Data In Context: Aiding News Consumers while Taming Dataspaces

Adam Marcus[*] , Eugene Wu, Sam Madden
MIT CSAIL
{marcua, sirrice, madden}@csail.mit.edu

...were it left to me to decide whether we should have a government without newspapers, or newspapers without a government, I should not hesitate a moment to prefer the latter.
— Thomas Jefferson

ABSTRACT

We present MuckRaker, a tool that provides news consumers with datasets and visualizations that contextualize facts and figures in the articles they read. MuckRaker takes advantage of data integration techniques to identify matching datasets, and makes use of data and schema extraction algorithms to identify data points of interest in articles. It presents the output of these algorithms to users requesting additional context, and allows users to further refine these outputs. In doing so, MuckRaker creates a synergistic relationship between news consumers and the database research community, providing training data to improve existing algorithms, and a grand challenge for the next generation of dataspace management research.

1. INTRODUCTION

One of the basic mechanisms through which end-users consume and interact with data is by reading a news source. Many articles are based on one or a few data points, be it the earnings of a company, new unemployments numbers for a country, or the number of people at a political rally. Infographics compactly present a larger set of points, typically through aggregate statistics grouped temporally or by category. Investigative efforts often uncover and join several datasets to deliver a story. In business reporting, the data is even more apparent: companies like Reuters and Bloomberg are successful in part because they generate and serve enormous amounts of data. In each of these examples, the end product—an article or a graphic—is conceptually a view over a dataset. For example, when reporting the earnings of a specific company, the article presents a view of a dataset of all company earnings in the same quarter, or earnings of the company during previous quarters.

In this light, an article's background includes tuples outside of the existing view; access to extra information would allow the reader to better understand the article's context. However, articles may miss key contextual cues for many reasons: 1) A lack of space or time, as is common in minute-by-minute reporting 2) The article is a segment in a multi-part series, 3) The reader doesn't have the assumed background knowledge, 4) A newsroom is resources-limited and can not do additional analysis in-house, 5) The writer's agenda is better served through the lack of context, or 6) The context is not materialized in a convenient place (e.g., there is no readily accessible table of historical earnings). In some cases, the missing data is often accessible (e.g, on Wikipedia), and with enough effort, an enterprising reader can usually analyze or visualize it themselves. Ideally, all news consumers would have tools to simplify this task.

Many database research results could aid readers, particularly those related to *dataspace management*. Matching records from articles with those in a relation is an entity resolution problem; aggregating contextual information from multiple sources is a schema matching and data integration problem; searching for the missing data is a deep web problem; extracting relations from web-pages and text is solved by projects like TextRunner [9] and Webtables [3]. While these efforts have pushed the limits of automated solutions, recent human-assisted approaches present new opportunities. Existing automated algorithms can be semi-automated by asking humans to vet algorithmic results and iteratively improve the algorithms over time. We believe news is an ideal match for this problem.

We believe that, given the promise of contextualized articles, readers would be willing to answer a small number of simple questions. For example, we might ask a user to highlight the data in question, identify related datasets, ensure the data in an article properly maps to a dataset, or to select a visualization. Fortuitously, each small task generates inputs for semi-automated dataspace management algorithms. Readers can also benefit from previous users' answers and view previously generated context without additional effort.

We view data in context as a grand challenge in dataspace management. If we design algorithms that are good enough to be guided by an average reader and provide the context they lack, then both society and the database community benefit. In addition, journalists can use the same tools to proactively identify relevant datasets or visualizations.

We envision a proof-of-concept user interface for an article contextualization service called MuckRaker. It serves as a front-end to the numerous database problems mentioned above. We have developed it as a bookmarklet for web browsers that allows users to select a data point on a webpage, answer a few questions about its context and origin, and see a visualization of a dataset that contextualizes their

[*] Eugene and Adam contributed to this paper equally.

reading experience. In this paper we outline:

1. A user interface for interacting with data items on the web that contextualizes them,
2. A study of the difficult dataspace management problems average news consumers can help solve, and
3. A collection of the challenges posed by data embedded in articles that span the web.

