Potluck: Semi-Ontology Alignment for Casual Users

David F. Huynh, Robert C. Miller, David R. Karger

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar St., Cambridge, MA 02139, USA
{dfhuynh, rcm, karger}@csail.mit.edu

Abstract. Potluck is a web user interface (Figure 1) that lets casual users—those without programming skills and data modeling expertise—repurpose heterogeneous Semantic Web data. It lets users merge, navigate, visualize, and clean up data all at the same time, using direct visual manipulation. This iterative process of integrating the data while constructing useful visualizations is desirable when the user is unfamiliar with the data at the beginning—a common case—and wishes to get immediate value out of the data without having to spend the overhead of completely and perfectly integrating the data first.

Keywords: mash up, drag and drop, faceted browsing, simultaneous editing, ontology alignment, end-user programming, semantic web, RDF.

1 Introduction

A central theme of the Semantic Web is of casual users collecting, merging, and repurposing information from multiple sources. But most tools that support this vision—descriptive logics, reasoners, and graph-based visualizers—posit data that has extensive schema information, and are far too sophisticated for casual users. We describe Potluck, a tool that allows casual users—who may not even know what an ontology is—to collect *messy* semantic web data from multiple sources, merge it into a coherent ontology, and visualize and navigate it effectively.

Potluck provides an *instant gratification* demonstration of the Semantic Web's benefits. Users who find multiple useful sources of semantic web data can instantly merge them into a single blob in their web browser. That the data is structured means faceted navigation and structured visualization of the data are immediately available. Then, rather than thinking hard about proper ontologies or writing functional descriptions of data transformation, users visually manipulate the data until it *looks right* for their purposes. Through direct manipulation, they align ontologies (unknowingly), clean up instances, create useful facets for navigation, and produce useful merged views for visualization. The end result can be saved as RDF that another user can consume for yet other purposes. Potluck eliminates the artificial separation between understanding and cleaning the collected data and using it to solve the user's problem, letting her tackle both at once.

Unlike automated alignment tools that rely on extensive ontological markup, Potluck works with "raw" RDF—it needs no schematic information beyond what is implicit in the properties used in the data. This offers several benefits:

- Far more raw RDF is available than "cooked". For example, DBpedia contains almost 100M triples, but almost no schema information. Users can use Potluck to interact with a subset of this data.

- By demonstrating that users can interact with raw RDF, Potluck shows that publishing Semantic Web data is easy: authors can publish data instances without having to picking the right ontologies.
- Even providers who *want* to create nice ontologies are likely to do it "wrong" for some users. Potluck lets these users correct the "errors" themselves.

Potluck is therefore an effective deployment tool for the Semantic Web, answering the well known "chicken and egg" complaint that good semantic web tools and good semantic web data are stuck waiting for each other.

2 Scenario

Before describing the user interface of Potluck, we motivate it with a scenario that illustrates various idiosyncrasies of personal mash-up construction. Let us be optimistic that within a decade, the Semantic Web will be prevalent and RDF data will be everywhere. This scenario argues that even in this future world, users will *still* face problems making mash-ups between data sources.

In 2017, a historian named Henry is documenting the first cases of a rare genetic disease called GD726. These first cases occurred in the Valentine family in the 1820s. He wants to include in his final report a genealogical tree of the Valentine family, annotated with the disease's infliction, as well as a comprehensive table of the Valentines' data in an appendix.

Like most historians, Henry is not a programmer but he is experienced in managing data in his work. The proliferation of RDF means that he does not need programming skills to scrape HTML himself: all information needed for his research has been converted into RDF by various independent organizations and individuals. Henry thinks it would be trivial to pool the RDF together and call it done.

Henry tracks down various birth certificate and death certificate issuing offices where the Valentines lived for their RDF data. Some offices use **`dc:date`** in their data to mean "birth date," some to mean "death date," and some "certificate issuing date." Thus, considering all **`dc:date`**s the same would be disastrous.

