Will this work for Susan? Challenges for Delivering Usable and Useful Generic Linked Data Browsers

m.c. schraefel[1], Daniel A. Smith[1], Igor O. Popov[1],
Max Van Kleek[1], and Nigel Shadbolt[1]

School of Electronics and Computer Science,
University of Southampton, SO17 1BJ Southampton, UK
{mc,ds,ip2g09,emax,nrs}@ecs.soton.ac.uk

Abstract. While we witness an explosion of exploration tools for simple datasets on Web 2.0 designed for use by ordinary citizens, the goal of a usable interface for supporting navigation and sense-making over arbitrary linked data has remained elusive. The purpose of this paper is to analyse why - what makes exploring linked data so hard? Through a user-centered use case scenario, we work through requirements for sense making with data to extract functional requirements and to compare these against our tools to see what challenges emerge to deliver a useful, usable knowledge building experience with linked data. We present presentation layer and heterogeneous data integration challenges and offer practical considerations for moving forward to effective linked data sensemaking tools.

1 Introduction

Tim Berners-Lee's TED talk in 2009 chanted the mantra "Raw Data Now"[1]. This plea came in concert with the then newly opened data.gov archives in the US and data.gov.uk resources in the UK - the plea was for EVERYONE - not just government - to free their data. Berners-Lee motivated the crowd by showing what kinds of new insights and knowledge we could build by being able to pull together multiple data sets to see new relationships. His example was Hans Rosling's Gapminder's The Health and Wealth of Nations which correlated population growth in countries and GDP: new knowledge from new combinations. Fabulous vision. Indeed, this is at least in part the goal of the linked data enterprise: not just to produce the data, but to let us gather it up, bring them together, see new things and create new knowledge. So far, our tools to support this munging across heterogeneous data sets have not achieved this goal. Our examples of what we can do with "raw data now" are variations of custom, hand crafted applications, either by scraping data sets and putting them into custom applications like Rosling's, or custom visualisations crafted over a single

[1] Tim Berners-Lee TED Talk: http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html

open data set like government spending[2] that can show US defence spending by region. While these are powerful hacks, they are not empowering applications. We can only operate the functions supported by the application; we cannot, for instance, simply compare in one view military with health care spending over time, nor can we add in other open data sets of our choosing, for instance, to compare military spending between the UK and US against GDP.

The data web is still experienced as largely inaccessible to citizen users. By citizen users we mean people willing to engage with edgy technology - to a point. Consider the response of twitter-literate users to the release of the much anticipated UK COINS open data about UK Government expenditures, commented furiously [3] about the lack of information of the format: "Why the hell isn't the data stored centrally in a database with web pages drilling into it? Most ppl (sic) don't know what to do with a CSV file. Surely if you going to go to the effort of publishing then do it properly so that non technical people actually do have access." and at the uselessness of the size: "Why make the files so large that most people cannot easily view them". Indeed, CSV downloads of the data crashed the computers of geeks too when excel was unable to handle the size of the data. In the Semantic Web community, our approach to the above issues is first, convert the data sources into RDF, as has begun with data.gov [4], and then, provide a linked data browser over that data. What we know from our own experience as semantic web and linked data researchers, and from the literature of effort in this space (detailed below), is that usable linked data interaction for citizen users (ie non-geeks), however, is hard to deliver, and our successes limited. In the EnAKTing project, our goal over the past six months has been to investigate specifically what makes delivering an effective, functional and generally usable interaction with linked data difficult, and from this, to review the specific problems both known and, in particular, surface up any new issues from this process, that if addressed, would more likely enable the community to develop effective user applications.

Our hypothesis for approaching this problem has been that we will generate the most useful results if we focus this interrogation through a sufficiently challenging and realistic use case. In other words, rather than focusing on the technology limits to perform certain functions, to look at citizen user requirements to prioritize problems to be solved. To this end we have used a standard requirements engineering/human computer interaction methodology of scenario generation based on specific user profiles and use cases. From these we carry out a coarse grained requirements elicitation. We note that this approach is not about User Interface design at this stage, but offers a functional analysis for eventual tool interaction design. Our analysis has produced requirements at two levels: the presentation layer and the heterogeneous data integration layer. The requirements for the presentation layer are characterized as linked data discovery, legibility, query manipulation and presentation. The heterogeneous data

[2] `http://www.usaspending.gov/`
[3] COINS release with comments: `http://data.gov.uk/dataset/coins`
[4] Linked Data on data.gov.uk: `http://data.gov.uk/faq#whatisthesemanticweb`

integration layer considers the particular problems associated when trying to blend sources for presentation.

