Computational Journalism: A Call to Arms to Database Researchers

Sarah Cohen
School of Public Policy
Duke Univ.
sarah.cohen@duke.edu

Chengkai Li
Dept. of Comp. Sci. & Eng.
Univ. of Texas at Arlington
cli@uta.edu

Jun Yang
Dept. of Comp. Sci.
Duke Univ.
junyang@cs.duke.edu

Cong Yu
Google Inc.
congy@umich.edu

1. INTRODUCTION

The digital age has brought sweeping changes to the news media. While online consumption of news is on the rise, fewer people today read newspapers. Newspaper advertising revenues fell by a total of 23% in 2007 and 2008, and tumbled 26% more in 2009 [1]. This continuing decline of the traditional news media affects not only how news are *disseminated* and *consumed*, but also how much and what types of news are *produced*, which have profound impact on the well-being of our society. In the past, we have come to rely heavily upon the traditional news organizations for their investigative reporting to hold governments, corporations, and powerful individuals accountable to our society. The decline of traditional media has led to dwindling support for this style of journalism, which is considered as cost-intensive and having little revenue-generating potential.

Today, there are fewer reporters gathering original material than in a generation. By some reports, full-time newsroom employment has fallen by one-quarter over the past ten years [2]. Bloggers, citizen journalists, and some non-profit news agencies have made up for only a small part of this loss. The growth in the online news organizations has been mostly in the role of *aggregators*, who read other blogs and news reports, and select, aggregate, edit and comment on their findings. There is a real danger that the proud tradition of original, in-depth investigative reporting will fade away with the ailing traditional news media.

Luckily, a second trend is on our side: the continuing advances in computing. We are connecting people together on unprecedented scales. Data collection, management, and analysis have become ever more efficient, scalable, and sophisticated. The amount of data available to the public in a digital form has surged. Problems of increasing size and complexity are being tackled by computation. Could computing technology—which has played no small part in the decline of the traditional news media—turn out to be a savior of journalism's watchdog tradition?

In the summer of 2009, a group (including two authors of this paper) of journalists, civic hackers, and researchers in social science and computer science gathered for a workshop at Stanford on the nascent field of *computational journalism*, and discussed how computation can help lower cost, increase effectiveness, and encourage participation for investigative journalism. In this paper, we present a more focused perspective from database researchers by outlining a vision for a system to support mass collaboration of investigative journalists and concerned citizens. We discuss several features of the system as a sample of interesting database research challenges. We argue that computational journalism is a rich field worthy of attention from the database community, from both intellectual and social perspectives.

2. A CLOUD FOR THE CROWD

News organizations today have little time and resources for investigative pieces. It is economically infeasible for most news organizations to provide their own support for investigative journalism at a sufficient level. To help, we envision a system based on *a cloud for the crowd*, which combines computational resources as well as human expertise to support more efficient and effective investigative journalism.

The "cloud" part of our vision is easy to understand. Treating computing as a utility, the emerging paradigm of cloud computing enables users to "rent" infrastructure and services as needed and only pay for actual usage. Thus, participating news units can share system setup and maintenance costs, and each reporter can access a much bigger pool of resources than otherwise possible. Popular tools, such as those supporting the Map/Reduce model, have made scalable data processing easy in the cloud. These tools are a perfect fit for many computational journalism tasks that are inherently data-parallel, such as converting audio or scanned documents to text, natural language processing, extracting entities and relationships, etc. Finally, sharing of infrastructure and services encourages sharing of data, results, and computational tools, thereby facilitating collaboration.

Cloud for computational journalism is becoming a reality. A pioneering example is DocumentCloud.org, started by a group of journalists at ProPublica and the *New York Times* in 2008. It hosts original and user-annotated documents as well as tools for processing and publishing them. Conveniently, it uses a Ruby-based Map/Reduce implementation to perform document OCR on Amazon EZ2. Going beyond DocumentCloud's document-centricity, the system we envision would also help manage, integrate, and analyze *structured data*, which may be either extracted from text or published by a growing number of public or government sources. With structured data, we can draw upon a wealth of proven ideas and techniques from databases, ranging from declarative languages, continuous querying, to parallel query processing. Implementing them in the cloud setting raises new challenges, and is a direction actively pursued by the database community. Computational journalism may well be a "killer app" for this line of research.