Henry also tracks down hospital records, which contain **`hospital:tod`** (short for "time of death"). Hence, **`hospital:tod`** is equivalent to some of the **`dc:date`**s. It would be hard to match **`hospital:tod`** with **`dc:date`** based on string analysis alone, yet match for some of the cases only.

The records all have geographical location names, but these names are not fully qualified. Those responsible for digitizing them thought that since all locations were within their country, the country name was redundant. Consequently, Henry needs to append the country name to all location names in order to map them.

People's names are encoded in two different forms: "first-name last-name" in some data sets and "last-name, first-name" in others. Nick names are also present (e.g., "Bill" instead of "William", and "Vicky" instead of "Victoria").

The hospital records also pose problems. While most of their admittance dates are in ISO 8601 format, a few are of the kind "Easter Day 1824." Such sloppiness has been observed in industrial and institutional databases, and should be expected on the Semantic Web.

Despite all these problems, there is one good thing about the data: Henry can reliably get the mother and father of each Valentine through the **`gen:mother`** and **`gen:father`** predicates, which seem to be very widely adopted. This helps Henry construct a genealogical tree visualization. However, as males and females both have equal chance of passing on GD726, Henry wants to treat **`gen:mother`** and **`gen:father`** the same while tracing the disease through the family. Unfortunately, adding an **`owl:sameAs`** equivalence between those two predicates will break his genealogical tree.