In the following sections, we describe an example scenario and walk through of the method used to inform our analysis of these components. We then detail the functions and associated challenges for interaction and application design in each of their related components. We conclude with a discussion of possible implementation approaches to some of these challenges and articulate the cost/benefits of various approaches that do not yet seem to be complete answers and where more work remains to be done. One of our early take aways from this exercise has been that the "killer app" for linked data may well not be a single application as we have a single browser for the web, but because of the diversity of tasks, we may be looking for a Gestalt: a suite of highly functional and usable services that can be readily combined to deliver a meaningful way to work with linked data for citizens.

2 Method

Our initial approach has been to use scenario development for requirements elicitation. From this we carry out a functional analysis of the requirements by looking at what tools currently support the task outlined in the scenario/use case and where there are gaps. In our case we follow up this analysis with a comparison of any non-linked data tools with any extant linked data tools against our specific user (also sometimes known in this context as a stereotype) to see how they do or do not pass the user usability test. This check helps us understand where work may need to be done to enable uptake by this person/class of users. We used multiple scenarios with multiple citizen users. In this paper, however, for reasons of space, we walk through one scenario, "where should i live?" For reasons of space, we are also compressing much of the detail of the scenario and stereotype, and so offer an outline of the process sufficient for illustration, and to ground the following sections.

> *Susan, a 22 year old Asian American university exchange student and citizen web user is studying sociology. She has a small fellowship to move to a university in England for a year of her PhD to look at particular population statistics for her thesis. Susan has never traveled abroad and while she has lived away from home for three years, has only lived in dorms while at university and now needs to find an apartment. She knows that she would like to be relatively near her school, in a safe neighborhood, a place that is fairly cheap, and that will support short-term renting or subletting.*

Walk through. Susan's exploration through the data is iterative. She begins her search by a general "student apartments in Southampton" and explores a number of listings of apartments, and services to return more results. However, as Susan is unfamiliar with the areas listed, it becomes difficult to use the sites

that require her to pick an area. For example, she asks "how's Shirley, Portswood or St. Deny's compared to Bassett for safety?" She is also finding it time consuming to work through questions like "how close is the nearest (pharmacy/grocery store)?", "how good is public transportation?", "where are laundry facilities", "parking", "traffic". "What are the people like who live in this neighborhood? How noisy is it typically? Crime? How conservative/liberal? Wealthy? How close is it to the city center? To the Uni?" As she does not yet know other students at the school she cannot see where others like her live. She has found the university's student letting page, but these too only list locations for flats rather than information about the neighbourhoods. While ideally a local expert would help synthesise much of this information if one can be found (similar to getting the ideal search result that returns the perfectly complete document), there may also be gaps in one expert's knowledge. For instance a Caucasian student may be less aware of the fact that a particular area that is otherwise student friendly has been the site of a series of attacks on Asian students. That data might be a deciding factor between two equivalent areas.

Available Data. Currently, if Susan were to iterate her search to look for data (rather than documents) related to areas in the UK, there are data sets about crime statistics [5] [6], hospital waiting times [7], air quality [8], public transit routes [9], average household incomes [10], education levels, occupations/unemployment rates, ages and life expectancies, ethnic backgrounds of residents [11], members of parliament (to infer political leaning) [12] providing that Susan includes the term "UK data" in her search and has infinite time to collate the information for herself. Amazingly while Susan might be able to see the crime rates in Southampton for any given year, the government's data service does not support comparing years; only regions [13].

Susan finds that more similar data sets exist on data.gov.uk but discovers that she cannot ask for "all information related to Southampton" (concept match, in an information retrieval perspective) but must ask for specific data sets (term match) - like crime or hospitals. Whether and how much of these sources may be useful for Susan is not clear without interrogating them, but many resources requires user accounts for each data source before being able to view the data. She aborts the effort. Back in the Web 2.0 world, Susan stumbles across their mash ups that plot Craigslist apartments on maps [14], but this is a US only service. With considerable manual effort and infinite time, she sees it would be

[5] http://rds.homeoffice.gov.uk/rds/ia/atlas.html
[6] http://www.dailyecho.co.uk/li/recorded_crime.in.southampton/
[7] http://www.performance.doh.gov.uk/waitingtimes/2007/q3/index.html
[8] http://www.airquality.co.uk/index.php?zone_id=8
[9] http://mappery.com/map-of/Southampton-Public-Transport-Map
[10] http://neighbourhood.statistics.gov.uk/dissemination/
[11] http://neighbourhood.statistics.gov.uk/dissemination/
[12] http://www.publicwhip.org.uk/project/data.php
[13] http://rds.homeoffice.gov.uk/rds/ia/atlas.htm
[14] HousingMaps: http://www.housingmaps.com/

possible to pour her apartment data into a suite of spreadsheets to try to craft comparisons of regions, at least according to public data, and to use spreadsheets to visualize these comparisons. While google in fact does produce a mapping of pharmacies in Southampton and Grocery Stores in Southampton, there is no way to blend these mappings, and they are not complete. The will to live slowly leaves Susan as she contemplates how much time this will take - minutes in her life she will never get back - and decides to stay in the states and to chat with other researchers in the UK via web-cam.

