The Web is My Back-end: Creating Mashups with Linked Open Government Data

Dominic DiFranzo, Alvaro Graves, John S. Erickson, Li Ding, James Michaelis, Tim Lebo, Evan Patton, Gregory Todd Williams, Xian Li, Jin Guang Zheng, Johanna Flores, Deborah L. McGuinness and Jim Hendler

Abstract Governments around the world have been releasing raw data to their citizens at an increased pace. The mixing and linking of these government datasets enhances their value and makes new insights possible. The use of mashups, i.e., digital works in which data from one or more sources is combined and presented in innovative ways, is a great way to expose this value. Mashups enable end users to explore data that has a real tangible meaning in their lives. Although there are many approaches to publishing and using data to create mashups, we believe linked data and semantic web technologies solve many of the true challenges in open government data and can lower the cost and complexity of developing these applications. In this chapter we discuss why linked data is a better model and how it can be used to build useful mashups.

1 Introduction

The deluge of raw data that has recently become available due to government transparency initiatives around the world has presented developers with new opportunities to create web-based *mashups*, or light-weight compositions of data and services displayed in compelling ways. Mashups based on linked open govenment data are an important new form of application that enable users in a variety of contexts to discover patterns and correlations that previously may not have been apparent.

Government datasets have been published using a variety of approaches ranging from web services [5] with RESTful APIs [13] (Application Programming Interface) to downloadable ”dump” files. As a result of this movement, users and developers have been able to access and derive benefits from data representing a rich variety of domains including financial markets, health, government, and environ-

Tetherless World Constellation, Rensselaer Polytechnic Institute, 110 8th St., Troy, NY 12180, USA e-mail: difrad@cs.rpi.edu

ment. At the time of this writing Programmable Web[1] lists over 3,000 APIs that are available for accessing data on the Web, illustrating the growing interest amongst producers in getting their data online and developers in consuming this data.

Many examples exist of compelling mashups that have been based on open government datasets. In one case, two EPA-published datasets containing data about ground ozone readings and information about collection sites were combined to visualize the levels of ground ozone in different parts of the US[2]. In another case, foreign aid supplied by the US was compared with that of the UK government. In this second example, not only were datasets from different governments used, but it was also necessary to translate currency from British pounds to US dollars in order to create a comprehensible visualization[3].

A key benefit of creating mashups using the methods described here is that they enable users to make certain observations that are not evident from individual datasets alone. Different organizations may collect data representing aspects of a phenomenon, making it difficult to observe it as a whole. For example, a phenomenon like heavy snow will be relevant to different organizations: The National Oceanic and Atmospheric Administration (NOAA) might report data related to the weather, while data from the Department of Agriculture (USDA) may focus on the effect of the snow on crops; the Department of Education (ED) could utilize this information to determine the conditions under which schools should close. Each organization will include data related to this event in one or more of their datasets, but it is not until we mash them up together that we can have a complete picture of the phenomenon.

2 Motivation

Government datasets published under open data principles contain a wealth of information that can impact the decisions and actions of stakeholders ranging from individuals to organizations. For example, the US Centers for Medicare and Medicaid Services releases data documenting the quality and services of every hospital and nursing home in the United States that accepts Medicare and Medicaid. If presented in the right context, this data could enable citizens to compare the hospitals in their area and enhance their medical decision-making process. Presented in a different context, organizations could use this data to compare and contrast the quality of medical care around the US and investigate what outside factors might be involved. In yet another context, community leaders could use this data to target medical care delivery problems in their area and leverage it to develop solutions. All of this can

[1] http://www.programmableweb.com/apis/directory

[2] http://www.data.gov/semantic/Castnet/html/exhibit

[3] http://data-gov.tw.rpi.edu/demo/linked/aidviz-1554-10030.html

be accomplished with a single dataset, presented in the appropriate context for the user.

Much more is possible when datasets are combined via related fields and areas. For example, the US Department of Labor publishes unemployment rates for every county in the US. By linking unemployment data with hospital metrics, a possible correlation between the unemployment rate and the quality of hospitals in a given area can be investigated. Raw data by itself can be difficult to parse and understand; a table of data devoid of context is unlikely to be meaningful to an end user. As seen in the example above, a single dataset might be useful by itself, empowering individuals, organizations and communities in many different ways: On the other hand, by linking datasets, a user can obtain a better insight on the context related to the data by looking to other datasets. This linking also enables users to see a larger picture of the world, exposing the connections and influences in situations and events. Mashups based on linked data enable stakeholders at many levels to experiment, to discover if potential connections really exist, and to pose questions and rapidly test hypotheses about how and why things happen. The mashup culture of prototyping, linking, experimenting and visualizing helps to bring out the full potential of open government data.