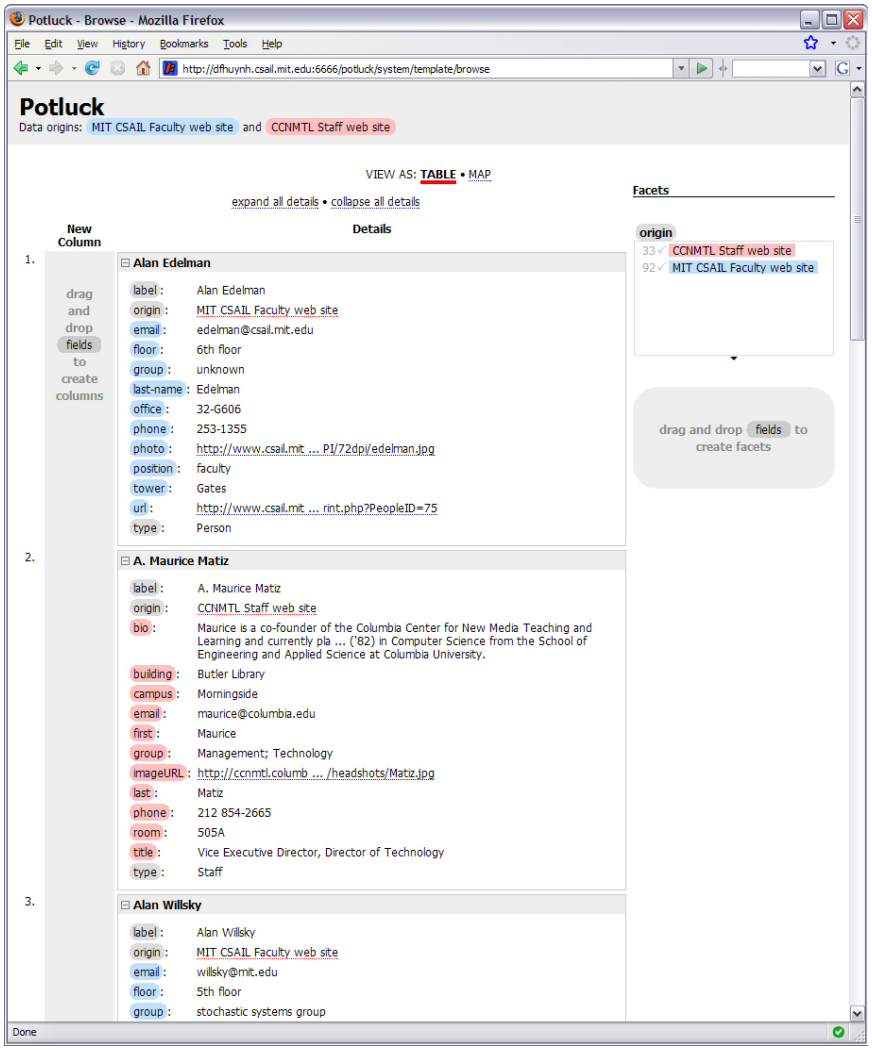

Figure 1. Potluck's user interface shows data mixed from two different sources. Fields are rendered as draggable "field tags," color-coded to indicate their origins.

While all parties involved in this scenario acted logically and responsibly, Henry still ends up with a mess of RDF. To fix up the data, Henry must be able to:

- Merge **`dc:date`**s into *several* groups (the birth dates and the death dates) even though they all use the same predicate URI.
- Merge **`gen:mother`** and **`gen:father`** together in some situations while keeping them separate in other situations. This precludes the simple approach of adding **`owl:sameAs`** statements in the data model to implement equivalences.
- Edit the data efficiently to unify its syntax.
- Fix up the data iteratively as he learns more and more about the data.

3 User Interface

We now describe Potluck's user interface, showing how it addresses the problems in the scenario above. The reader is encouraged to view a screencast to understand Potluck's interactivity: http://people.csail.mit.edu/dfhuynh/research/media/iswc2007/. Potluck starts up with a text box where the user can paste in several URLs of Exhibit-powered web pages and click a button to yield the results in Figure 1, which lists data records from the original web pages. The records are interleaved by origins—the pages from which they have been extracted—to ensure that some records of each data set are always visible.

Fields are rendered as *field tags*: label, position, and title. Field tags are color-coded to indicate their origins: blue from one source and pink from another. Three core fields, **`label`**, **`type`**, and **`origin`**, are assigned to all records and their tags are colored gray. Fields from different origins having the same name are considered different—a crucial choice for **`dc:date`** in the above scenario.

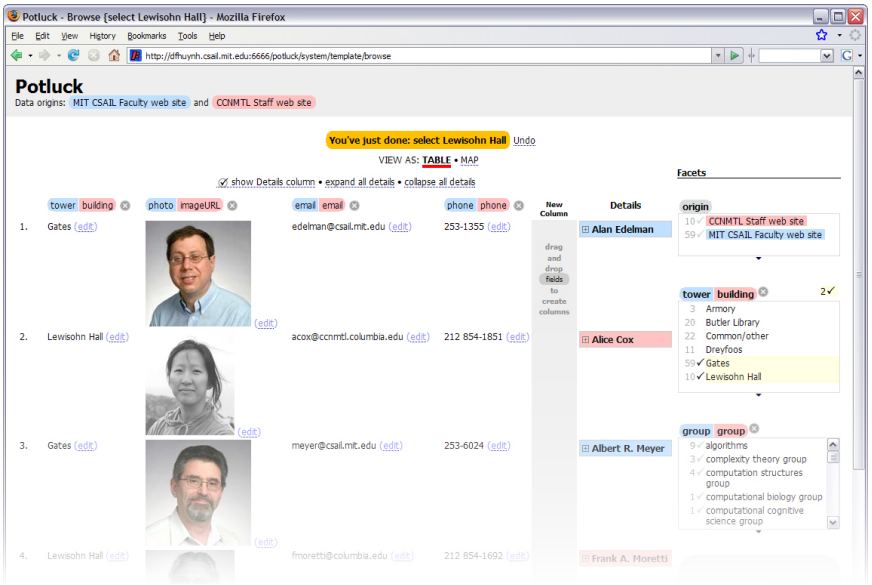

Figure 3. A screen shot of Potluck showing several columns and facets of merged fields. The records' details have been collapsed to make space for the columns.