2. TALE OF TWO COMMUNITIES

The problem of contextualizing data lies at the intersection of two communities—news consumers and the database community—and can benefit both. As a social endeavor, we believe it encourages the general population to interact with data. MuckRaker not only helps curious readers better understand the news, but can help users spot biased attempts to generalize a single outlier data value as the norm. It serves the data management systems by helping clean, normalize, and aggregate data that consumers care about.

The unfettered MuckRaker vision encompasses several open problems facing the database community. We believe variants of the problem are tractable and can be solved with careful application of existing approaches. In the rest of this section, we illustrate how a constrained instance of this problem can be solved for an example article that centers on a car bombing in Iraq[1]. We first describe MuckRaker from a user's perspective, then explore a viable system implementation, and finally explore extensions to increase MuckRaker's utility and introduce interesting research problems.

2.1 A Tool for News Consumers

Suppose a user reads an article about a car bomb in Iraq and wants to know the scale of similar attacks. She clicks the MuckRaker bookmarklet (Figure 1a), which asks her to select the region of text that she would like contextualized. The user highlights the sentence "Twin car bombs exploded in central Baghdad on Tuesday, killing at least 19 people." MuckRaker extracts key entities and values around the highlighted region, and presents the data to the user (Figure 1b). MuckRaker could not identify attribute names for the first and last column, and prompts the user to fill them in. The attribute name in the third column is not precise enough, so the user can click it to edit the name. When the user is satisfied, she clicks "Find Context."

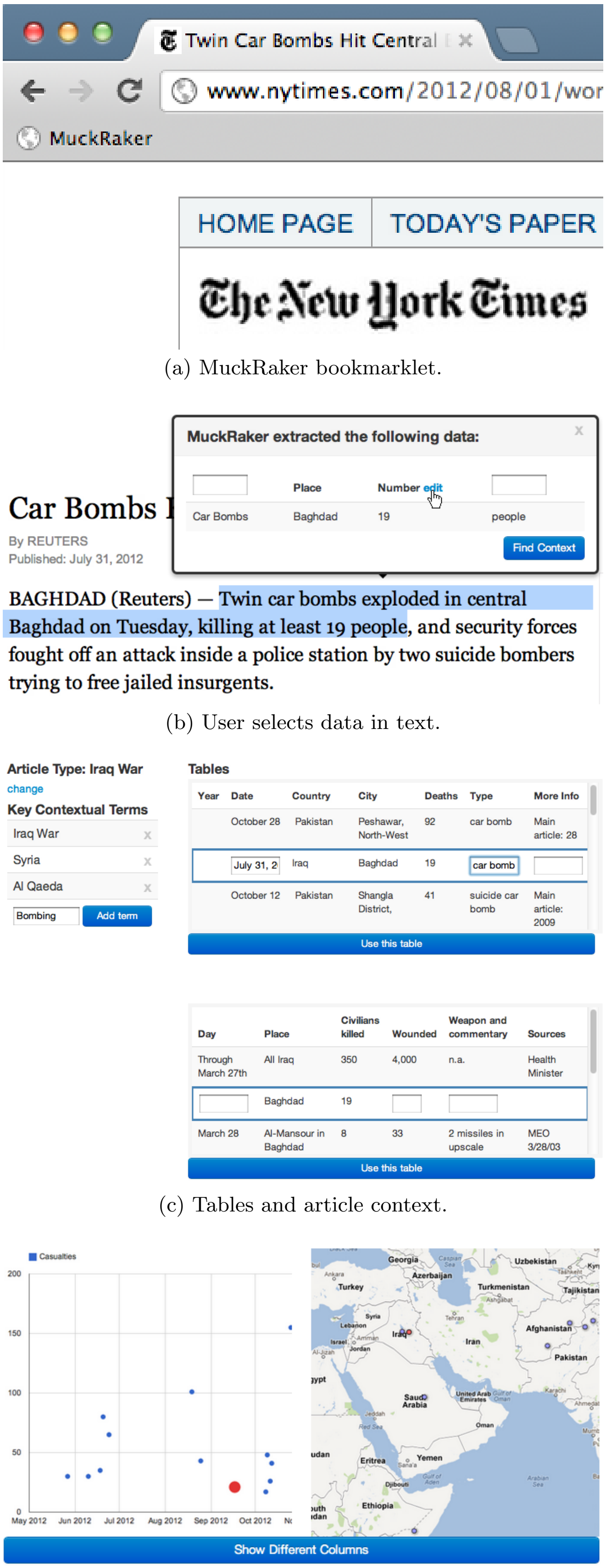
MuckRaker receives a request containing the highlights and extracted data. MuckRaker finds contextual information about the article using article type classification, fact, entity and numerical extraction, and other natural language processing techniques. This information, weighted by its distance to the selected text, is fed to algorithms described in Section 2.2. The information and the tables ranked most relevant by the algorithms are presented to the user (Figure 1c). The user can select the desired table, or update this information and re-execute the matching algorithms.