We would like to emphasize the "crowd" part of our vision more, however. While the "cloud" part copes with the need for computational resources, investigative journalism will never be fully automated by computation. How can we leverage the "crowd" to cope with the need for human expertise? As a start, DocumentCloud allows sharing of human annotations on documents and facilitates collaborative development of tools. With declarative information extraction and declarative querying, we could further share and reuse results (final or intermediate) of data processing tasks initiated by different users. From the perspective of the system, such collaboration occurs opportunistically. Ideally, we want to *actively* direct the efforts of the crowd. For example, in 2009, *The Guardian* of London put up close to half a million pages of expense documents filed by British MPs on the Web, and asked viewers to help identify suspicious items. Within 80 hours, 170,000 documents were examined, making this crowdsourcing effort a spectacular success [3]. As an example from the database community, the *Cimple* project on community information management [4] relies on user feedback to verify and improve accuracy of information extracted from Web pages. Many existing approaches assign jobs to the crowd in simple ways (e.g., pick any document to check). They will certainly get the job done in the long run, but if we have in mind an idea for a story with a tight deadline—which is often the case for journalists—assigning many jobs whose outcomes bear little relevance to our goal will dilute the crowd's efforts.

What we envision instead is a system that intelligently plans (or helps to plan) the crowd's efforts in a *goal-driven* fashion. Given a computational task (e.g., a query), the system would generate a tentative result based on the data it currently has. At this point, the result can be quite uncertain, because the data contain all sorts of uncertainty, ranging from imperfect information extraction to errors in publicly released datasets. Suppose the result looks newsworthy, and a journalist is allocated a limited amount of time and resources to investigate this lead. To leverage the power of the crowd, our system would come up with mini-tasks to be crowdsourced. These mini-tasks can be as simple as checking an entity-relationship extracted from a document, or as complex as reconciling entries from two different public databases. Different lists of mini-tasks would be presented to different users according to their expertise and preference. Mini-tasks whose completion contributes most to reducing the overall result uncertainty will be listed as having a higher priority. As results of mini-tasks become available, the system would reevaluate the overall result, and adjust and reprioritize the remaining mini-tasks accordingly. This idea can be seen as a generalization of the pay-as-you-go approach to data integration in *dataspaces* [5], which considered mini-tasks that establish correspondences between entities from different data sources.

To better illustrate the new challenges and opportunities involved, put yourself in the shoes of a journalist who just noticed a huge number of blog posts about high crime rates around the Los Angeles City Hall. First, are there really that many posts about high crime rates in this area, or did the automated extraction procedure pick up something bogus? Second, does having a large number of blog posts necessarily increase the credibility of the claim, or did most of these posts simply copy from others? Knowing that the number of *original* sources for a story is almost always very low, a seasoned journalist will likely start with a few popular posts, verify that they indeed talk about high crime rates around Los Angeles City Hall, and then trace these posts back to find the original sources. When planning for crowdsourcing, our system should try to mimic the thought process of seasoned journalists. In particular, it would be suboptimal to assign mini-tasks for verifying extraction results from a lot of posts, because the number of posts making a claim is a poor indicator for the accuracy of the claim anyway, and checking just a few may boost our confidence in the extraction procedure enough. It is also suboptimal to ask the crowd to trace the sources of many posts, because a handful of them may lead us to the few original sources.

In this case, it came down to a couple of sources: a Los Angeles Police Department site for tracking crimes near specific addresses, and EveryBlock.com, which publishes large bodies of public-domain data (such as crimes and accidents) by location and neighborhood. Further investigation reveals that EveryBlock.com republished data from LAPD, so our task reduces to that of verifying the claim in the LAPD database (a topic that we shall return to in Section 3). Interestingly, according to the geocoded map locations of crimes in the database, the numbers check out: the crime rate at 90012, ZIP code for the Los Angeles City Hall, indeed ranked consistently as the highest in the city. But a true investigative journalist does not stop here; in fact, there is where the fun begins. It would be nice for our system to help journalists quickly eliminate other possibilities and zoom in on the fun part.

As it turned out, there was a glitch in the geocoding software used by the LAPD site to automatically convert street addresses to map locations. Whenever the conversion failed, the software used the default map location for Los Angeles, right near the City Hall, hence resulting in a disproportionally high crime rate. Arriving at this conclusion does require considerable skill and insight, but our system can help by providing easy access to the full dataset (with street addresses included), and by crowdsourcing the mini-tasks of checking crime records that have been geocoded to the default location (if no alternative geocoding software is available).