Creating columns and facets. A field tag can be dragged and dropped onto the gray column to the left (Figure 1) to create a new column listing that field, or onto the gray box to the right to create a facet for filtering by that field. Figure 2 shows a newly created column. A column or facet can be moved by dragging its field tag and dropping the tag between other columns or facets. Deleting a column or facet (by clicking its ⊗) removes the column or facet from the display but does not delete the corresponding field's data.

Merging fields. A field tag can be dropped onto an existing column or facet to make that column or facet contain data for both the original field and the newly dropped field. Such an operation creates a *merged field*, whose field tag is rendered as a visual juxtaposition of the original tags in a pill-shaped form position title. Figure 3 shows several columns and facets of merged fields. Merged field tags can be dragged and dropped like elemental field tags can to create new columns and facets, or to merge into other existing columns and facets.

Creating a merged field does not disturb the elemental fields. Thus, in the scenario, it would be easy to have **`gen:mother`** and **`gen:father`** merged together for one purpose while keeping them separate for another purpose, all at the same time. Furthermore, the merging operation is *not* transitive, so that, say, merging fields **`mother`** and **`father`** together (to mean **`parent`**) and then **`mother`** and **`grandmother`** together (to mean **`female ancestor`**) does *not* force all three fields to be merged into **`mother/father/grandmother`**.

Simultaneous editing. The edit link next to each field value opens up the Simultaneous Editing dialog box where the values of that field can be edited *en masse* (Figure 4). The concept of simultaneous editing originated from LAPIS [6], a text editor that displays several keyboard cursors simultaneously on a text document, generalizes the user's editing actions at one cursor, and applies them to the text at the rest of the cursors. Based on the user's mouse clicks, LAPIS guesses how to divide the text document into records (often into lines or paragraphs) and where the cursors should be placed within those records (e.g., after the second word of the third sentence in each paragraph). Whereas LAPIS has to guess what a record is for the purpose of simultaneous editing, Potluck already has the field values conveniently separate. Potluck groups field values into columns by structural

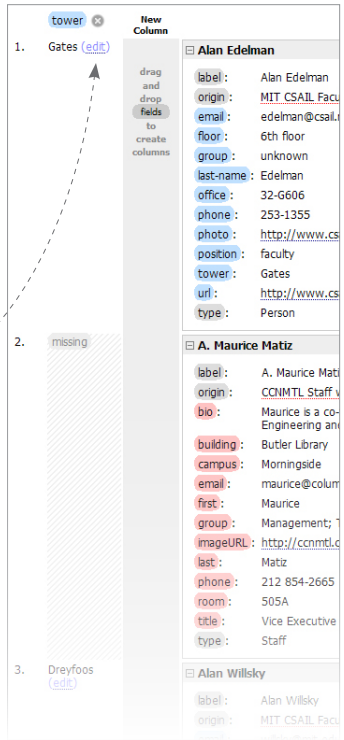

Figure 2. Potluck renders a new column to the left when tower is dropped into the New Column drop target. Since the second record is not from the same origin as the dropped field, its cell in that column shows missing.

similarity, e.g., the phone numbers in the second column all have area code 212. These columns serve to visually separate out values of different forms, call out outliers (such as “Easter Day 1824” in the scenario), and let the user edit different forms differently. The user can click on any field value to give it keyboard focus, and editing changes made to it are applied to other values in the same column in a similar way. The multiple cursors in Figure 4 give visual feedback of the simultaneous editing operations in progress.

Simultaneous editing is useful for correcting inconsistencies between data sets that occur many times, such as prefixing area codes to phone numbers; for reformatting a field, such as changing “first-name last-name” into “last-name, first-name”; and for making a new field out of an existing field, such as extracting building numbers (32) from within office numbers (32-582).

Faceted browsing [8] lets a set of records be filtered progressively along several dimensions in any arbitrary order. It is useful in Potluck as Potluck often handles multidimensional data. Exploration through such data is needed for selecting out just the desired subset and for isolating records that need cleaning up.