In this example, the first table is a list of mass bombings[2] and the second is of civilian casualties during the first two weeks of the Iraq war[3]. If a matching row exists, MuckRaker will attempt to populate empty fields with new data values.

[1] http://www.nytimes.com/2012/08/01/world/middleeast/twin-car-bombs-hit-central-baghdad.html

[2] http://cursor.org/stories/iraq.html

[3] http://en.wikipedia.org/wiki/List_of_mass_car_bombings



(a) MuckRaker bookmarklet.

MuckRaker extracted the following data:

	Place	Number edit	
Car Bombs	Baghdad	19	people

Find Context

Car Bombs I

By REUTERS
Published: July 31, 2012

BAGHDAD (Reuters) — Twin car bombs exploded in central Baghdad on Tuesday, killing at least 19 people, and security forces fought off an attack inside a police station by two suicide bombers trying to free jailed insurgents.

(b) User selects data in text.

Article Type: Iraq War
change

Key Contextual Terms

- Iraq War x
- Syria x
- Al Qaeda x

Bombing | Add term

Tables

Year	Date	Country	City	Deaths	Type	More Info
	October 28	Pakistan	Peshawar, North-West	92	car bomb	Main article: 28
	July 31, 2	Iraq	Baghdad	19	car bomb	
	October 12	Pakistan	Shangla District,	41	suicide car bomb	Main article: 2009

Use this table

Day	Place	Civilians killed	Wounded	Weapon and commentary	Sources
Through March 27th	All Iraq	350	4,000	n.a.	Health Minister
	Baghdad	19			
March 28	Al-Mansour in Baghdad	8	33	2 missiles in upscale	MEO 3/28/03

Use this table

(c) Tables and article context.

(d) Visualizations.

1: The MuckRaker user interface.

In this case, neither table contains the user-selected data point, so MuckRaker inserts the data into both tables, and fills as many of the matched fields as possible. The row is highlighted for the user so that she can optionally fill in missing fields. She is interested in mass bombings, so selects the first table, corrects the date field, and fills in the **type** column with "car bomb," which is autocompleted. When she clicks "Use this table,“ the updated row is sent to MuckRaker, and she is shown condidate visualizations.

MuckRaker selects an initial visualization using the selected dataset. Previous users commonly created timeseries and maps, and Figure 1d shows these top plots: deaths by date and deaths by location. It highlights the new data point to identify contextual data (the red points). The user can specify that she is interested in different columns and construct her own chart through a chart builder, where she can specify chart types, axes, filters, and facets. MuckRaker stores the visualization configuration that the user picked, and uses it to bootstrap future requests.

2.2 Database Community

We believe that data in context can be implemented at a basic but useful level using existing technology. We hope the database community can use the core system as a starting point for numerous research problems. In the rest of this section, we sketch a basic implementation, and then describe how user interactions can both improve existing algorithms with training data, and introduce new challenges that must be addressed by future data contextualizing systems.

2.2.1 Core Implementation

The MuckRaker interface is implemented as a browser bookmarklet (Javascript that operates on the current web page). We assume that we start with a reasonable collection of tables that are clean, deduplicated, and complete. In addition, table metadata includes the surrounding text (e.g., Wikipedia article text). This data can be bootstrapped from sites like Wikipedia and groups such as the world bank, or through techniques like those in work by Cafarella et al [3].