Much of what we described in this example took place in real life, and was the subject of a 2009 story in the *Los Angeles Times* [6]. We do not know how many bloggers picked up the false information, but the fact that the *Los Angeles Times* published this piece about the software glitch instead of a column on crimes near the City Hall is both comforting and instructive. As the Internet has made it trivial to publish (and republish) information—especially with the proliferation of social networking—there is a real danger of misinformation going viral. It is important for computational journalism to help preserve journalistic principles and to facilitate fact-checking (more on these in Section 3).

Many research challenges lie ahead of us in supporting intelligent planning of crowdsourcing for investigative journalism. Before we can hope to replicate or improve the cost-benefit analysis implicitly carried out in the mind of a seasoned journalist, we first need frameworks for representing prior knowledge and uncertainty in data (raw and derived) and reasoning with them. There has been a lot of work on probabilistic databases [7], and it would be great to put the techniques to a serious test. For example, how do we represent a large directed acyclic graph of uncertain dependencies among original sources and derived stories? How do we capture the belief that the number of original sources is small? We believe studying our application will necessitate advances in data uncertainty research.

Given a declarative specification of what we seek from data, we also need methods to determine what underlying data matter most to the result. Akin to sensitivity analysis, these methods are crucial in prioritizing mini-tasks. Work on lineage [8] took an important initial step towards this direction, but we are interested in not only *whether* something contributes to the result, but also *how much* it would affect the result when it turns out to be something else. Work on dataspaces [5] laid out an interesting direction based on the concept of the *value of perfect information*, but with general mini-tasks and more complex workflows involving extraction, cleansing, and querying, the problem becomes more challenging.

In addition to the benefit of a mini-task, we will also need to quantify its cost. The idea of exploring the cost-benefit tradeoff has been investigated in the context of *acquisitional query processing* in sensor networks [9]. The human dimension of the crowd creates many new problems. Interests, expertise, and availability vary greatly across users. Some mini-tasks may never be picked up. Sometimes users do a poor job. Therefore, it is difficult to predict a mini-task's cost and result quality (which affects its *actual* benefit). Also challenging are the problems of adjusting crowdsourcing plans dynamically based on feedback, coordinating crowdsourcing among concurrent investigations, and allocating incentives using, say, the Amazon Mechanical Turk. Efforts such as American Public Media's Public Insight Network have taken a qualitative approach toward building and using participant profiles. It would be interesting to see whether a more quantitative approach can be made to work for a crowd.

To recap, we envision a system for computational journalism based on "a cloud for crowd," which hosts tools and data (raw and derived, unstructured and structured), runs computational tasks, and intelligently plans and manages crowdsourcing. It combines resources and efforts to tackle large tasks, and it seeks to leverage and augment human expertise. Within the system, there are endless possibilities for innovative applications of computing to journalism. In the next section, we describe a few specific ideas with clear database research challenges, which help attract participation and maintain a healthy ecosystem that encourages accountability in both subjects and practice of reporting.

3. FROM FINDING ANSWERS TO FINDING QUESTIONS

Much of the database research to date has focused on answering questions. For journalism, however, finding interesting questions to ask is often more important. How do we define interestingness? Where do we come up with interesting questions? To gain some insights, we start with two news excerpts as examples:

The water at American beaches was seriously polluted … with the number of closing and advisory days at ocean, bay and Great Lakes beaches reaching more than 20,000 for the fourth consecutive year, according to the 19th annual beach water quality report released today by the Natural Resources Defense Council (NRDC). … [10]

… Hopkins County also maintained the lowest monthly jobless rate in the immediate eight-county region for the 29th consecutive month. … [11]

Like the two excerpts above, many news stories contain factual statements citing statistics that highlight their newsworthiness or support their claims. Oftentimes, these statements are essentially English descriptions of queries and answers over structured datasets in the public domain.

What if, for each such statement in news stories, we get a pointer to the relevant data source as well as a query (say, in SQL) whose answer over the data source would support the statement?

With such information, we can turn stories *live*. We are used to static stories that are valid at particular points in time. But now, our system can continuously evaluate the query as the data source is updated, and alert us when its answer changes. In the beach water quality example, a continuous query would be able to monitor the alarming condition automatically year after year. If more detailed (e.g., daily) data are available, we can tweak the query to make monitoring more proactive—instead of waiting for an annual report, it would alert us as soon as the number of closing days this year has reached the threshold. Thus, the query in the original story lives on, and serves as a lead for follow-ups.