Potluck extends faceted browsing for the mash-up task in which data arrives from many sources. First, if within a facet there are records missing the corresponding field, the facet explicitly shows a choice for filtering to them (Figure 5). This visual element, not present in conventional faceted browsing interfaces, also serves to remind the user that, if that field is an elemental field instead of a merged

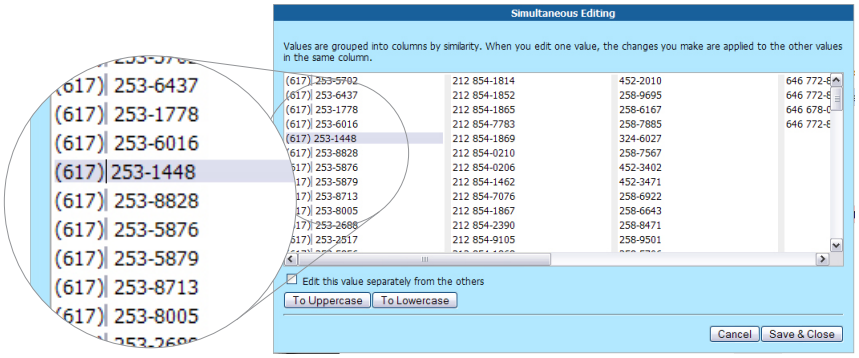

Figure 4. The Simultaneous Editing dialog box lets the user change several similar values simultaneously by editing any one of them.

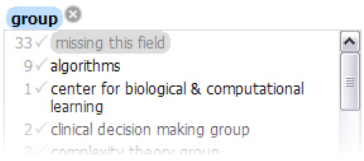

Figure 5. If inside a facet there are records missing the corresponding field, the facet shows missing this field as a choice to get to those records.

Figure 6. The origin facet does not remove choices for which there are no records. Moreover, it pops up messages to call the user’s attention to those filtered out origins.

field, the field is not present for records in other data sets. Second, whenever a facet choice causes all records from an origin to be filtered out completely, that origin remains in the origin facet and a warning message is displayed (Figure 6).

Visualizations. Potluck currently provides two visualizations: a tabular view and a map view. Figure 7 shows the map view in which any field containing street addresses or latitude/longitude pairs can be dropped onto the map view to plot the records. The map markers can also be color-coded using drag and drop. Faceted browsing is supported concurrently so that the user can construct a map while browsing through the data at the same time.

4 Related Work

We discuss related work at greater length in our ISWC paper [4]. End-user tools such as Dapper [1] scrape information from multiple HTML pages. The proliferation of structured data on the Web will hopefully eliminate the need to scrape fragile HTML, allowing tools such as Piggy Bank [3] and Tabulator [2] to collect it from multiple already-structured Semantic Web sources. Even so, the data still has to be cleaned up and aligned before it can appear coherent to the user and thus become useful. The amount of broken HTML code on the present Web forebodes messy real-world RDF in the future, broken perhaps not just in syntax but also in semantics. Previous tools have largely ignored this problem, assuming that once data is in RDF, visualizations and browsing techniques previously designed to work on individual coherent data sets can be applied readily on mashed up data.