An important advantage of mashups in the government space is their speed and low cost of development when compared with alternative approaches, such as standalone applications created by contractors. Previously, a visualization project could take weeks or even months for a software contractor to implement and cost on the order of thousands of dollars. This meant that organizations and small communities with limited resources could not easily use government data to address their specific questions or to guide their decision-making processes. Now, mashups can be built in hours or days using free tools and services readily available on the Web, including tools like Google Visualizations [7] and Protovis [4]. In general, rapid learning curves enable communities, organizations and even individuals to create their own mashups using open government data without having to hire outside developers or domain experts. Web-accessible visualization tools and APIs enable developers to focus on the important work of linking and mashing the actual data. Communities and organizations now have the ability to not only build and use mashups for their own goals, but also to test and experiment with different ideas and hypotheses without the risk of wasting time and money. Stakeholders can now engage in an iterative, rapid prototyping process to learn and discover more about the specific datasets they care about, and to build better visualizations to communicate their message and achieve their goals.

3 Linked Data

The use of linked data and the application of semantic web technologies alleviates the mashup development challenges stated above. The linked data premise is that data should “work” in the same way the Web of Documents currently works; we

argue that developers can be more productive building mashups based on the emerging Web of Data based on linked data principles. A key advantage of the linked data approach is that it is decentralized, enabling providers to be spread out and their datasets referenced from across the Web. Unlike traditional APIs which provide a single source for data access and querying, datasets published using linked data principles may be hosted, queried and referenced across many sources. This works much like the current web, in which documents are hosted and accessed by systems distributed across the world.

Linked data is inherently modular; there is no need for coordination or planning to link concepts and ideas from different datasets together. Any linked data can be mashed up or combined with any other linked data. The beauty of this approach is that the "real" mashup takes place within the data graph itself, not in the application code as it must with web API-based development. And because links between the datasets are actually in the data itself, they are accessible and reusable to other developers and users. This is analogous to the current Web of Documents: anyone can put up a web page and link from it to any other web page on the Web, and consumers do not need to coordinate or seek permission from the owners of any of the web pages they link to.

These factors combine to make linked data a more scalable approach to constructing open government data mashups. It is easy to extend linked data that has already been published; this remains true even as the definitions and structure of the data change over time. Following linked data principles, anyone can publish semantic data and link to any other semantic dataset. Unlike proprietary APIs, the core technologies that make up linked data — HTTP URIs (Universal Resource Identifiers), RDF (the basic data description language for the Semantic Web) and SPARQL (a query language for RDF) — are open standards and are W3C (World Wide Web Consortium) recommendations. This allows developers who combine web application skills with knowledge of RDF and SPARQL to reuse their knowledge as they encounter new datasets. Applications built using standard Semantic Web technologies can more easily use new semantic datasets that they weren't originally built for. By using SPARQL in tandem with linked data, developers have much more freedom with the types of queries and questions their mashups can ask from the datasets. Developers are not restricted by a simple interface to access and query the data anymore, but rather have access to a complete, robust query language to accomplish their goals.

The linked data approach facilitates reuse by exposing the dataset data model to developers and users. This transparency enables developers and users to see exactly how the datasets are structured and can help the community understand, improve and reuse these models. Developers can still use visualization APIs, web services, programming environments, frameworks, and platforms in the same way as before. Indeed, we can use web services to send SPARQL queries using standard REST methods, and we can get our SPARQL results back in any of the formats we are familiar with (JSON, XML, RSS, etc). The transition to linked data does not require developers to substantially change their patterns for building applications and visu-

alizations, and it enables them to take advantage of and contribute back to the Web of Data.

3.1 Alternatives to Linked Data

In this section we discuss other approaches to create mashups and we contrast them with the benefits of Linked Data.

3.1.1 Raw Files

One of the simplest methods for publishing data on the Web is making files available on a server. In this way people can download those files and use them to create mashups. However these mechanism brings several problems for mashup creators:

- **Lack of background:** The first problem for mashup developers is to understand what the data is about. To reduce this problem, some organizations provide relevant names to their files as well as "README" files in the same directory with metadata and a description of the files. One of the problems with this is that the description is not in the data, making up to the developer to understand the inherent relations between the different values in the dataset. For example, it is difficult for developers to understand when a field accepts a range of values (instead of a any numerical value): Expressing this restrictions in the data itself (as it is done using Linked Data) makes it easier for developers to understand and simplifies applications and mashups.
- **Difficulty of reuse:** Another related problem is that for every new developer trying to use the data, she will have to start understanding the data from scratch. Depending on the complexity of the data, this cost (in terms of time and effort of the developer) may be too big, discouraging people to use this data.
- **Raw data cannot be uniquely identified** Another problem that appears is the lack of unique identifiers for the entities that data describes. Thus, if records from different datasets are identified by number "1324" is not clear that they describe the same entity. Another example is when two or more datasets describe information about New York: Do they mean New York *city* or New York *state*?. Linked data base
- **It is hard to search information from other datasets:** When using raw files as a main publishing mechanism, searching for other datasets that describe the same entities can be very time consuming: It will require the developer to manually look for these new datasets using search engines and other mechanisms. Linked Data allow to specify explicitly where and how (by URI dereferencing) to find more relevant information about a certain entity.