The above exercise alone reinforces how current data oriented resources fail often because they have simply been produced; many of their features have not been tested against scenarios for stereotypical users. In the following two sections, however, we consider how an analysis of Susan's dilemma unfolds into two classes of problems: first, presentation, including source discovery, legibility, querying, and presentation of results, and second, integration of heterogeneous data. We already see weaknesses in single data sets that do not support inter-data set comparisons. Each of these sense-making actions is exacerbated as soon as we wish to blend in related sources.

3 Challenge 1: Presentation layer for a linked data browser

Susan's search quest, while specific to her needs, represents a typical engagement with search as expressed as information retrieval and sensemaking tasks (summarised in [16]). : (1) source discovery - initial review of kinds of information that are needed, seek out available sources (2) triage - from the available sources, assess each rapidly to see if it is worth further probing; (3) once a data set is determined to be useful, interrogate it further to produce a result. In the following section we consider how best of breed application attributes facilitate such discovery, triage, legibility and analysis. We then look at how these approaches might be applied to linked data tools for more effective data presentation and manipulation.

Data set discovery. On the document web, discovery usually starts a search engine even though hierarchical catalogs of information such as offered by portals like Yahoo! offer well known advantages such as immediately displaying to the user the categories of items it knows about, for enabling exploration [17]. Indeed, so far data set discovery is balanced between portal and search approaches. In Web 2.0 (non-linked data) many data sets live at the service ManyEyes[15]which hosts a catalog of contributed datasets, and offers search on these datasets by tags; getting to the raw data or its visualization is within short click distances. Datasets hosted by Google Public Data[16], not surprisingly are retrieved as top hits to Google searches that contain terms that match a data set's description,

[15] http://manyeyes.alphaworks.ibm.com/manyeyes/
[16] http://www.google.com/publicdata/home

e.g. "co2 emissions for the UK". Such a hit consists of a small thumbnail visualisation of the data, a short description and link to data in Google Public Data. In each case we see that effective discovery of data sources offers (1) making it quick and easy to find interesting datasets around particular topics of interest and (2) the process from discovery to getting to the actual data needs to be quick in order for users to start interrogation i.e. unpack what a particular dataset contains.

In linked data, voID stores[17] [18], serve as access points for voID descriptions of various datasets. voID describe a dataset using a number of dimensions: the size of the dataset, the vocabularies used, and the dataset subject. Additionally, voID descriptions are labeled with DBPedia concepts to help users more easily identify the domain of the data set. However, the voID store currently supports limited browsing of these voID descriptors - via URI links or performing SPARQL queries, making it difficult for users like Susan who cannot easily browse all available voID descriptors by clicking each one individually, or that requires her to know SPARQL. Furthermore, the effectiveness of voID descriptors towards providing adequate cues to the casual user looking for a relevant dataset to her question has yet to be evaluated. Search engines over linked data such as Sig.ma [14], or VisiNav [7] provide another discovery mechanism for finding linked data by description or label. Unfortunately, searching over the data as if searching the web of documents still generally leads to unsatisfactory results; for example, (as of June 27, 2010) a keyword search for "UK cities crime" in Sig.ma yields the result "Safest Cities in America". Thus while Sig.ma today may be a useful tool for a linked data knowledge engineer who is aware of current limitations to find some RDF for a particular topic, it would be of questionable use to Susan's quest of finding relevant datasets.

Data set triage. On the document web, once a list of pages are returned from a search query, a user may quickly triage the results, comprised of document title, descriptive snippet, highlighted terms from one's search, source, author and document type to find the documents of likely relevance. Extensive work has been done to optimize page search page snippets to help people more quickly and accurately triage results [15]. Some search engines, such as PubMed[19], additionally include faceted filtering of results, which can filter and sort by various criteria such as source, author, rankings, and date published. PubMed also lets users change the presentation of search results to include more or less citation detail. In linked data triage, deducing whether a data set contains information relevant to a person's question can be difficult to ascertain, particularly if the data set is large and heterogeneous. Snippet-similar, at-a-glance summaries of data sets, and interfaces for quickly searching, clustering elements of, and filtering data sets so that the scope and coverage of a data set can be quickly assessed are missing. A best of breed example is Microsoft's

[17] http://kwijibo.talis.com/voiD/Describer
[18] http://void.rkbexplorer.com/
[19] http://www.ncbi.nlm.nih.gov/sites/pubmed

Pivot [20] collection browser, a faceted filtering interface for structured items that provides mini-visualisations of the values of the distribution of the properties of the elements of a collection of data, which can each be used to filter views of the collection. Pivot also provides the ability to visualise statistics over the collection in various ways so that users can look examine aggregate properties of the data. These affordances are also useful for analysis and question answering, described in the next section. Representation of data also plays an important role to make the data legible for deeper interogation. For example, representing the data as spreadsheets are good for both understanding the data as well as for many analytical tasks. Spreadsheets allow users to easily sort, filter and transform data. HCI related research [10] have shown that spreadsheets are not only enabling but are also acting as a catalyst for end users supported sensemaking. As S. Hudson notes [8]: "Spreadsheets are one of the few true success stories among systems for end-user programming - that is, systems designed to allow non programming users to create computations of their own design". Of course this assumes well constructed sheets where column/row headers are meaningful.