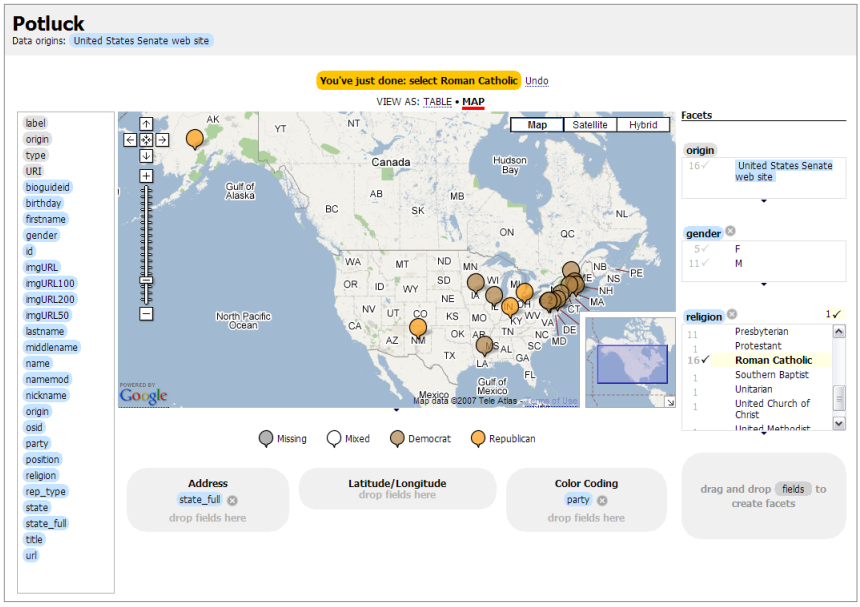

Figure 7. Potluck's map view allows plotting and color-coding records by dropping field tags into drop target areas. Faceted browsing is also offered during map construction.

An exception to these tools is WebScripter [7], which offers casual users data alignment features to create coherent tables of data collected from several sources. However, WebScripter does not encourage users to browse and visualize the data while aligning it, and offers no features for fixing data at the syntactic level, and it has not been formally evaluated on actual users.

Research data-alignment tools have been built mostly for experts and research has focused primarily on data modeling theories and automated agents for ontology alignment (surveyed in [5]) rather than on user interfaces for making practical use of aggregated data. They implicitly assume that users work with the data in delineated stages, first aligning the data and cleaning it up, and then using it in some other tools. We believe that users actually work iteratively on data, switching from aligning and cleaning up the data to using the data, and back, as they get to know the data better over time. Potluck explicitly supports this approach.

5 Conclusion

This paper presented Potluck, a tool for casual users—those without programming skills and data modeling expertise—to "mash up" data by themselves. Potluck is novel in its use of drag and drop for merging fields, its integration and extension of faceted browsing for focusing on subsets of data to align, and its application of the simultaneous editing technique for cleaning up data syntactically. Potluck also lets the user construct rich visualizations of data in-place as the user aligns and cleans up the data. This iterative process of integrating the data while constructing useful visualizations is desirable when the user is unfamiliar with the data at the beginning—a common case—and wishes to get immediate value out of the data without having to spend the overhead of completely and perfectly integrating the data first. It thus offers instant gratification to end users who want to *do something* with their data without stopping to solve ontology alignment problems.

6 Acknowledgements

This work was supported by the National Science Foundation (award number IIS-0447800), Nokia, and the Biomedical Informatics Research Network. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funders.

7 References

[1] Dapper: The Data Mapper. http://www.dapper.net/.
[2] Berners-Lee, T., et al. Tabulator: exploring and analyzing linked data on the Semantic Web. *SWUI* 2006.
[3] Huynh, D., S. Mazzocchi, D. Karger. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. *ISWC* 2005.
[4] Huynh, D., R. Miller, and D. Karger. Potluck: Data Mash-Up for Non-programmers. *ISWC 2007*.
[5] Kalfoglou, Y. and M. Schorlemmer. Ontology Mapping: The State of the Art. *The Knowledge Engineering Review*, Volume 18, Issue 1, 2003.
[6] Miller, R. and B. Myers. Multiple Selections in Smart Text Editing. *IUI* 2002.
[7] Yan, B., M. Frank, P. Szekely, R. Neches, and J. Lopez. WebScripter: Grass-roots Ontology Alignment via End-User Report Creation. *ISWC* 2003.
[8] Yee, P., K. Swearingen, K. Li, & M. Hearst. Faceted Metadata for Image Search and Browsing. *CHI 2003*.