3.1.2 Relational Databases

Another alternative is to use Relational Databases (RDBs). The main benefit is the maturity of this technology, scalability and robustness. However, in a dynamic environment, RDBs present different issues:

- **Need to know the schema** *apriori***:** One of the biggest problems is that developers need to know the schema (that is, tables, indexes and other objects) in order to make a successful (and efficient) query. Part of the knowledge related to the data is expressed in this schema and hence developers must be aware of both instead of being concerned only about the data. On the other hand, Linked Data relies on RDF, which is a schema-less language: The structure of the data is in the data itself and not determined by a schema.
- **Flexibility:** In general, small changes in the data will imply a change in the schema, however this is not trivial: Adding more data could imply a new redesign of the schema including adding new tables, primary and foreign keys, changing column types and so forth. The use of Linked Data allows people to mix data from different sources given its graph structure.
- **Sharing:** Another problem arises when people shares and reuses datasets stored in RDBs: Different RDB engines uses slightly different versions of SQL making it difficult to import the data from one engine to a different one.
- **Reuse and Security:** An important aspect is that most of the RDBs are not designed to be used in the open Web: Applications needs to access the database using a security scheme (usually user/password on the organization's internal network). Thus, it is difficult for developers outside the organization to use the data stored in the RDB. On the other hand, Linked Data is focused in sharing the data on the Web: Most of the endpoints available are open for any developer. Also, it is possible to download dumps of the datasets so others can use it in their own endpoints.

3.1.3 APIs (Application Programming Interface)

Current mashup development practices make heavy use of traditional web services and APIs. Raw data released as downloadable files is not directly useful to end users or developers as a basis for applications. Typically data must be structured in some way so that it can be more readily queried, and from those query results applications can be built. RESTful [13] APIs in recent years have been more widely used as a means to expose structured data to developers, enabling applications to retrieve data from web services and other applications through standard HTTP requests. For example, using the Twitter search API, a developer can use a HTTP GET request on `http://search.twitter.com/search.json?q=#semweb` to retrieve a JSON file full of tweets containing the hash `#semweb` in them.

The RESTful API approach is attractive because it enables developers to easily query data using almost any platform or programming environment. In the open government data space we have seen many of these APIs emerge to help develop-

ers build applications. These APIs come from both the private sector (e.g. Sunlight Foundation, MapLight.org, FollowTheMoney.org) and public sector (e.g. Recovery.gov, TradeStats Express, Business.gov Web Service API). RESTful APIs, while incredibly useful for accessing data, pose several challenges in terms of their development and application:

- **RESTful data APIs only answer the queries they were built to answer.** An important aspect of Web users is that their nature can be described as a "long tail", which means there is never a truly "average" user [2]. Data providers should assume that a large percentage of users will want to ask questions of their data, make connections and discover things that the providers might not have anticipated. Mashup developers may seek answers from the data that are not supported by an API or available through a very complex composition of their services.
- **RESTful data APIs by themselves are not standards.** Each API has been designed around the particular service its creator wishes to provide. Knowing how to query using one API does not imply knowledge of how to use another. REST is a standard architecture that many of these APIs are based on, but the specific implementation and interface design of individual APIs can vary greatly. This means each time one encounters a new API, one must re-learn how to handle queries, what types of queries are supported, what format they come in, etc. The need to learn every new API makes it difficult to add and mash new APIs together.
- **RESTful data APIs are opaque.** Many APIs make it difficult to see, reuse or improve the underlying data or data model. Time and energy has been expended modeling and structuring data to create a usable API, but the resulting data model invariably gets hidden from developers and users. A better approach would make such data models systematically available for other systems and datasets to reuse and to be improved by the community. Moreover, data is "trapped" inside these APIs, creating "data silos" which only return the results of the queries that the given API supports, in the formats that it allows. Because of this, the linking and mashing of data in a mashup must happen in the application code of the mashup rather than natively within the graph. This means that linking must be hard-coded in the mashup and can't be easily reused by other developers and users, forcing developers to start each new mashup from scratch. Developers spend time and resources determining how to interconnect the results from their chosen APIs, but it is difficult for others to reuse this work.

Beyond the specific constraints imposed by APIs, there are other challenges developers must face when creating mashups, the biggest of which is *data quality*. It is critically important that developers use "good" data that is well-defined and understood. Developers must establish the history of the data, its origins, how was it collected, and especially what processing has been performed on it so that it can be presented correctly. A cursory review of the many raw government dataset catalogs worldwide reveals that data quality varies greatly, with no established standards. Moving forward, linked data practitioners must find ways to help improve this situ-

ation across the board to help developers and users understand and use these datasets correctly.