Supporting analysis and question answering. Studies like [11] have shown that transformations over data occur repeatedly during analytical activities demonstrating that the ability to quickly transform data is a key enabler of sensemaking. Susan however is foiled in her attempts to be able to to run the kinds of comparisons she wants with the tools provided, compromising her ability to build new knowledge. There are however numerous and increasing examples of public visualization tools: in social data sharing sites such as ManyEyes, Swivel and Dabble DB new visualizations appear in the hundreds on a daily basis, showing that ordinary citizens can quickly visualize data with a few simple steps with the right tools and data representations. Some Linked Data interfaces such as Tabulator [2] and VisiNav [7] have some support for tabular representation of data. Tabulator has integrated a number of widgets that allow users to select specific property values such as longitude and latitude and time and date values to represent data on on a map and a timeline correspondingly. Similar widgets are supported by Parallax, including representing data on charts through selection of x and y coordinates on a generated table. However none of these support transformations over the data. Additionally interaction between the user and the UI could inform the UI how to display certain data. For example, social data sharing sites such as Dabble DB and NeedleBase allow users to annotate fields in the data to concepts predefined in UI to offer better services. Users can state that something is a address which the UI can then know to display on a map, or something can be stated to be a data to enable advanced querying such as searching for "this month" or "yesterday".

[20] http://www.getpivot.com/

4 Challenge 2: Making data blend

As many compelling mashup applications (such as Housingmaps, MusicMesh, WeatherBonk, TwitterVision, PopURLs) have demonstrated, real world tasks often require information drawn from a wide number of data sources. Susan's information needs do, likewise. Currently, building mashup visualisations requires writing code, domain knowledge and reconciliation expertise. In the following section we present challenges for working with heterogeneity for sensemaking.

Are agreed-upon URIs and common ontologies becoming obsolete? Euzenat and colleagues characterize the possible sources of heterogeneity at 4 different levels: syntactic (structural), terminological, conceptual, and semiotic (aka pragmatic). [5] [3] How often do each of these types of heterogeneity arise in linked data today? At the lowest of the syntactic level, the use of RDF mandates a relational graph interpretation (model) out of subject-predicate-object triples, and establishes a common set of serializations for these triples (e.g., N3, nTriples, RDF-XML). Since all RDF linked data at this basic level are represented the same way, the syntactic and basic structure is uniform and no reconciliation at this level is needed. At the upper syntactic (structural) and terminological levels, heterogeneity occurs pervasively. Although URIs were introduced to allow systems to unambiguously refer to the same concepts, instances and relations, the effect of distributed authoring of linked data is that agreement upon common URIs has been difficult/rare. Moreover, despite the linked data community encouraging the re-use of existing standard ontologies for common concepts (such as foaf for people/social networks, DCMI for documents, etc) many people hand craft their own ontologies, creating structural fragmentation. Why is this? Examining the authoring process reveals a number of contributing factors. First, re-using URIs and others' ontologies can be more difficult, or more perilous, than minting one's own. Creating a URI for a concept provides the author safety in two ways. First, if URIs are to be made de-referenceable as recommended by current linked data authoring best practices [21], an author may have no choice but to create new URIs for concepts so that they can be hosted under a domain they control. Second, because by "minting" a new URI for a particular concept, an author leave it up to the consumer (user) of the data to determine whether this set of statements generalize and are consistent with others' conceptualizations of the same entity.

The decision to reuse an existing common ontology versus crafting one's own faces similar considerations. A person wishing to publish data as linked data starts with some source representation, such as a relational database or a series of spreadsheets, created with some set of underlying assumptions and/or an intended purpose. She has the option to search for a common ontology that closely matches the domain and assumptions underlying the data she wishes to publish, or to create her own. Creating an ontology will allow her to capture more fidelity than a common ontology as she can express and represent

[21] Linked Data Tutorial: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

specific idiosyncrasies of her data in her ontology. She can also reflect conceptual and semiotic differences of her data - reflecting in her ontology the context surrounding how the data was captured, and intended use. Finally, it affords her the greatest control over the interpretation and future evolution of the interpretation of her data as she sees fit. Her other choice, to re-use an existing common ontology offers only one significant advantage: it reduces heterogeneity and thus makes the data easier to consume. Thus the choice is between making it convenient for the author versus that of the consumer. One argument suggests that people are self-interested, and thus authors will ultimately choose their own needs over those of the consumer of the data, particularly when the specific target consumer(s) and use(s) are unknown. A stronger second argument advocating fidelity (heterogeneity) over convenience of consumption, put forth by the aforementioned best practices, is that resolving heterogeneity issues is ultimately easier than re-gaining fidelity lost by projecting data imperfectly onto a common ontology. With these distinct author-centric advantages surrounding creating-your-own URIs and ontologies for publishing your data, will common ontologies and truly universal URIs becoming obsolete? If so, solving structural and terminological heterogeneity needs to be a priority for all applications of linked data where such heterogeneity poses a problem. Next, we focus on the case for end-user exploration of linked data.