User-selected text is sent to a backend server, which extracts numbers and their units (e.g., $, miles), dates, and known entities. Entity extraction algorithms such as KnowItAll [6] can identify the key nouns and topics. We can ask the user to clean and highlight the extracted values.

The set of possible tables is filtered by clustering the article text with table metadata. For example, an article related to the Iraq war will match tables extracted from Wikipedia articles about Iraq. We can further reduce and rank tables by canonicallizing user-specified attribute names using techniques similar to those used by Das Sarma et al [5] to perform schema ranking. A final ranking comes from comparing the table values with those in the user-extracted record.

2.2.2 Research Problems

The user interaction provides a number of strong signals that make for interesting research problems. We describe some problems in the context of interactions in the extraction, selection, and visualization phases of the user workflow.

Data Extraction and Integration. The user-selected text explicitly defines record boundaries. The collection of all user highlights can be used to train classifiers to detect strings that contain records, and focus the analysis that data record extractors like TextRunner [9] need to perform.

The user can facilitate record extraction, but values may be named, formatted, or scaled inconsistently with existing tables. With a tighter human-in-the-loop training cycle, we have more hope for improving such extraction anomalies.

Another classification challenge lies in identifying the type of context that the user interested in. She may want to see an IBM earnings report in the context of historical earnings instead of similar technology companies (select the appropriate attributes). Alternatively, a European reader may prefer to see European companies rather than American companies (select the best records). Subdividing context automatically according to user preferences is a key challenge.

Structured Search. Das Sarma et al. recently studied related table search [5], where a user specifies a table and the search engine identifies others that either extend the table horizontally or vertically. MuckRaker requires a similar search, but uses partial examples extracted from article text.

In addition, identifying the table is not enough. To be useful, tables must be transformed (e.g., currency conversion), projected, filtered (e.g., identify small number of representative rows), and aggregated (e.g., aggregate county statistics to report state granularity statistics). Learning these steps is another research challenge.

Visualization. Automated Visualization selection is difficult because it is both a dimensionality reduction problem and a design problem. Historical earnings are best plotted as a time series, while the comparative earnings of similar companies is better represented with a bar chart. A human in the loop would better assist and train these decisions.

MuckRaker can gather large volumes of training data of user-perferred columns based on the final visualizations that the user selects. One project that has facilitated user-driven visualization construction is the ManyEyes project [8], and we can use its findings as a basis for our design.

Data Management. The projects that have most closely integrated these individual research problems into a larger data integration and search system are TextRunner [9] and WebTables [3]. To the extent that these projects have been evaluated by how much deep web data can be added to web search indices, we think that the grand challenge raised by contextualizing data serves as a higher bar. While indexing websites by the data they store is useful, being able to retrieve datasets that are relevant to a user's current context would be even more powerful. The WebTables authors realize this as well: Fusion Tables [7] surfaces the data found in web tables and other datasets directly in search results, suggesting that search-based structured data retrieval is a meaningful measure of the effectiveness of these techniques.

3. GENERALIZING MUCKRAKER

In the interface described above, all user actions succeeded: the user found the correct dataset, the new record was reasonably extracted, there were no duplicate records in the table, and roughly one relevant record was extracted from the article. We now consider more challenging cases, and how the user interface can be augmented to handle them.

No matching datasets. Consider a situation where the user highlights some text and clicks "Search," no useful datasets are returned. In these situations, the user can utilize the search bar in Figure 1c to enter her own keywords, and potentially find the dataset. Failing that, she can click on "I can't find a dataset," and be prompted to either: 1)

Point at a webpage containing the dataset, or 2) Specify column headings (the schema) and row that can be extracted from this document in a spreadsheet interface. In scenario 1, an automated extractor can present the user with the newly extracted dataset, and in scenario 2, MuckRaker can search again for a matching dataset with the given schema and data point. Should this final search fail, the user will be invited to add more entries to the spreadsheet.

Incorrect extraction. In situations where a dataset is correctly identified but records are extracted incorrectly, the user can edit the row in the familiar spreadsheet interface of Figure 1c. It is possible that a user will incorrectly add field values, but MuckRaker aggregates multiple user corrections before trusting any one user's input.

Duplicate data. Duplicate rows within a table can arise if multiple users submit different articles about the same event. We can handle these by calculating a similarity measure between rows. For any newly added row that is above some threshold similarity to an existing row, we can ask a user to verify that the user indeed means to add a new data point. Data duplicated across tables requires more care. We wish to know when a table should be merged with another table, which might happen when enough rows between two tables are similar. In this situation, we can ask a user during dataset search (Figure 1c) whether the table they selected is the same as another one. If the user indicates that it is, they are then presented with the columns of both tables aligned by a schema mapping algorithm, and invited to re-arrange the mapping as they see fit. The system can merge two tables if enough users mark them as merge candidates.

Article-embedded datasets. So far, our user has highlighted a sentence that roughly translates to a single record in a table. It is often the case that an article discusses more than one data point. For example, an article that describes a trend essentially embeds multiple points from a timeseries into a dataset. Alternatively, an article summarizing a study that compares multiple groups of people would embed data about each group. Summarizing all of the extracted points in a table might be cumbersome for the user. It might be simpler to summarize the extacted data in a visualization, allowing the user to drag the points of a timeseries to match a trend, or move the bars in a bar graph to represent the relative differences between groups.

Uncertain facts. It is often the case that the news covers facts that contradict one-other (e.g., "Prior link between cancer and fruit juice challenged in latest research"). Other facts might simply expire over time. For example, records that refer to "The President, aged 51" refer to a different president or an incorrect age depending on the date of the article. A user overseeing the data extraction that knew good schema design practices (e.g., storing The President's date of birth rather than an age) could have avoided some of these issues, but MuckRaker does not leave schema design to expert database designers. To handle these types of expiration and uncertainty, attaching a source and date to extracted data may help, as would periodically asking users whether certain records are still valid in a table.

4. CONCLUSION

The core contribution of MuckRaker is to utilize a mixed-initiative interface to improve dataspace management operations as a byproduct of contextualizing the news. It would be interesting to see where else the insertion of a lightweight user interface can act as a boon to database research while benefitting another community.

There have been other calls to arms in the database community to assist the journalism process. Most prominently, Cohen et al. outlined many ways in which computational journalism can be aided by database research in areas such as fact checking and hypothesis finding [4]. The PANDA project [1] aims to provide centralized dataset storage and search functionality within newsrooms. DataPress makes it easier to embed structured data and visualizations into blog posts [2]. MuckRaker approaches the journalism-data interface from a different perspective: it seeks to aid news consumers in situations where the Journalism process has left them with an incomplete picture of the world. It can also help journalists and editors preempt this problem by helping them find contextualizing datasets and visualizations.

A key question in designing the MuckRaker experience is whether the interface we are designing is lightweight enough, or whether we are asking for too much from any one user. In exchange for context behind an article, we believe users are willing to answer a few small questions, mostly through point-and-click interfaces. If it turns out that we are asking too much from each user, however, we can design interfaces that load-balance the data integration, extraction, and visualization tasks across users, especially in scenarios where multiple users are reading the same article.

In presenting MuckRaker, we hope to bridge the gap between end-users and deep data exploration. We hope that the database community is excited to improve on its algorithms with help from the average news consumer.

5. REFERENCES

[1] The PANDA project, August 2012. http://pandaproject.net/.

[2] E. Benson, A. Marcus, F. Howahl, and D. Karger. Talking about data: Sharing richly structured information through blogs and wikis. In *ISWC*. 2010.

[3] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 2008.

[4] S. Cohen, C. Li, J. Yang, and C. Yu. Computational journalism: A call to arms to database researchers. In *CIDR*, 2011.

[5] A. Das Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *SIGMOD*, 2012.

[6] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134, June 2005.

[7] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon. Google fusion tables: web-centered data management and collaboration. In *SIGMOD*, 2010.

[8] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. M. McKeon. ManyEyes: a site for visualization at internet scale. *IEEE Trans. Vis. Comput. Graph.*, 2007.

[9] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland. TextRunner: open information extraction on the web. In *ACL*, 2007.