4 Workflow for Developing Linked Data Mashups

As previously stated, the objective of mashups based on linked data is to replace APIs and other transitional backends for mashups and applications with linked data accessed through SPARQL queries. With this approach the power of semantics and semantic technologies can be leveraged while still utilizing familiar visualization tools and programming environments.

4.1 Converting, Understanding and Enhancing Linked Data

The TWC mashup workflow consists of several steps[4]. First, raw data must be converted into RDF, the fundamental data model for the Semantic Web. There are many tools and converters that may be employed for this; at TWC we have built our own tool, `csv2rdf4lod` [9], that converts tabular data such as raw CSV (comma-separated values) files into RDF. The `csv2rdf4lod` tool, including its source code and documentation, has been released as an open source project[5] to enable other developers and data publishers to follow TWC's model for their own data conversions.

The next step is in many ways the hardest and most time-consuming when building mashups from any data: gaining a deeper working understanding of the datasets that will be the basis of the mashup. There are no step-by-step procedures or automated processes for this; it requires developers to do research and to manually inspect the data. Many interesting datasets have been released by a number of government agencies and organizations, but they all vary in the metadata and data dictionaries released with their datasets as well as the actual structure of their data. There are few standards in this regard and it can be very difficult to understand the context, definitions and history of the data we want to work with. Even when dataset metadata is available it may be difficult to work with the data absent the proper domain knowledge for the dataset. For example, we might be interested in a dataset containing "readings of ground ozone around the United States over time," but if we do not know that ground ozone is considered an air pollutant, then we will not see a potentially valuable use of this data. Understanding the data involves researching the organizations that have released the data, the processes they've used in collecting and publishing data, and gaining some of the domain knowledge that the data represents.

---

[4] For a more comprehensive discussion of the TWC LOGD conversion process, please see Li Ding, et.al., "TWC LOGD: A Portal for Linked Open Government Data Ecosystems" (this volume)

[5] `https://github.com/timrdf/csv2rdf4lod-automation/`

Once we understand the data better, our goal is to enhance the converted RDF and to introduce links to other datasets. We also annotate the dataset with the metadata and provenance information we've found through our research to help other developers and data consumers more easily use the data without having to start from scratch. These enhancements enable the dataset to become more reusable and saves the time and resources of others in using open government data. At TWC we re-apply the `csv2rdf4lod` converter again to enhance our datasets with ranges and descriptions of the different properties found in the data.

Equipped with an understanding of the context and broader domain knowledge of the data, we can target other datasets that may be good candidates to link to and mash. We also look for common properties between datasets to use as "bridges" to link them together. For example, if two datasets both have geographic information (e.g., describing facts about different US states), we are able to link these values (in the form of URIs denoting each state) to get a more complete picture of what is happening, based on data about different states drawn from multiple datasets.

Different strategies may be used to link resources from multiple datasets. First, if we have a deep understanding of the datasets involved and what their values mean we can directly claim that two URIs should be linked. A common example of this is asserting that two URIs represent exactly the same entity; (for example, New York State). Hence we can say that

```
ex1:resourceA owl:sameAs ex2:resourceB .
```

where `owl:sameAs` is the predicate to express identity of instances in OWL [10]. It is possible to express semantically different relations, for example using predicates from SKOS [12] (`skos:broader`, `skos:narrower`) or Dublin Core [14] (`dc:relation`). Another possibility is to use literals to discover potential links between URIs. For example, consider two datasets that both provide information about US states. One uses `New York` as a state name while the other uses `NY`. We could then establish that both entities refer to the same concept and use `owl:sameAs` to link them. Sometimes links are less obvious and depend on domain knowledge of individual datasets and the level of abstraction desired; for example, the URI for the "President of the United States" and the current president may be identical, but only during a determined period of time.

A string literal like `NY` may be "promoted" into a URI that is unique across the whole web. For example, consider a dataset providing average rainfall data for all states in the US for the year 2009. The string `NY` occurring in this dataset can be promoted into a URI like `http://example.com/average_rainfall_2009/NewYork`. Other references to the state of New York in other datasets may then be linked via this new URI. Consider then a second dataset listing the populations of all the states in the US for the year 2009 and which refers to the state of New York as `http://other-example.com/population-2009/New_York`. These two URIs may be explicitly linked together using `owl:sameAs` as mentioned above. Once linked in this way, these datasets may be queried together to obtain results for average rainfall and population in 2009 for the state of New York. Another benefit of linking datasets by introducing explicit URIs is that other developers may reuse

these links and add their own to datasets they are interested in. Currently this can be a time-consuming process; there are tools that can help make this process easier, but there still must be a human in the loop.