Heterogeneity and its effects on linked-data exploration. Structural and terminological heterogeneity manifest in a data exploration interface can directly interfere with a user's ability to effectively navigate, understand (summarize), filter, and analyze combined data sets. Missing co-reference information, for example, becomes manifest in fragmentation, redundancy and misplaced relations: if, for example, an interface does not know that two instance URIs co-refer, the instances will become presented as two separate ones (which can be mistaken by the user as two separate entities), each with a subset of the true number of relations. Thus, this very simple flaw will make it impossible for an interface to accurately count (aggregate) instances due to redundancy, determine connectivity or which instances actually possesses certain properties, due to missing relations. Likewise, if an interface does not know how to reconcile the structural differences between instances of the same (conceptual) type, then these instances become very difficult to compare. For example, take a dataset comprised of an individual's friends, represented in three different, but similar ontologies: vCard, FOAF, Apple Addressbook, and Facebook's OpenGraph. Without a mapping between like properties, a faceted filtering interface will be able to accurately filter entities that have particular properties. Furthmore, structured querying and statistical aggregation operations will be unable to identify the appropriate properties, making impossible queries such as "what is the average income of individuals in this neighborhood?" or "where do the majority of students at university live?" Nearly every technique for structured query, data aggregation, summarisation, visualisation, and navigation thus become vulnerable to unresolved structural heterogeneity.

While structural and terminological heterogeneity result in inconsistencies and incomplete linkage, conceptual heterogeneity in an exploration interface yields a problem of "apples and oranges" - two or more sets of representations that are not directly comparable at all. An example are two datasets which model geography differently - a dataset of physical geographic features versus geopolitical boundaries, for example. Since entities from two or more such data sets would not have any simple correspondence (mapping), combining such datasets is extremely difficult - scaling proportional in complexity to the number of pairwise relations between instances as number of such representations increases. Such heterogeneity occurs quite often in certain domains (such as geographic boundaries, representations of time, data sets containing measurements of scientific data) and strategies for dealing with this heterogeneity vary widely. It may be possible to relate two different conceptual representations of the same domain using a vocabulary of comparative relations - for example, based on containment and overlap. But creating such descriptions for an arbitrary pair (or set) of conceptually heterogeneous datasets may require substantial domain knowledge, which is currently outside the short-term foreseeable future. In particular, none of the emerging approaches at automatic alignment or conversion work for this type of heterogeneity, suggesting that the only possible approach to deal with this type of heterogeneity may be to surface it to the user.

Best of breed: User-directed approaches to resolving heterogeneity. Much work from the database, information extraction (IE) and semantic web research communities have sought automatic algorithms for resolving terminological and structural heterogeneity, specifically allowing instances and structures from one ontology to be transformed and re-expressed in another. These algorithms work on pairs of ontologies, and examine surface level similarities among ontological elements: names and structural similarity among classes, relations and instances. Despite significant progress, and the promise that such algorithms hold towards facilitating data set exploration, the area remains an open area of research; accuracy varies widely and is often dependent on characteristics of the particular ontologies being paired.In addition to these automatic approaches, number of systems have sought to make easier the manual reconciliation of terminological and structural heterogeneity. In some cases, these systems are used in conjunction with some automatic matching to let users easily verify the output of these algorithms. These systems let users view and work with instances from one or more data sources and modify them en masse in various ways. These systems employ various metaphors and techniques to enable mass editing - some a visual programming language, others employ a spreadsheet metaphor, while three of the most flexible systems also incorporate programming-by-demonstration (PBD) techniques. A list of systems and approaches are visible in Table 1.