We will be able to make stories *multiply*. The original story may choose to focus on a particular time, location, entity, or way of looking at data. But given the data source and the query, we can generalize the query as a parameterized template, and try other instantiations of it on the data to see if they lead to other stories. In the jobless rate example, an interested user may instantiate another query to compare her own county of residence against its neighbors. Note that the original query requires more skills than might appear at first glance: it searches for a Pareto-optimal point (x, y) to report, where x is the number of neighboring counties and y is the number of consecutive months. By enabling a story to multiply, we facilitate reuse of investigative efforts devoted to the story, thereby alleviating the lack of expertise, especially at smaller local news organizations.

We will be able to *fact-check* stories quickly. Fact-checking exposes misinformation by politicians, corporations, and special-interest groups, and guards against errors and shady practices in reporting. Consider the following example from FactCheckED.org [12], a project of the Annenberg Public Policy Center. During a Republican presidential candidates' debate in 2007, Rudy Giuliani claimed that adoptions went up 65 to 70 percent in the New York City when he was the mayor. The city's Administration for Children's Services (ACS), established by Giuliani in 1996, made a similar claim by comparing the total number of adoptions during 1996-2001 to that during 1990-1995.

If we were given the data source and the query associated with the claim above, our system can simply run the query and compare its result against the claim. You may be surprised (or perhaps not so) to find that many claims exposed by FactCheckED.org cannot even pass such simple checks. This example, however, requires more effort. According to FactCheckED.org, the underlying adoption data, when broken down by year, actually show that adoption began to slow down in 1998, a trend that continued through 2006. Lumping data together into the periods of 1990-1995 and 1996-2001 masks this trend.

Even when simple automatic fact-checking fails, making sources and queries available goes a long way in helping readers uncover subtle issues such as the one above. When reading stories (or research papers), we often find ourselves wondering why authors have chosen to show data in a particular way, and wishing that we get to ask questions differently. In reality, most of us rarely fact-check because of its high overhead—we need to identify the data sources, learn their schema, and write queries from scratch. By making sources and queries available for investigation, we can significantly increase the crowd's participation in fact-checking to help us ensure accountability.

We have discussed three useful tools—making stories live, making stories multiply, and fact-checking stories—all based on the assumption of having sources and queries to support claims. Naturally, the next question is how to get such information. We could ask makers of claims to provide this information (akin to requiring data and methods for scientific papers), but this approach does not always work. These people may no longer be available, they may have obtained the answers by hand, or they may have motives to withhold that information. Instead, can our system help us identify the data source and reverse-engineer the query associated with a claim?

This problem seems to entail solving the natural language querying problem, which several research and productization efforts attempted in the past but has not caught on in practice. We believe, however, that two new approaches will give us extra leverage. First, we have the text not only for the query, but also for its answer. Evaluating a candidate query on a candidate dataset and comparing the answer can serve as a powerful confirmation. Second, and perhaps more importantly, as more people use our tools on stories, our system can build up, over time, a library containing a wealth of information about data sources, queries and answers, as well as how they are used in actual stories. Suggestions for relevant data sources may come from stories on similar topics. Reverse engineering of queries can benefit from seeing how similar texts have been translated.

This library also leads us to the interesting possibility of building a *reporter's black box*. An investigative piece may involve hundreds of hand-crafted queries on a structured database. Apprentices of investigative journalism face a steep learning curve to write interesting queries. In fact, the majority of these queries seem to conform to some standard patterns—grouping, aggregation, ranking, looking for outliers, checking for missing values, etc. A reporter's black box will be a tool that automatically runs all sensible instantiations of "standard" query templates on a database. For databases that are updated, the black box will automatically monitor them by evaluating the queries in a continuous fashion. The library of datasets and queries maintained by our system will help us discover and maintain collections of interesting query templates. It will also help us find patterns across datasets (or those with particular schema elements), allowing us to "seed" template collections for new datasets.

The number of interesting query templates for a dataset may be large, and the number of instantiations will be even larger, since a parameter can potentially take on any value in the dataset. When the reporter's black box runs on a dataset, it should present query-answer pairs in order of their newsworthiness, which helps journalists focus their efforts. Ranking criteria may incorporate generic ones such as query length (compact queries are more compelling) or template-specific ones such as answer robustness (answers that change with small perturbations to query parameters are less compelling). The library of datasets and queries maintained by our system is also useful. For example, queries with templates that have been used by many high-impact stories probably should be ranked higher, especially if their answers have not appeared in old stories based on the same templates.

Running a large number of queries and monitoring tasks *en masse* poses interesting system and algorithmic challenges. Cloud parallelization helps. Techniques in multi-query optimization and scalable continuous query processing are applicable. However, queries in the reporter's black box can get quite complex (if you have trouble motivating skyline queries, look here), which complicates shared processing. On the other hand, more sharing arises from the fact that many queries are instantiated from the same template. The need to produce ranked query-answer pairs also presents unique challenges and opportunities. Instead of devoting an equal amount of computational resources to each query, we would give priority to queries that are more likely to yield high-ranking query-answer pairs.

Interestingly, there is one area of news where a specialized reporter's black box has been immensely successful—sports. It is amazing how commentaries of the form "player X is the second since year Y to record, as a reserve, at least α points, β rebounds, γ assists, and δ blocks in a game" can be generated seemingly instantaneously. Replicating this success for investigative journalism is difficult. While sports statistics attract tremendous interests and money, public interest journalism remains cash-strapped, and has to deal with a wider range of domains, smarter "adversaries," more diverse and less accurate data sources, and larger data volumes. The ideas presented in this section will hopefully help combat these challenges, by providing efficient, easy-to-use computational tools that aid journalists and citizens in their collaboration to ensure accountability, and by creating a positive feedback cycle where participation helps improve the effectiveness of these tools.

4. CONCLUSION

In this short paper, we have outlined our vision for a system to support collaborative investigative journalism. We have focused on several features of the system to highlight a few important database research challenges. As the Chinese saying goes, we are "throwing a brick to attract a jade"—there are many more interesting problems, both inside and outside the realm of database research: privacy, trust and authority, data mining, information retrieval, speech and vision, visualization, etc.

The need to realize this vision is already apparent. With the movement towards accountability and transparency, the amount of data available to the public is ever increasing. But at the same time, the ability to work with data for public interest journalism remains limited to a small number of reporters. Computation may be the key to bridge this divide and to preserve journalism's watchdog tradition. We hope to motivate you, both intellectually and civically, to join us in working on computational journalism.

Acknowledgments *The authors would like to thank all participants of the 2009 workshop on* Accountability through Algorithm: Developing the Field of Computational Journalism, *organized by James T. Hamilton and Fred Turner, and hosted by the Center for Advanced Study in the Behavioral Sciences at Stanford.*

C.L. and J.Y. would also like to thank HP Labs in Beijing, China for hosting their visits in the summer of 2010, during which some ideas in this paper were discussed.

5. REFERENCES

[1] Pew Research Center's Project for Excellence in Journalism. "The state of the news media." March 15, 2010.

[2] American Society of News Editors. "Decline in newsroom jobs slows." April 11, 2010.

[3] Andersen. "Four crowdsourcing lessons from the Guardian's (spectacular) expenses-scandal experiment." Nieman Journalism Lab. June 23, 2009.

[4] Doan et al. "Community information management." *IEEE Data Engineering Bulletin*, 29(1), 2006.

[5] Jeffrey, Franklin, and Halevy. "Pay-as-you-go user feedback for dataspace systems." *SIGMOD* 2008.

[6] Welsh and Smith. "Highest crime rate in L.A.? No, just an LAPD map glitch." *The Los Angeles Times*. April 5, 2009.

[7] Dalvi, Ré, and Suciu. "Probabilistic databases: diamonds in the dirt." *CACM*, vol. 52, 2009.

[8] Widom. "Trio: a system for data, uncertainty, and lineage." In Aggarwal, editor, *Managing and Mining Uncertain Data*, Springer, 2009.

[9] Madden et al. "TinyDB: an acquisitional query processing system for sensor networks." *TODS*, 30(1), 2005.

[10] Natural Resources Defense Council. "Beach closing days nationwide top 20,000 for fourth consecutive year." July 29, 2009.

[11] Bruce Alsobrook. "Local unemployment rate best in eight-county area for 29th straight month." *The Sulphur Springs News-Telegram*. September 22, 2010.

[12] FactCheckED.org. "Dubious adoption data." June 6, 2007.