5 Case Studies

We now present detailed case studies illustrating how mashups may be used with open government data. First, we present a mashup that display information related to crime, transport and education in the UK. Second, we show how diverse datasets were used to create an iPad/iPhone application enabling users to review the visitors to the White House and who they visited. Lastly, we show how US and UK government data may be used to compare the levels of foreign aid provided by these governments to other countries.

5.1 See UK

"See UK" is an application that displays information related the United Kingdom taken from multiple datasets from *data.gov.uk*. The user can select what information to be displayed from different datasets available related to crimes: Thus, it is possible for example to visualize only crimes classified as burglary, robbery, and others. See UK can display information about transportation, thus allowing users to visualize train or bus stations. Finally, the application can show different information related to schools, students enrollment, absences, among others. The user can select an area by indicating its postcode or by clicking on it in a map; Depending on the level of zoom, this area can be a ward, a county or a region (including several counties). Once the user selected an area, displays a pie chart displaying the values using by colors (green when crime is low, red when is high). The outer layers of the pie reflect the values for the adjacent areas to the one selected.

The data is described using RDF and it was collected from different datasets Internally, See UK uses 4Store as a triple store and as SPARQL endpoint for the backend while the front-end is implemented using PHP and JQuery. A screenshot of SeeUK can be seen in Figure 1.

5.2 Whitehouse Visitor Log Demo

Another mashup developed by TWC was based on visitor records from the White House[6]. The Whitehouse Visitor Log Demo enables users to search for people who

[6] `http://logd.tw.rpi.edu/demo/white-house-visit/search`

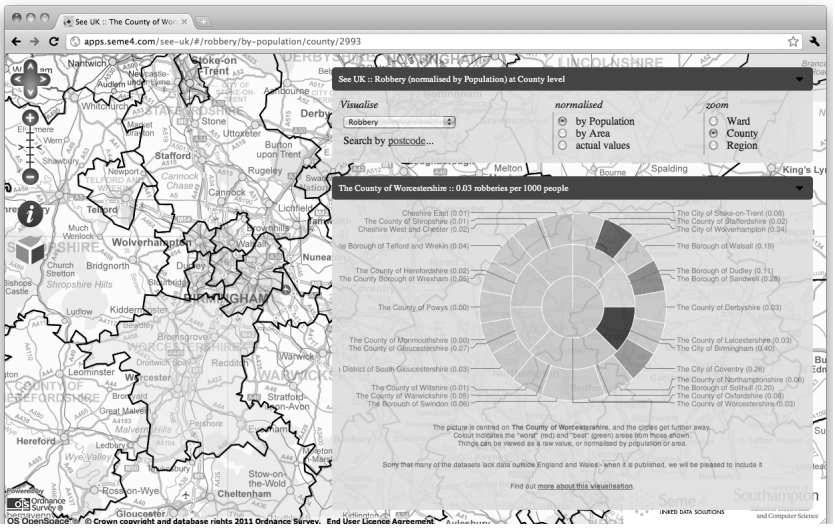

Fig. 1 Screenshot of SeeUK, a mashup that display information about education, transportation and crimes in the United Kingdom. Users can specify an area in the map and a pie chart will display the value for that area and its neighbors.

have visited the White House and the people they met there. For the subset of White House employees with Wikipedia pages the mashup displays information including their title, a picture, a short biography and their homepage. A user for example can search for "POTUS" (President of the United States) to see the top 25 people who have visited him (displayed using a table, bar chart and pie chart) in addition to information about him from Wikipedia. This additional, linked information provides the user with additional context about who this person is and insight into why specific people might be visiting him.[7]

Our first step in creating the mashup was to obtain, study and convert the raw data. The data for the White House Visitor Search came from the White House Visitor Logs dataset [8], published as a CSV file that included information on the first and last name of everyone who has visited the White House, when they made their visit and the staff member they visited. In working with this dataset we made the assumption that an individual's name uniquely identified them in the dataset and that there would be no overlap, meaning there wouldn't be two people with the same name. In general this will not be true, but given the dataset and accompanying documents and metadata it was a reasonable assumption to make. We used the `csv2rdf4lod` conversion tool to convert this raw data into RDF.

[7] One obvious extension of this would be to display available information about the visitor, including their party affiliation and employer.

[8] `http://www.whitehouse.gov/briefing-room/disclosures/visitor-records`

Based on our RDF version of the visitor logs dataset, we explored ways to link this data with other outside datasets. We knew for example that many of the White House employees had pages written about them on Wikipedia and would therefore also have records in DBpedia [3], a semantic data version of the information found at Wikipedia. DBpedia provided us with a great target to link into and allowed us to query for more information about the people in our original dataset. To demonstrate this we manually found 30 White House employees who could be linked to DBpedia; using this information we were able to enhance the White House Visitor dataset with links to DBpedia.

With these linked data "hooks" in place we could then query the White House Visitor dataset, obtain the DBpedia links for the correct results, use these URIs to query the SPARQL endpoint for DBpedia[9] and get additional results like the employee's picture, title and homepage from Wikipedia.

In addition to our web mashup, we wanted to explore creating a mashup based this data for mobile platforms. Publishing a Semantic Web application in the mobile space required a different focus on the presentation of the data, especially due to restricted display space and the fact that the existing suite of visualization tools used in the web based version were written in Adobe Flash, making them —at the time of this writing— unusable on iOS. Since we expected this work would be presented at locations where wireless access was prohibited, caching mechanisms were necessary in order to store data on the device in the event that a network connection was unavailable.

The mobile version of the application provides a simple wrapper around three libraries: one that provides an RDF data model, another that provides methods for performing SPARQL queries and iterating over the results of SPARQL select statements, and a third that generates pie graphs of retrieved data. SPARQL results are serialized out to a cache directory in additional to any supplementary data retrieved from external sources.

As with the web mashup, the mobile app takes advantage of linked data by performing queries against sites like DBpedia that may provide additional annotations about an individual, including their current office and title. Also, by using the Freebase Globally Unique ID (GUID) retrieved from DBpedia the mobile app can perform image lookups. DBpedia also includes links to the New York Times linked data, and the mobile application can search through the New York Times API [10] to find recent articles discussing the individuals in question. Figure 2 provides a screenshot of the mobile White House Visitor Log applications.

Since a single lookup can potentially produce up to five queries that must be executed, the mobile app includes options for adjusting the `LIMIT` clause in the SPARQL query, providing better responsiveness to the user when the query is limited to a smaller set of results. The user is also given control over what linked data sources will be used to build a profile of the individual. By default, DBpedia, Freebase, and the New York Times are all enabled. Turning off these options can increase

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

[10] `http://data.nytimes.com`

application performance by reducing the number of external queries at the cost of limiting the comprehensiveness of the results, given that the Data.gov datasets are extremely targeted.

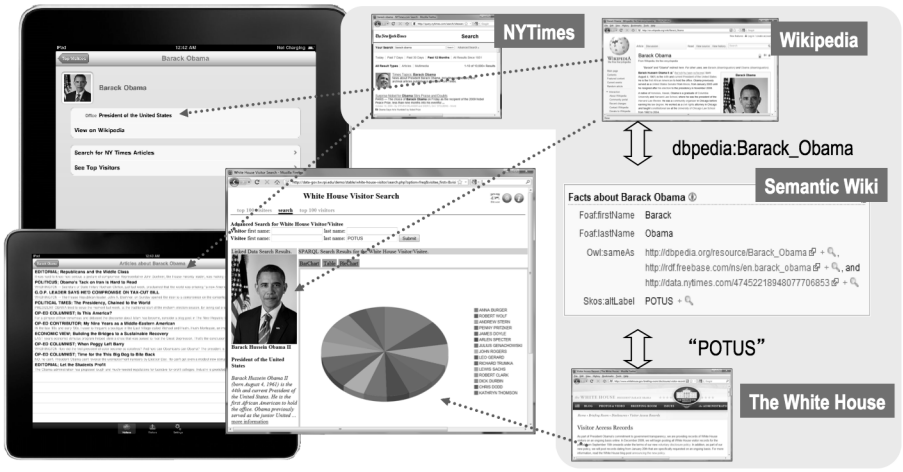

Fig. 2 Screenshot of the mobile version of TWC's White House Visitors Demo. This mashup enables users to explore which White House employees have been visited and by whom. By linking URIs of different people to those in DBpedia, Freebase and other external datasets, it is possible to obtain more detail information about these people.

5.3 Comparing USAID (US) and DFID (UK) Global Foreign Aid

In this TWC-created mashup, foreign aid figures provided by the United States Agency for International Development (USAID)[1] and United Kingdom's Department for International Development (DFID)[2] from 2007 are jointly presented via the interface.

Here, each aid-receiving country is shaded based upon total USAID and DFID aid received. Upon clicking a particular country, three pie charts are presented:

1. A comparison of total USAID and DFID spending.
2. A breakdown of USAID spending by category.
3. A breakdown of DFID spending by category.

Additionally, contextual information for each country is pulled in from two sources:

[1] http://www.data.gov/raw/1554/

[2] http://data.gov.uk/dataset/dfid-statistics-on-international-development

1. **The New York Times Article Database**[3]**:** This provides news stories involving foreign aid efforts.
2. **The CIA World Factbook**[4]**:** This provides demographic data, such as literacy rate and life expectancy.

The first step in designing this mashup involved processing raw data from USAID and DFID — published in Microsoft Excel spreadsheets – into RDF using the `csv2rdf4lod` tool described earlier. Following this, three issues had to be resolved to effectively mash up data from these two agencies with the contextual information described above:

1. **Country Names**: Slight variations on labels assigned to countries could be observed between USAID, DFID data (e.g., "Congo (Kinshasa)" versus "Congo (Dem Rep)") and CIA World Factbook data.
2. **Currency**: USAID and DFID figures were represented in the raw data using US Dollars and Pound Sterling, respeciviely.
3. **Fiscal Year**: Additionally, USAID and DFID figures were represented with different fiscal years (April 6th to April 5th in the UK, versus October 1st to September 30th in the US).

To resolve the country name mapping issue, an RDF-based vocabulary was defined which assigned a unique ID to each aid receiving country. Attached to these IDs would be sets of country labels, corresponding to potential variations across datasets. This approach established label equivalence among datasets, and helped facilitate SPARQL-based retrieval of content from multiple data sources.

The currency issue was resolved by performing a conversion calculation at the application level. The currency conversion step was accomplished by representing DFID figures in US Dollars, using an RDF-based record of currency exchange figures from January 2006 to January 2008. These figures were derived from a CSV file downloaded from the online currency conversion service oanda.com.

Finally, the fiscal year issue was resolved by normalizing DFID figures to the US fiscal year. This was accomplished an application-based calculation which averaged DFID figures reported for the UK 2006 and UK 2007 fiscal years.
Designing this demo helped illustrate some current challenges faced by developers in mashing up international data.

First, an appropriate agency-to-agency mapping had to be determined. Both USAID and DFID are tasked with managing foreign aid contributions. However, it could be argued that other agencies in both the US (e.g., the Department of Agriculture) and UK would need to be considered to establish a better comparison. Without a standardized inter-agency comparison for developers to reference, such mappings will remain a matter of debate.

Second, the USAID and DFID data required prior processing before any form of mashup could be made. Specifically, the data had to be normalized to a common currency and fiscal year. As with the first issue, the specific processing steps may

[3] http://developer.nytimes.com/docs/article_search_api

[4] https://www.cia.gov/library/publications/the-world-factbook

not be agreed upon by all developers. Nonetheless, such processing must be carried out in some form to enable mashups of international data to be produced.

6 Teaching the Art of Mashup: Workshops and Hack-a-thons

The Tetherless World Constellation has hosted a number of events, usually presented as *workshops* or *hack-a-thons*, for teams from government and non-governmental organizations (NGOs). In these hands-on events we've demonstrated the potential of semantic web technology and linked data best practices and have shown how to create visualizations and mashups relevant to the particular audience.

6.1 Motivation and Audience

The focus of TWC-run workshops and hack-a-thons is to show others the true advantages of using linked data in creating mashups and to motivate attendees to do it themselves. Each mashup creation step is covered in depth, and participants are encouraged to practice the methods using their own data. Attendees at TWC workshops and hack-a-thons typically are stakeholders from a variety of backgrounds; it is not unusual to find web developers and hackers intermingled with data analysts, managers and government contractors.

6.2 Event organization

Workshops and hack-a-thons conducted by members of the TWC team usually consist of an audience from the stakeholder organization, one or more tutors and one or more "featured" speakers. A hack-a-thon agenda will typically include the following steps:

1. **Presentation:** Speakers and tutors are introduced and the objective for the meeting is presented.
2. **Introduction to Semantic Web:** The speaker presents a brief overview of the Semantic Web, covering its goals, its potential and the current technology available. This also includes a brief high-level description of Linked Data and how it is applied to Open Government Data.
3. **Group presentation:** Attendees form groups, introduce themselves and decide what they want to accomplish with their demo. Each group presents their goal to the other groups.
4. **Group work:** The Groups code their demos, basing their work on the tutorials and demos discussed during the presentation and the introductory Semantic

Web and Linked Data material. Depending on the number of tutors and groups, tutors might work with one group directly or they might roam between groups, providing guidance and answering questions.

5. **Final presentation:** At the end of the hack-a-thon — at the end of a day, or the next day if it lasts for 24 hours — groups present their progress, ideally showing a finished demo. The group explains what techniques and tools they used and the challenges and difficulties they encountered.

6.3 Tools and resources

Nearly all of the tools used in TWC workshops have been documented in the TWC Linking Open Government Data (LOGD) Portal (`http://logd.tw.rpi.edu`) as well as its Wiki-based predecessor (`http://data-gov.tw.rpi.edu`). From these sites it is possible to learn how to use semantic technologies by working through a number of tutorials dealing with many aspects of mashup creation (`http://logd.tw.rpi.edu/tutorials`) or by worked examples in our "Demos" section (`http://logd.tw.rpi.edu/demos`), where users can review the actual source code. Another important resource is the LOGD dataset catalog (`http://logd.tw.rpi.edu/datasets`) which lists all the datasets converted to-date by the TWC team. Each dataset may contain different versions and enhancements, so each converted dataset on TWC LOGD also has a description page with more details.