System	Description	UI Approach	Output
SIMILE Potluck [9]	Heterogeneous instance alignment tool for Web 2.0 sources: Allows collections of structured instances from 2 ontologies to be aligned through simple drag and drop	Programming by demonstration; user manually takes examples and system generalizes	Instance document with merged collection of elements; some correspondences between properties
Freebase Gridworks	Freebase data clean-up tool that provides facilities for mass-editing and linking of instances	Spreadsheet with mass (parallel) editing and textual reference detection; Programming by Demonstration	Freebase instances
ITA Needlebase	Page and deep-web scraper based on templates	Spreadsheet with semi-automatic duplicate detection and domain-specific canonicalisation	JSON
uberblic	KB creation tool that facilitates importing data from heterogeneous external sources	Graphical ontology editor but requires manual textual specification of SPARQL-like patterns to extend system to merge in new data sources	KB and exploration interface for it
Yahoo Pipes!	Generic RSS/Atom feed-processing programming environment	Visual programming language	Feeds with processed (merged \filtered \transformed) elements
Intel Mash-Maker [4]	Web 2.0 (REST/JSON) Mashup creator	Graphical mashup creator but requires manual functional specification and patterns	KB, and visualisations that use mashed up data

Each of these interfaces support reconciliation of different types of heterogeneity and differing interface mechanisms for manipulating data which each vary considerably in required use effort. First, at the lowest level of manipulation, Yahoo Pipes! is a visual programming language for manipulating data streams syntactically - which is equivalent to writing a script using drag and drop operations. While creating scripts in Pipes! requires considerable effort, such authoring can be completed collaboratively, and are results are shared online for public use. Several of the other systems also provide the ability for end-users to syntactically modify all datatype property values en mass: Potluck and Gridworks are powered by Mass Edit a programming-by-demonstration approach of first clustering all datatype values by similar structure, and letting the user edit entire clusters simtulaneously. Needlebase uses a similar clustering algorithm to facilitate simultaneous parallel editing. With respect reconciling terminological inconsistency, Potluck, Gridworks, Needlebase and uberblic support alignment of properties - although the former three provide easier interfaces for doing so; drag and drop as opposed to editing a textual query modification. Gridworks, Needlebase and uberblic also support merging like entities (instances) by simple drag and drop or click-to-confirm dialogues. The approaches offer a number of advantages to purely automatic approaches to transformation. First, ultimately human users are likely to be much more equipped in general to handle reconciliation at various levels of difficulty - from terminological to the semiotic/pragmatic for the foreseeable future. Second they afford a greater degree of control, letting users inspect and understand the structure of the data, as well as the opportunity to modify an ontology or instance representation as they see fit. But the

use of such tools immediately opens up a number of questions, the first of which is, letting users easily modify and derive new representations is convenient for the user - but what happens when everyone does this for each of their applications? Each original source representation becomes fragmented and mirrored into a virtually infinite set of similar, derived representations. Moreover, despite the functionality that these systems provide, reconciliation using these interfaces can still be quite tedious, error prone and requires considerable effort. Even in the best case- that is, by an expert user, of a PBD-accelerated interface (e.g., Gridworks, Potluck or Needlebase), simple terminological and structural alignment can take minutes away from the user's primary task. Similarly, the effective use of a majority of these tools requires a learning curve which may be too steep for all but extremely experienced computer users. Finally, it is likely that a large number of users will simply not have the necessary knowledge to carry out certain alignments – establishing correspondences and ontological conversion assumes knowledge pertaining to how each dataset was authored, its limitations and intended use.

Taking user-directed resolution to web scale: future directions. The aforementioned problems can be amortized simply by letting users easily share the alignments they create. The success of Yahoo! Pipes owes much to this, as the site makes it easy for users to start with someone else's pipe and modify it, reusing the original author's work. In order for a user's alignment efforts to be applied to future data using the same source ontologies, the author would have to preserve the alignments and transformation operations performed, rather than merely the transformed data. While Needleworks and uberblic preserve these mappings, the others do not explicitly support the exporting of them. A strategy often applied by humans to cope with conceptual heterogeneity is to identify common relations between them and describing each of the respective entities in the incomparable representations in terms of these common links. For example, although a "B.A. in Computer Sciences" degree from a university in the United Kingdom is technically different from a "S.B. degree in Computer Science" from a university in the United States, they are treated equivalently in most circumstances, such as determining a person's qualifications, for example. To support the user in such a strategy, an interface could first identify candidate join points automatically (by structural/terminological matching on property values) and then facilitate comparison of the membership of these relations graphically among concepts. To make the interaction usable, many classes of users performing common tasks will not need to know the difference. In fact, when such differences are manifest across a large number of relationships, these irrelevant distinctions will dilute linking, and, much like terminological inconsistency, will make exploration for these users extremely difficult if not impossible. Displaying such entities as their true identities is useful in some cases, though, such as when the data is being authored or edited. An application needs to address such use "perspectives" which, when added to a representation of particular data, transforms its representation by asserting task-relevant equivalences. These equivalences are, in turn, supplied by end-users.

5 Discussion: Making It Real

In order to support the challenges of viewing heterogeneous data and providing intuitive interaction to create a useful and usable generic linked data browser, realtime performance and data scalability need to be achieved. Studies have shown that users on average abandon loading web pages if they take over five seconds to load thus realtime performance is critical to a successful linked data browser. In order to support realtime performance, backend query systems must provide a high level of scalability. We proceed by introducing the web interaction paradigm and detailing related work.

