RailGB: Using Open Accessibility Data to Help People with Disabilities

Yunjia Li, E.A. Draffan, Hugh Glaser, Ian Millard, Russell Newman, Mike Wald, Gary Wills, and Magnus White

School of Electronics and Computer Science,
University of Southampton, UK
{yl2,ead,hg,icm,rn2,mw,gbw,mpw}@ecs.soton.ac.uk

Abstract. With the fast growth of Linked Data, many datasets have been made available for public access. However, with an increased number of elderly and people with disabilities becoming technologically knowledgeable, it has been noted that few applications have been developed for this population group using Linked Data. In this paper, we present RailGB, a linked-data driven mobile web application to provide dynamic searches for accessible underground stations in London. RailGB may also prove useful for users such as those with heavy loads or baby buggies. Data in RailGB is collected from different datasets and are mashed-up within a user-friendly mobile web interface. A Query Agent is developed to automatically build SPARQL queries linked to different user requirements. Data around access requirements tends to be detailed on a wide range of static web sites and rarely offers options when visiting particular places. RailGB offers a fundamental shift in approach by reacting to the users' needs whilst travelling with choices of access to a range of stations around the particular area to be visited using Linked Data. Further research is required to provide a wider range of information as more high quality open data becomes available.

Keywords: linked data,accessibility,semantic web,open accessibility data

1 Introduction

London underground offers maps with different accessibility information. However, the traditional underground maps have at least two issues. Firstly, it is not straight-forward to find out the nearest accessible station as the user's current location is not marked on the map. Applications like Google Map can trace users' current location on mobile devices, but there is no accessibility information for the nearby stations on the map. The second issue is that the accessibility information on the maps is not published in machine readable format, no matter whether it is a paper map or electronic map in PDF or image format. Accessibility data is not available for applications to offer intelligent services to those with disabilities. With the fast development of Linked Data [1], datasets in various domains have been published for public access in machine-readable format.

The growing number of datasets have brought opportunities to innovative applications which use open data from different resources. However, we note that few applications have been developed to benefit the everyday life of people with disabilities.

In this paper, we present a mobile web application named RailGB, which makes uses of the machine-readable data related to the accessibility facilities in London underground stations. The goal of this application is to use the accessibility-related open data, i.e. Open Accessibility Data (OAD), to help people with various disabilities, such as wheelchair users and visually impaired persons, to locate the nearest accessible underground stations. This application can also help other people with special requirements, such as adults taking a baby or heavy loads, by finding the stations with lifts, escalators, etc.

While RailGB is a location-based, Linked Data driven application at the backend, we keep the user interface simple and easy to use. All the data used in RailGB is real-world data published in different places. The focus is given on using the accessibility data that is semantically described in these datasets. The Query Agent in RailGB transfers the disabilities' requirements to SPARQL queries, which hides the complexities of data collecting and mashup from end users. In the rest of this paper, we will firstly have a walk-through of the application in Section 2. System design and the semantic features will be presented in Section 3. Section 4 and 5 discuss the result of evaluations and the lessons we learned. Finally, Section 6 gives the conclusions.

2 RailGB Walk-through

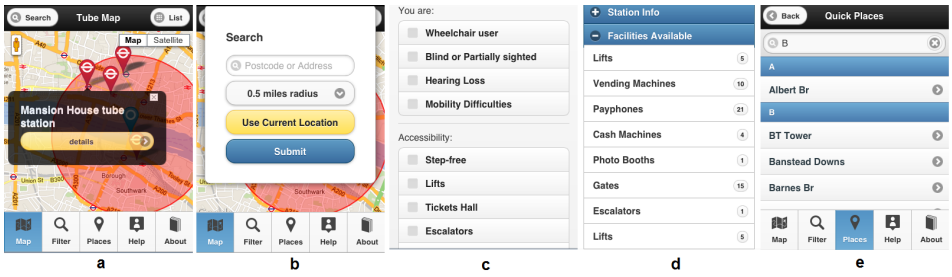

Fig. 1. RailGB Screenshots

RailGB currently is a mobile web application, where users can search the underground stations around a certain place in London, filtered by required facilities. On the front page, users can search for places by name, postcode or their current location, and RailGB will display the underground stations meeting the requirements within a certain radius on a Google Map (Figure 1a and 1b). To filter the stations with the facilities available, users can set their preferences on

the filter page (Figure 1c). These facilities include lifts, escalators, help points, etc. Users can also choose their disabilities and the application will automatically check the desired facilities.

On the map, users can click the marker of the station to find out the name of the station (Figure 1a). A list of all stations in the search results can be found on the list page. The detailed page (Figure 1d) displays more information of the station, which include all the facilities either available or not available. The abstract and depiction image on this page is fetched from the DBpedia SPARQL endpoint[1]. In order to simplify the search experience, we also provide a list of quick places identified by their names (Figure 1e), such as London Eye, British Museum, etc. Users could select the places they are looking for and the application will automatically search the nearby underground stations.

3 System Design and Semantic Features

Figure 2 demonstrates the architecture of RailGB. We firstly need to find out the datasets which are publicly available for the functions described in Section 2, especially the data about accessibility facilities. If the data is not in machine-readable format, we need to convert it to structured data and re-publish it via SPARQL endpoint. In order to connect the re-published dataset into the Linked Data Cloud, Co-reference Resolution Service (CRS) sameAs.org [2] is used to interlink the new dataset with DBpedia. When all the data is available, Query Agent makes SPARQL queries based on users' search requirements and Data Mashup module presents the data in mobile web user interface.

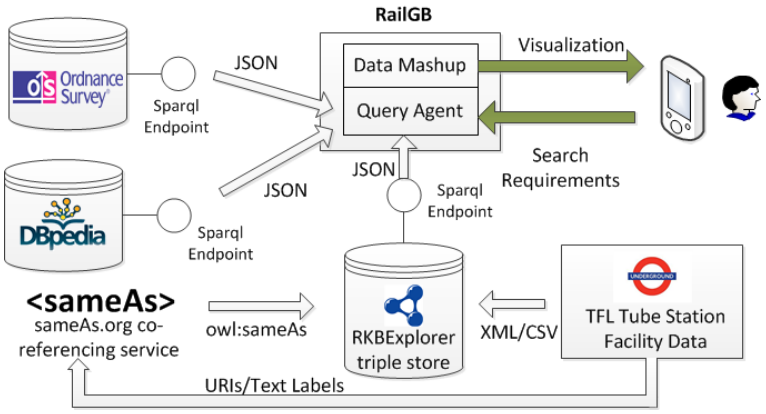

Fig. 2. RailGB Architecture

[1] `http://dbpedia.org/sparql`

3.1 Datasets and Interlinking

Generally, RailGB needs data for (1) the availability of accessibility facilities in each underground station; (2) the location of each underground station in form of longitude, latitude or postcode , and other details of the stations; (3) and the name of visitor attractions in London and their locations.

Some datasets have been published containing basic information of London underground stations, such as "Tube stations and lines"[2]. However, the accessibility data is mainly published at "Station Facilities Data" (in XML format) and "Tube Station Accessibility Data" (in Excel spreadsheets) in Transport for London (TFL) API[3]. We have published them as Linked Data, so that other applications can easily access these datasets in machine-readable format.

As far as we know, there is no ontology that we can re-use to describe the facilities of the underground station. So we create Tube Facility (TF) ontology[4] according to the XML structure of the TFL station facilities API and we have identified 14 kinds of facilities. Each station in TFL is identified by a number (for example, 1000254 is Waterloo tube station), so we mint our own URIs for each station in RailGB in the following format *http://www.railgb.org.uk/id/tube/stationid*.

We wrote a script to dump the station facilities and station locations from TFL API into RDF/N3 format, hosted by RKBExplorer[5]. We also reference the Tube Station Accessibility Data for lifts information in case the Station Facilities Data is not up-to-date. As there is no step-free information from TFL API, we had to manually get this data from Step-free Tube Guide[6] and TFL website. The RailGB dataset is exposed through SPARQL endpoint[7]. The newly created RailGB dataset is standalone, so we need to link it to the Linked Data Cloud and obtain more information about each station, which is available in other datasets. We put equivalent assertions (owl:sameAs) between RailGB station URIs and DBpedia URIs by checking the stations' names using sameAs.org service and Sindice [4]. The sameAs.org service can automatically return co-referencing URIs from DBpedia, Freebase[8], etc, but some manual work is still needed to identify the best URIs, where the pictures and descriptions of the stations can be dereferenced.

For the quick places function mentioned in Section 2, Ordnance Survey geo-location dataset[9] is used to retrieve the places with their names and postcodes. When users select a place from the list, RailGB will send a query to the OS SPARQL endpoint and find out the postcode of this place.