In addition to the TWC LOGD dataset catalog, an important tool we use to create mashups during hack-a-thons is the SPARQL Proxy (`http://logd.tw.rpi.edu/sparql`) which provides an endpoint for several of the more popular datasets. This proxy enables developers to query the data and obtain the results in different formats (XML, JSON, etc.) according to their needs. When data needs to be converted that is not available from the TWC LOGD catalog, the TWC-developed CSV2RDF4LOD converter may be used to convert tabular data to RDF [9].

In most of our hack-a-thons the TWC team has promoted a lightweight, rapid approach to development. For example, we find that using client-side JavaScript as a programming model leads to less complexity in terms of configuration and installation than using server-side languages. Since most of the development done during these events is based in HTML and JavaScript, the list of requirements for creating demos is simple: a modern browser, a text editor and an Internet connection. Moreover, there are multiple external tools and services written in JavaScript that can be easily integrated into mashups. We have found that Google Visualizations [7] is particularly well suited for this due to its simplicity and short learning curve. We have also explored other visualization tools, such as Yahoo! Pipes [6], Simile Exhibit [8] and others.

Finally, we use Drupal [11] as a Content Management System (CMS) for publishing data, demos and tutorials. Drupal provides a robust and well-tested infrastructure that can be extended by adding external modules developed by the Drupal community. In particular, our team has customized the TWC Drupal6 instance, enabling it to publish RDFa [1] related to datasets, demos, people and others from our SPARQL endpoint embedded in XHTML pages.

6.4 Lessons learned

Based on our experience leading or participating in a large number of workshops and hack-a-thons we have found that there are certain factors that will improve the experience of participants, instructors and tutors:

- **Group composition:** It is not always the case that groups will consist of technology-savvy participants. This makes it difficult for the group to work together to create a new demo, for example due a lack of experience in web development. Organizers should mix up the composition of groups to ensure that each has members from different backgrounds.
- **Preparation of materials:** If the background and concerns of the audience are known ahead of time, organizers should prepare a set of tutorials and demos specifically designed with these interests in mind. For example, if the majority of attendees work on environmental issues, tutorials and demos should be created or adapted using datasets from that particular domain. An example of a demo created specifically for one of these events can be seen at `http://logd.tw.rpi.edu/mashathon2010/example/2`. In this case the TWC team demonstrated mashups that mixed data describing the cigarette taxes with data specifying the number of books in each state.
- **Publication of results:** Attendees should be encouraged to ”take home” a final working product they can review with their colleagues after the hack-a-thon. We recommend that demos created during the event remain publicly available so attendees can revisit them after the event. For example, the results from a hack-a-thon done in August 2010 can be visited at `http://logd.tw.rpi.edu/mashathon2010`.

7 Conclusions

In this chapter we have considered the role played by mashups in unleashing the full potential of open government data. We have considered the pros and cons of other popular approaches of developing mashups based on raw data, Relational Databases and RESTful APIs and have argued that the use of semantic web technologies has distinct technical advantages over them, due especially to the inherent benefits of Linked Data. We have outlined a step-by-step procedure for creating mashups based

on the open government data available through the TWC LOGD Portal. Finally, we have presented a model plan for hosting a government data hack-a-thon event including the organization of the event, the tools that should be used, and specific recommendations based in our experience that will enable attendees to better understand the principles of mashups and explore real use cases as they create compelling mashups based on linked open government data.

References

1. B. Adida and M. Birbeck. RDFa primer 1.0: Embedding rdf in XHTML. *W3C working draft*, 12, 2007.
2. Chris Anderson. The long tail. *Wired*, 12(10), October 2004.
3. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. *The Semantic Web*, 4825:722–735, 2007.
4. M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1121–1128, 2009.
5. F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *Internet Computing, IEEE*, 6(2):86–93, 2002.
6. J.C. Fagan. Mashing up Multiple Web Feeds Using Yahoo! Pipes. *Computers in Libraries*, 27(10):8, 2007.
7. Google Inc. Google Visualization API `http://code.google.com/apis/visualization/documentation/gallery.html`.
8. D.F. Huynh, D.R. Karger, and R.C. Miller. Exhibit: lightweight structured data publishing. In *Proceedings of the 16th international conference on World Wide Web*, pages 737–746. ACM, 2007.
9. T. Lebo and G.T. Williams. Converting governmental datasets into linked data. In *Proceedings of the 6th International Conference on Semantic Systems*, pages 1–3. ACM, 2010.
10. D.L. McGuinness, F. Van Harmelen, et al. OWL web ontology language overview. *W3C recommendation*, 10:2004–03, 2004.
11. D. Mercer. *Drupal: Creating Blogs, Forums, Portals, and Community Websites*. Packt Publishing, 2006.
12. A. Miles and S. Bechhofer. SKOS simple knowledge organization system reference. 2008.
13. L. Richardson and S. Ruby. *RESTful web services*. O'Reilly Media, 2007.
14. S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin Core metadata for resource discovery. *Internet Engineering Task Force RFC*, 2413, 1998.