Retrieving data from the web. On the web a browser presents a web page to a user and allows them to click hyperlinks to external pages, which are then retrieved and rendered. The web's interaction paradigm offers benefits such as the guarantee that information is up-to-date, because pages are retrieved on-demand from their publisher, and that there is no requirement for a user to have a locally hosted cache. These benefits come from the fact that web sites are simple renderings that are attached to back-end processes, which can vary in complexity from static web pages to international banking and shopping systems that perform complex logic on data before being rendered and delivered to the user's browser.

Retrieving linked data from the Semantic Web. Using state of the art linked data browsers, the paradigm of browsing is the same: when you load a linked data browser at a FOAF file, it shows where someone lives, who their friends are and so on. The user can then click their friends, load their FOAF file, and see whey their friends work, one by one. However while this allows a user to see renderings of individual data files, it does not allow questions to be asked over all of the links such as "where do all my friends work?" or "how many of my friends work for the University of Southampton?". Due in part to a reiteration of how the web works, there is an expectation that semantic web applications will work the same way, and offer the same affordances, without taking into account the reality of querying across large number of sources. The likely cause of this is that, as pointed out here, that supporting centralised systems is seen as a bad idea, since the scalability benefit of the world wide web lies with its decentralised architecture, and that this should be supported throughout the Semantic Web too, in order for it to scale as the web did. This goal is sensible and required, but it does not mean that a linked data browser can't pull in multiple sources, much like a price comparison website does on the web, because linked data isn't a renderable human view, it's more than that: it's a machine readable model of some data. This means that seeing a linked data browser as we see a web browser isn't necessarily correct, and that it's rather more suitable to let machines read the data first, and then browse the outcome of that process.

Beginning an exploration: Discovering relevant sources. An initial challenge for a browser is therefore how to get at the data. Linked Data is by

its very nature distributed over a number of files, as well as between different data providers, as on the Semantic Web. There are efforts to provide distributed directories of linked data, and ways to query the linked data of a single site in a standard way. Typically, SPARQL endpoints are set up by data providers, or by third parties (in particular in the case of the BBC /programmes, where they do not provide a SPARQL endpoint themselves, and Talis have taken up a position of offering a public endpoint over this data), although discovery of a SPARQL endpoint is an open issue [22] where proposals of a *well known location* such as "/sparql" have been proposed. In addition, a voiD description of a dataset also aids in discovery, providing a link to sparql endpoints, as well as a directory of metadata about the dataset contents, such as the URIs it has data for. Another such approach to querying a whole provider's dataset has been focused on use by user interfaces, with the contribution of the linked data API [23] which offers a bridge to a dataset via simple HTTP calls which return JSON data that is simple to understand by humans and parse by web2.0 applications.

Querying across sources. Of course, these contributions do not concern federated querying across data publishers, so that if a user wanted to, for example find musical groups that played in a BBC programme, and then use DBPedia to find their collaborators, queries over both the BBC /programmes dataset as well as the DBPedia [1] data set are required. The situation is further clouded when the question of leveraging a user's personal data is mixed in — because while it's possible to assert a number of public datasets into a single knowledge base (as a number of public services such as uberblic has done), a user will not, in general, wish to have their private data sent to a public service to enable them to query it, and so this solution is not acceptable, even if we assume that it would be scalable in the first place.

Creating a Scalable Browser. Thus, a challenge for a browser is how to leverage a rich linked data resource to do more than simple one-file rendering, to allow rich queries to be performed. Approaches vary, from Tabulator's approach of live gathering raw RDF from the web, and feeding it into a client-side "bucket," to the server-side approach as in Virtuoso's SPARQL engine that can automatically crawl RDF based on queries. Some operations, such as rendering individual entities, can be performed entirely on the front end, since data about a single entity is typically small enough to be processed by the client. However, other types of operations require a larger amount of data to be queried, and as such, for some sizes of dataset it would be inefficient to transfer all relevant data to the client for the purposes of performing an aggregate task, such as counting — on some large sizes of datasets it would even be intractable. Counting entities that match certain criteria is the kind of aggregate operation that a server-side database or triplestore is optimised for, and as such it is reasonable to offload

[22] `http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenDataGoodPractice`

[23] Linked Data API: `http://code.google.com/p/linked-data-api/`

such an operation to the server, and return only the result(s) to the client. Similarly it is possible to sort data on the server and return a limited viewport over the results, for example if Susan loads a dataset containing 4000 possible apartments, it isn't necessary to load the full information for each one, since only a limited number can display on her screen at once. This can be seen in web search engines like Google, where results pages are paginated. Similarly, an intelligent scrolling system could support loading the top 100 (or any N) results, and load more as she scrolls down the list, as in the mSpace faceted explorer [12]. Likewise if she then reorders by price or by distance from school, the top N that are loaded will change, but the full dataset does not need to be downloaded.

