies is beneficial for machine learning as well as information retrieval [1]. While the data will be easily accessible by machines, Humans can also benefit from accessing the data by using a query language such as SPARQL, which is the recommended query language for querying RDF triplestore.

In the field of Bioscience and Wildlife conservation, researchers tend to collect data using various sensors such as temperature, location and speed. Therefore, hundreds of gigabytes were collected over the years that would be extremely valuable if stored as a knowledge graph in an RDF triplestore. However, users usually feel intimidated to use Linked-Data as they are obliged to understand SPARQL and the underlying data structure [2]. In order to encourage these Bioscience researchers to adopt semantic web technologies in their field, it is essential to present a toolkit that fulfils their requirements to freely access the data store without the need to worry about its underlying technology.

In this demo, we introduce ForestQB, a tool that aims to facilitate the knowledge extraction out of the RDF triplestores by allowing the researchers to construct their query visually. The

The 21st International Semantic Web Conference, October 23–27, 2022, Hangzhou, China
*Corresponding author.
mussao@cardiff.ac.uk (O. Mussa); ranaof@cardiff.ac.uk (O. Rana); goossensbr@cardiff.ac.uk (B. Goossens); orozco-terwengelpa@cardiff.ac.uk (P. Orozco-terWengel); pererac@cardiff.ac.uk (C. Perera)
0000-0001-8614-6550 (O. Mussa); 0000-0003-3597-2646 (O. Rana); 0000-0003-2360-4643 (B. Goossens); 0000-0002-7951-4148 (P. Orozco-terWengel); 0000-0002-0190-3346 (C. Perera)

tool provides a high level of abstraction for users by supporting a Form-based interface with an integrated conversational AI to support natural query construction.

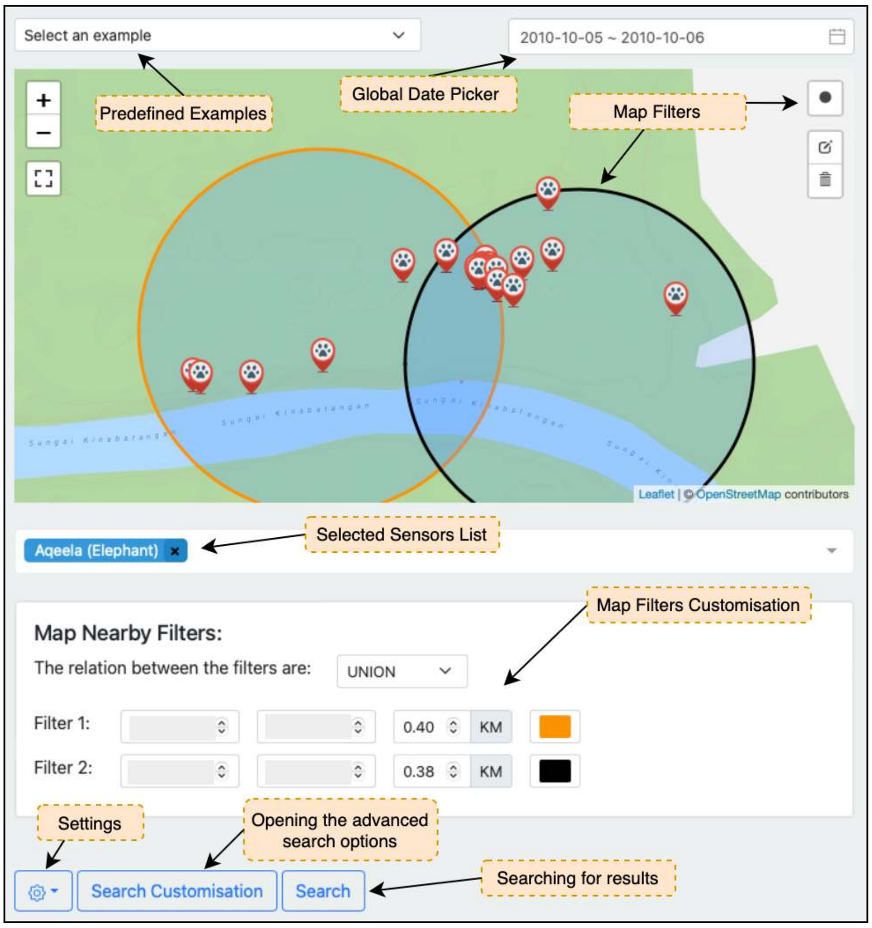
2. ForestQB Features

ForestQB is a web application that works on the browser to query and explore RDF triplestores that were exposed as a SPARQL endpoint. ForestQB was implemented based on the requirements collected by reviewing the current progress in the field and interviewing experts in the Bioscience field (the stakeholder). The recent features of the tool can be outlined as follows:

- **General overview**: the initial interface provides a basic search functionality by allowing the user to select sensors (observable properties) as a starting point to explore the data. The interface contains a list of sensors loaded from the endpoint when the toolkit was initially loaded. The sensors' data are fetched by sending a SPARQL query that can be adjusted from the settings to customise the tool if querying different endpoints. Hence, The user can click the search button to start loading the results related to that sensor without requiring additional steps. In addition, the interface includes a map with geospatial filtering capabilities by drawing circles directly on the map. It also contains an upper date picker that will strict the results to a general date range. The upper left side includes a list of predefined examples that will populate all required fields to promote learning by example. The primary goal is to have a user-friendly interface that will support the user to explore the data quickly.
- **Detailed querying**: once the users decide to have more advanced querying, they can click on the search customisation to display all the detailed querying features. The interface will display all the linked properties for the chosen sensor with an additional sublist allowing them to manually add filters to each property. The filters that the user can add will vary based on the property XSD data type. For example, a string will include Contain, Match and Regex filters. Also, the same filters will act differently based on the property data type. For example, the number and date both have a Range filter, but while the number will show a numeric text field, the date will show a date picker. Furthermore, the user can choose to hide the entity from the results or mark an entity as optional to be ignored if it does not exist.
- **Conversational AI**: the ForestQB includes a component that will display a chatbox where the user can express their query in natural language to populate the corresponding fields automatically. In addition to constructing queries, it can help the user to inquire about the underlying data structure. For example, the chatbot can answer simple questions such as "What are the sensors?", "What is Aqeela?" and "Where is Aqeela?" as shown in Figure 2. The user can completely hide the form-based interface to use the chatbot as a stand-alone query builder tool to query and retrieve the results.

3. Our SPARQL endpoint

The used dataset contains sensitive historical data collected by bioscientists that were modelled as Linked-data. Thus, our SPARQL endpoint is privately available. The dataset populates



Map Nearby Filters:

The relation between the filters are: UNION

Filter 1: 0.40 KM 🟧

Filter 2: 0.38 KM ⬛

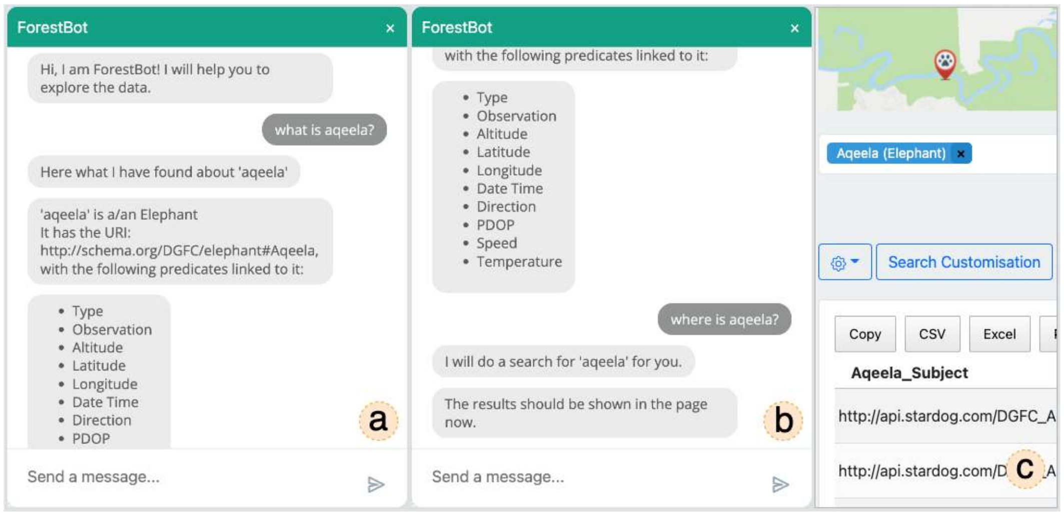
Figure 1: Overview of a simple query to retrieve data of a particular animal within a specific date and location. The Longitude and Latitude of the animals were hidden due to the sensitivity of the data.

Forest Observatory Ontology (FOO) [1], a novel ontology that describes wildlife data generated by sensors. It was developed by reviewing the state-of-art and reusing entities from different mature and self-contained ontologies. FOO arose from the efforts to review Open Data Observatories [3] in order to build the Forest Observatory. The data integrates multiple ontologies to define its underlying structure, including SOSA [4] ontology. Hence, ForestQB initially relies on SOSA to get all connected sensors and their observable properties, which means it will potentially work with another endpoint if it applies SOSA.

4. Demonstration

The demo will illustrate the ForestQB interface and explain its functionality. Figure 1 shows an overview of the tool to create a simple query using map filters, demonstrating the basic search with a limited number of filters to apply. The "customisation search" button is where the user

[1]https://naeima.github.io/Forest-Observatory-Ontology/


ForestBot

Hi, I am ForestBot! I will help you to explore the data.

what is aqeela?

Here what I have found about 'aqeela'

'aqeela' is a/an Elephant
It has the URI:
http://schema.org/DGFC/elephant#Aqeela,
with the following predicates linked to it:

- Type
- Observation
- Altitude
- Latitude
- Longitude
- Date Time
- Direction
- PDOP

Send a message...

ForestBot

with the following predicates linked to it:

- Type
- Observation
- Altitude
- Latitude
- Longitude
- Date Time
- Direction
- PDOP
- Speed
- Temperature

where is aqeela?

I will do a search for 'aqeela' for you.

The results should be shown in the page now.

Send a message...

Aqeela (Elephant)

Search Customisation

Copy CSV Excel

Aqeela_Subject
http://api.stardog.com/DGFC_A
http://api.stardog.com/D

Figure 2: Overview of the conversational AI (ForestBot). (a) Retrieving information about “Aqeela” sensor. (b) Constructing a query to find “Aqeela”. (c) The result will then be reflected on the Query Builder as if the user has selected them, and the search process will be triggered.

can apply a more advanced search. The customisation is always hidden until the user decides to display it to allow a more neat look and hide the complexity of the interface. The results will be presented in a tabular format to reflect the user query.

The conversational AI can work as an assistive tool to the Form-based builder or as a standalone query builder. The user can type their query in natural language, and it will be reflected on the interface. For example, a question such as “Where is aqeela?” will select the correct sensor from the sensors list and trigger the search process (see Figure 2).

5. Technical Details

The ForestQB interface was designed after reviewing the current state-of-the-art and gathering requirements from stakeholders. The design has numerous layers and components, each of which serves a unique purpose with distinct technical details behind it. Thus, this section briefly discusses some of the technical aspects of ForestQB.

- **Web Application**: Plain JavaScript and Vue.js (vuejs.org) Framework have been used to build the tool as a Single-page application. The tool will generate a JSON object that will be identical to the tool choices. Once the user clicks search, this object will be sent to the conversion engine (as an AJAX request) to be converted to SPARQL and sent back to the tool to be used for querying the endpoint. The separation of the SPARQL conversion mechanism from the tool will allow it to be shared between the ForestQB and the conversational AI. Both ForestQB and conversational AI share a centralised store using Vuex (vuex.vuejs.org).

- **Conversational AI model**: The conversational AI is split into two pieces: the front-end and the classification model. The front-end is part of the main web application components.

However, the Natural Language Understanding (NLU) model has been built using RASA (rasa.com) framework. Thus, all of the logic behind the NLU lies under a different web server that is powered by RASA. The front-end will send all user messages to the chatbot server to understand the user's intent. Most of the actions (responses) are implemented within the web front-end. The conversational AI will adjust the centralised store based on the classified intent and entities to modify the JSON Query object. Then, it will trigger the search process. The chatbot will either answer simple questions about the data schema or retrieve the results when the user asks.

- **Map Filters**: ForestQB is offering map filters to narrow down the results. The implemented map uses the Leaflet (leafletjs.com) library to visualise and allow the map's main functionality. Thus, drawing circles on the map allows the user to create nearby filters. Once a circle is drawn, its coordinates and parameters are saved on the JSON Query object and then translated into SPARQL. In addition, the user can define the relationship between the circles as intersections or unions (as shown in Figure 1). As an adaptive feature, the map (GeoSPARQL) filters are not applied to data that do not include Longitude and Latitude. However, the other type of filters will still be valid.

6. Conclusions

This paper has briefly introduced the ForestQB, a toolkit that aims to assist bioscientists in querying Linked-Data. The tool supports generating the query by mixing the conversational AI with the Form-Based query builder. The ForestQB is still a work in progress, as we plan to improve its conversational AI to handle more complex sentences. In addition, it will support more visualisation options based on our future user study.

References

[1] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, A. Bielefeldt, Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph, in: 17th International Semantic Web Conference, volume 11137 LNCS, 2018, pp. 376–394.

[2] P. Warren, P. Mulholland, A Comparison of the Cognitive Difficulties Posed by SPARQL Query Constructs, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 12387 LNAI, Springer International Publishing, 2020, pp. 3–19.

[3] N. Hamed, O. Rana, P. Orozco-terWengel, B. Goossens, C. Perera, Open Data Observatories: A Survey, Technical Report, Cardiff University, 2021. URL: https://orca.cardiff.ac.uk/id/eprint/140048/14/NaeimaHamed_26_MAR_2021_V5.pdf.

[4] K. Janowicz, A. Haller, S. J. D. Cox, D. Le Phuoc, M. Lefrançois, SOSA: A lightweight ontology for sensors, observations, samples, and actuators, Journal of Web Semantics 56 (2019) 1–10. URL: https://www.sciencedirect.com/science/article/pii/S1570826818300295.