[2] `http://publishmydata.com/datasets/transport-for-london`
[3] `http://www.tfl.gov.uk/businessandpartners/syndication/16493.aspx`
[4] `http://www.railgb.org.uk/ns/2012/9/tubefacility.owl`
[5] `http://oad.rkbexplorer.com`
[6] `http://www.tfl.gov.uk/assets/downloads/step-free-tube-guide-map.pdf`
[7] `http://oad.rkbexplorer.com/sparql`
[8] `http://rdf.freebase.com`
[9] `http://data.ordnancesurvey.co.uk/`

3.2 Querying Agent

On the filter page, we allow users to specify requirements related to their disabilities. The Query Agent will automatically select the required facilities in the query. To offer this user-friendly function, RailGB needs to seamlessly translate a disabilities' requirements of using underground stations to the SPARQL queries. As a simple example, a blind or partially sighted person may need an assistant to buy tickets, go to the platform and get off the train, which means help points must be available in ticket halls. According to the design of TF ontology in this application, the following assertion should be defined as true:

$$Station(?station) \wedge hasHelpPointsInTicketHalls(?station, ?th) \wedge facilityAvailability(?th, ?tha) \wedge (= true)(?tha) \Rightarrow isBlindAccessible(?station, true)$$

The Query Agent uses two approaches to achieve this function. One approach is to hard-code different conditions on the left-hand side into SPARQL queries, so that the query changes along with the users of different disabilities. This method is easy to realise, but when the conditions involved too many variables, the query may become very complex and the response time will become unacceptable. For example, for persons with mobility difficulties, lifts must be available or both escalators and staff must be available, so a member of staff can help you take escalators safely. The other way is to use the CONSTRUCT query to put new triples into the triple store according to the left-hand side conditions. In this example, the CONSTRUCT query is[10]:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tf: <http://www.railgb.org.uk/ns/2012/9/tubefacility.owl#>
CONSTRUCT { ?station tf:isBlindAccessible "true" } WHERE {
  ?station tf:hasHelpPointsInTicketHalls ?ht .
  ?ht tf:facilityAvailable ?htAvailable .
  FILTER (?htAvailable=true) .}
```

This method is also easy to implement and it saves querying time compared with the first approach. But the first approach is still useful in RailGB in that some of the users may not have disabilities and we should allow them to filter the stations by facilities instead of selecting the disabilities directly.

4 Initial Evaluation

To evaluate the use of the application in real situations, we have done some initial evaluations in London. One of our team members started a journey from Amberley train station in Sussex and she was going to Victoria train station, British Museum, Lincoln's Inn and the Court of Justice. Our goal is to check that if RailGB can locate the correct stations nearby with correct facility information

[10] The endpoint in RKBExplorer is SPARQL 1.0

of step-free, help points, lifts and escalators. The device we use is Safari browser in iPhone 4S, iOS 6.

As a result, we used Victoria, Holborn, Chancery Lane and Temple tube stations following the planned journey. The tester found out that the underground stations were correctly located. The response time for the query is acceptable when the 3G connection was good, but sometimes, it could take 20 seconds or even longer to get the result. The quick places are useful when planning the journey outside of London, while in London, it is more efficient to use the current location. We find that the main issue of the application is the accuracy of the facilities information from the data source. In the four underground stations we have checked, step-free, lifts and escalators information is accurate. The original Station Facility Data indicates that none of the four stations have help points, but actually they have some help points in ticket halls, along corridors or on platforms. Our main data source, Station Facility Data, is generated in the year of 2010, so some data is out-of-date. Unlike step-free, lifts, we did not use Tube Station Accessibility Data to improve the accuracy of help points information and that is why many newly created help points are missing.

5 Lessons Learned

In the development of RailGB, we have identified several challenges and lessons. Firstly, it is very difficult currently to find a publicly-available, high quality and structured dataset containing accessibility data. Some websites have rich accessibility data about public transport in UK, such as DisabledGo[11] and DirectEnquiries[12]. Unfortunately, they are not published as Linked Data. In current RailGB dataset, some information is still missing. For example, 58 out of 301 stations do not have any facility data at all and only 237 stations have full information. In addition, some information in the original Tube Facilities Data is not well formatted. For example, the value of "Vending Machines" sometimes is a number and sometimes it is a string like "1 snack, 2 drink", which makes it difficult to model the relationships in the ontology.

Another lesson we have learned is that it is still difficult to link the requirements of disabilities with the facilities via ontology reasoning and inference. One aspect of the problem lies in the complexity of rules for the reasoning based on the real world situations. The assertions for *isBlindAccessible* in Section 3.2 is not complex, but for some disabilities, rules are still unclear as to what facilities are required and likely to change on an individual basis. For example, a station has lifts does not infer that the station is step-free, as there might be stairs from street level to the lifts. Some stations are step-free in one direction but not the other. So the complexity of the real world problem decides that it is difficult to infer which station is step-free from the data we can get. The other aspect of the problem is the technology support for OWL inference. Compared with the two approaches we use in Query Agent, a better solution should be adding SWRL

[11] `http://www.disabledgo.com/`

[12] `http://www.directenquiries.com/`

rules [3] into the TF Ontology or using Jena rule engine[13], so that the triple store can infer new triples based on the rules in real time.

6 Conclusion and Outlook

In this paper, we present the RailGB mobile web application, which uses OAD to help people with disabilities to locate the nearest underground stations with certain facilities in London. We collected data from different resources, such as TFL API, Ordnance Survey, DBpedia, etc, and mashup the data in easy-to-use mobile web interface. The Query Agent makes SPARQL queries to find the stations according to users' requirements. The main lesson we have learned is that few structured and high quality OAD datasets are publicly available, which affects the accuracy of the service. Another challenge is semantically bridging people's requirements to the facilities in the RaiGB dataset.

The RailGB application is available online for everybody to use[14]. As a future work, we are planning to add facilities data about railway stations across UK to RailGB. We hope that more accessibility data could be made available, in order that more useful applications can be built for people with disabilities. Information such as toilets, cash machines, car parks are also useful for anyone with some special needs at times. The big vision of OAD is not only linking accessibility facilities in transportations, but also other public places, such as universities, museums, hotels and restaurants. OAD will be part of the Linked Data Cloud and well connected to other datasets.

References

1. Berners-Lee, T.: Linked Data - Design Issues (2006), http://www.w3.org/DesignIssues/LinkedData.html
2. Glaser, H., Jaffri, A., Millard, I.: Managing co-reference on the semantic web. In: WWW2009 Workshop: Linked Data on the Web (LDOW2009) (April 2009)
3. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. Syntax 21(May), 79 (2004), http://www.w3.org/Submission/SWRL/
4. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudr-Mauroux, P. (eds.) The Semantic Web, Lecture Notes in Computer Science, vol. 4825, pp. 552–565. Springer Berlin / Heidelberg (2007)

Appendix

Table 1 addresses the minimal (M) and additional (A) requirements of the Challenge Criteria. We give a quantitative rating from 1 (low rating) to 5 (high rating) for each criteria.

[13] http://jena.apache.org/documentation/inference/index.html
[14] http://m.railgb.org.uk

Requirement	Rating	Explanation
(M) End-user Application	5	RailGB is designed for the real need of disabilities as well as normal people.
(M) Sources - diverse ownership - heterogeneous - real world	5	The data is collected from TFL API, Ordnance Survey and DBpedia. The content of the datasets is obviously different from each other, even though they are related to each other. All the resources are real-world data.
(M) Meaning of Data - represented by SW - data manipulation - alternative tech	5	The facilities and their accessibilities to different disabilities are model by Tube Facility ontology. All the data used in RailGB are in RDF format and the datasets are interlinked using co-referencing service sameAs.org. The co-referencing URIs are also used in data mashup via SPARQL endpoint. These functions are not possible for alternative technologies.
(A) Web interface	5	The mobile web interface is easy to use and can be used across different devices.
(A) Scalable	5	RailGB is built on diversity of datasets, which are integrated following Linked Data principles. So it is highly scalable.
(A) Rigorous evaluation	4	We have done some evaluations and the result shows that the data integration using Linked Data is successful and most results are valid.
(A) Novelty	5	RailGB addresses the real needs of disabilities using semantic Web technologies and open data. As far as we know, this has not been done before.
(A) Information retrieval	4	Information retrieval is the basic function of RailGB. We go beyond it in that we automatically filter the facilities based on disabilities using semantic Web technologies.
(A) Commercial Potential and large existing user	5	Public places, such as hospitals, museums, etc, can advertise on their accessibility facilities on the map. Not only disabilities, but also people with special requirements at times, such as people taking babies or luggages all could be the potential users of this application.
(A) Rating and Ranking	1	There is no rating or ranking functions currently in RailGB
(A) Use of Multimedia	2	Pictures of underground stations are retrived from DBpedia, but multimedia is not the main function.
(A) Dynamic data	1	RailGB dataset is static, but we will update the data quarterly through TFL API.
(A) Results accuracy	4	According to our evaluation, most results are useful and accurate, but it still needs to improve by applying more up-to-date data.
(A) Multi-language and accessibility	4	RailGB currently is not a multi-language application, but it supports most main-stream mobile phones with web browsers.

Table 1. Minimum and Additional Features