Query mechanisms: The state of SPARQL. This leads to the question of how a UI should/could technically ask for limited information from a dataset. Typically, state of the art Semantic Web tools query knowledge bases using SPARQL, a query language that allows patterns over an RDF graph to be specified. However SPARQL 1.0 did not include support for operations such as counting and grouping, which are used extensively in traditional databases (through languages such as SQL), making the use of standard SPARQL when browsing quite difficult and inefficient, because more data than necessary is required to be queried from the knowledge base and postprocessed outside of the optimised triplestore code. SPARQL 1.1 (in working draft at time of writing) adds some features to enable such browsing to be possible, specifically aggregate operations such as counting and algebraic operations such as summing. In order for an application to be able to determine the possible operations that are appropriate to a dataset, it must be initially queried to determine what is possible. For example, in order to produce a histogram of instances, the interface must know the range and domain of its properties, if a histogram thumbnail (as used in MS Pivot [24]) is desired. While this is straightforward over a single dataset, performing these types of initial probing operations across sources, and combining them, has not been widely demonstrated in Semantic Web literature. Instead, data from multiple sources is downloaded and combined into a single knowledge base [13, 14, 6], which makes it feasible to produce statistics using traditional methods. A challenge here is therefore to determine if it is possible to run a, perhaps map/reduce style, of statistical data querying over remote sources, so that we can avoid having to load the information into a single place, and whether this can be done using standard SPARQL queries, or if a recommendation to the W3C working group is required. In order to further explore this subject, it is necessary to know the operations that need to be supported. Good examples of previous UIs support "semantic zooming" of large data sets, by showing distributions of values, tag clouds and histograms. For example, a summary to display a histogram of the years of birth of people from multiple SPARQL endpoints, coming with a number of local private data sources (such as facebook friend data and management hierarchy information).

[24] MS Pivot: `http://www.getpivot.com`

6 Conclusion

Our motivation in this paper has been to develop an understanding of the strengths and weaknesses of our community's approach to date in delivering tools that will fulfil the promise of linked data as an empowering mechanism for citizen knowledge building. We have therefore used standard methods and sensemaking approaches to consider requirements for such tools and to use that analysis for an assessment of both best of breed approaches and work specifically in this space to date. The contribution of this paper is that our findings can be used by the community as additional information for focusing development plans for addressing heterogeneous integration and presentation by checking "Will this work for Susan?"

Acknowledgements This work was supported by the EnAKTing project funded by the Engineering and Physical Sciences Research Council under contract EP/G008493/1.

References

1. 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*, pages 722–735, 2007.
2. T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.
3. P. Bouquet, F. Giunchiglia, F. Harmelen, L. Serafini, and H. Stuckenschmidt. C-owl: Contextualizing ontologies. *The Semantic Web-ISWC 2003*, pages 164–179, 2003.
4. R. Ennals, E. Brewer, M. Garofalakis, M. Shadle, and P. Gandhi. Intel mash maker: join the web. *SIGMOD Rec.*, 36(4):27–33, 2007.
5. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag New York Inc, 2007.
6. H. Glaser and I. Millard. RKB Explorer: Application and infrastructure. *Proceedings of Semantic Web Challenge*, 2007.
7. A. Harth. VisiNav: Visual Web Data Search and Navigation. In *Database and Expert Systems Applications*, pages 214–228. Springer, 2009.
8. S. Hudson. User interface specification using an enhanced spreadsheet model. *ACM Transactions on Graphics (TOG)*, 13(3):239, 1994.
9. D. F. Huynh, R. C. Miller, and D. R. Karger. Potluck: Data mash-up tool for casual users. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):274 – 282, 2008. Semantic Web Challenge 2006/2007.
10. B. A. Nardi and J. R. Miller. An ethnographic study of distributed problem solving in spreadsheet development. In *CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pages 197–208, New York, NY, USA, 1990. ACM.
11. M. Russell, R. Jeffries, and L. Irani. Sensemaking for the rest of us. Sensemaking Workshop at CHI (2008), 2008.
12. M. schraefel m.c., Wilson, A. Russell, and D. Smith. mSpace: improving information access to multimedia domains with multimodal exploratory search. *Communication of the ACM*, 49(4):47–49, 2006.
13. N. Shadbolt, N. Gibbins, H. Glaser, and S. Harris. CS AKTive Space. *IEEE Intelligent Systems*, 19(3):41–47, 2004.
14. G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig. ma: live views on the web of data. In *Proceedings of the 19th international conference on World wide web*, pages 1301–1304. ACM, 2010.
15. A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams. Fast generation of result snippets in web search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 127–134, New York, NY, USA, 2007. ACM.
16. M. L. Wilson, m.c. schraefel, and R. White. Evaluating advanced search interfaces using established information-seeking models, 2007.
17. K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM.