Data Capture with the Crowd: Exploring the Continuum of Implicit to Explicit

Jeffrey Nichols, Jalal Mahmud
IBM Research – Almaden
650 Harry Road, San Jose, CA 95110
{jwnichols | jmahmud}@us.ibm.com

ABSTRACT
The crowd can provide a tremendous amount of data to help us understand the world around us. Data from social network sites, such as Twitter, allow us to implicitly mine data from the crowd, for example to understand different opinions on political issues, the important moments in an event, or the sentiment around particular brands. Other systems, such as CreekWatch, ask members of the crowd to explicitly collect data in a particular domain, such as waterway quality. In this position paper, we describe several dimensions along which crowd-sourced data capture solutions can be described and explore solutions on the continuum between implicit and explicit.

INTRODUCTION
At IBM Research - Almaden, we have been working on a variety of crowd-related projects for a number of years. For example, one of our first systems in this space, CoScripter [4], used a wiki-based approach to crowdsource a collection of web automation scripts that helped IBMers carry out internal business processes. A more recent project by others at Almaden, CreekWatch [2], has explored the use of a mobile application to crowdsource useful scientific data from motivated citizens. We have also used crowd services, such as Mechanical Turk, as tools in our research to outsource data collection [3] and user studies [1]. The first author also used Mechanical Turk to attempt to predict the outcome of the 2010 NCAA Basketball Tournament [5]. Most recently, our focus has shifted to mining the status updates generated by social networks, primarily Twitter.

We are interested in all forms of crowdsourcing, which we see as broken down into three types:

- **Data Capture:** Data about individuals, events, locations, etc. is gathered from the crowd. It might be mined from social networks such as Twitter, or manually entered into specially designed collection systems, such as CreekWatch.
- **Analysis:** Data is shown to humans and the humans classify or otherwise extract information from it. Obviously this is most typically done for tasks that are difficult or impossible for machines.
- **Acting:** Human workers are used to create new content or modify existing content. This might be combined with an analysis task simultaneously, but not necessarily.

Most of our previous and continuing work can be seen as data capture, and we focus primarily on that area in this position paper.

The remainder of the paper will discuss some dimensions along which we believe crowd-sourced data capture systems can be designed, and then focus on two point designs in that space on which we are currently working.

SOME INITIAL DIMENSIONS OF DATA CAPTURE
Figure 1 shows three different dimensions along which crowd-based data capture solutions lie. These are described in the sub-sections below.

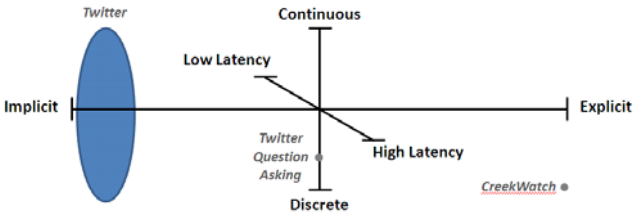

Figure 1. Different Dimensions of Data Capture Systems

Implicit vs. Explicit
The X-axis in Figure 1, this dimension describes how the data generating users are involved with the system. In implicit systems, such as those that collect data from Twitter, the users generate data for their own reasons and may not be aware that their data is being analyzed by the system. In explicit systems, the users often generate data specifically for the system and are usually aware that their data is being analyzed.

There are trade-offs for choosing one extreme of this dimension versus the other. With implicit systems, it is easy to collect high volumes of data from many users, but

CHI 2011 Workshop on Crowdsourcing and Human Computation, May 8, 2011, Vancouver, BC, Canada.

because this data is not generated with the data capture process in mind, it can be very noisy and hard to analyze. On the other extreme, users generate data specifically for the system. Thus the generated data is likely of higher quality and structured in such a way that it is easier to analyze. Generating high volumes of data from different users is more difficult however, because users must be willing to enter their data into the specialized data collection tool.

This dimension is particularly interesting to us because there seem to be few point designs in between the two extremes and we believe this may be a fruitful area for designing new data capture systems.

Continuous vs. Discrete

This dimension is shown as the Y-axis in Figure 1, and it refers to the sampling rate at which data is collected. In some cases, this may be restricted by the data source, but in most cases it can be controlled by the system designer. For example, systems based on collecting data from Twitter can exist at either extreme of this dimension depending on their design; A system that collects data from Twitter's streaming API is more continuous than one that polls Twitter hourly. Many explicit systems are very discrete, such as CreekWatch, because users must manually enter each data point, but it is possible to create a more continuous system by integrating sensors that report more frequently. Google Latitude is arguably one example of an explicit continuous system, where users explicitly decide to share their GPS data and that data may be sampled frequently.

Unlike with the implicit/explicit dimension, it does not seem that choosing a different point along this axis has any affect on quality of the data that is captured.

Latency

Latency is represented as the Z-axis in Figure 1, and it refers to the rate at which useful conclusions can be made based on the collected data. The latency of any given system depends on how it is used, and systems may exist a multiple points on the latency dimension for different use cases. For example, Google Latitude has very low latency if you want to know where a given online user is right now, but potentially a very high latency if you want to know where a user is likely to be at a given time of day.

Latency can be heavily influenced by whether a system is always capturing data or does so "on-demand" at the request of a user. Most of the examples given so far are constantly collecting data, but there are others, such as social Q&A systems, that collect data only after the user has requested it. A key question for these latter systems is how to reduce latency.

TWO CROWDSOURCING PROJECTS

This section discussed two projects that we are currently conducting in the crowdsourcing data capture space.

Summarizing Events From Implicit Capture

Our goal in this project is to generate journalistic summaries of events based entirely on data collected from Twitter. This work is somewhat similar to Shamma et al.'s [6] work on the Presidential debates and other events, but while their focus was primarily on visualizing twitter data, our goal is to produce a meaningful text/video summary. Our first attempt in this space is to produce SportsCenter-style highlight reels using video of recorded games with voiceover generated from Twitter data.

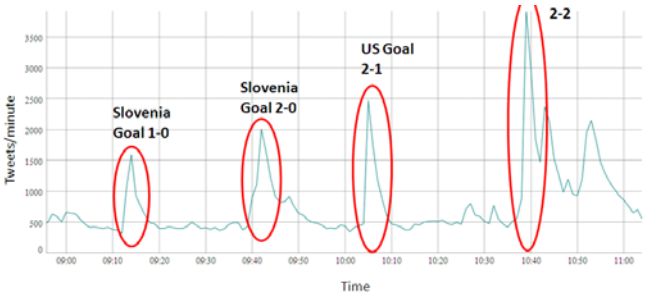

Figure 2. A graph of Twitter volume in tweets/minute over the course of the World Cup game between the US and Slovenia. Goals are highlighted.

We believe this is possible because of the properties of the tweets that are made during an event. Figure 2 shows a graph of the tweet volume recorded for the US vs. Slovenia World Cup game in July 2010. This data was recorded during the game via the streaming API using the keywords "#worldcup" and "#wc2010". This approach means that our data set is incomplete and noisy; we probably did not capture all tweets about this game, and some of the tweets that were captured are about other World Cup topics and not the US vs. Slovenia game. The first thing to notice about the tweet volume is that there are noticeable spikes in the data. These spikes correspond to important moments in the event, including the start, half-time, and end of the game, each goal, penalties which warranted a yellow card, a disallowed goal, and at least two other questionable calls by the ref.

Our approach to generating a summary is as follows:

1. We identify spikes in the tweet volume using a slope-based threshold metric. For this particular game, 9 spikes are identified.
2. For the minute surrounding each spike, we extract the longest sentence from each tweet, tokenize the sentences, and generate a phrase graph containing nodes for each word and edges between words that are adjacent. The graph also contains weights on the nodes and edge based on each occurrence of the word or adjacency respectively. This is a based on an existing tweet summarization algorithm [7].
3. We then score each sentence in our set using the phrase graph, and pick the top n non-overlapping sentences above a score threshold as a summary of the spike.

Figure 3 shows an example of a moment in the game and two summary sentences.

This work is in progress and more remains to be done. We are currently exploring using additional heurstics, such as sentiment and the type of sentence segmentation used in the tweet to further refine the sentences that are chosen. We also have yet to explore how to extract snippets of video based on the spikes that show the event in question.

Our current approach also does not use any domain information, either about sports in general or the particular sport being played. We intend to examine using domain information as an alternate approach.

Figure 3. Slovenia's first goal based on tweets might be summarized as: "Valter Birsa scores, putting Slovenia in the lead 1-0 against the United States." "Tim Howard was stunned doing nothing."

A Hybrid Implicit/Explicit Data Capture Approach

Of particular interest to us is exploring a middle ground between implicit and explicit data capture techniques. To our knowledge, Tweak the Tweet [8] is one of the few systems that exists in this space. In the case of that system, data collectors specially format their tweets using a particular combination of hashtags so that the system can understand their information without additional analysis. This system seems to be primarily explicit in that users must carefully format their tweets, however using Twitter as a transport layer allows other implicit systems to track the information that users input even if other systems don't understand the particular format.

We are working to build a different data collection system that utilizes aspects of both implicit and explicit. The idea is to rely on implicit mechanisms for identifying people who may have good data to share, and then explicitly asking the user to provide that information. For this to work, people will need to be motivated to respond to the question, and we plan to examine different ways of motivating users.

The first prototype of this system that we are building now is a TSA Security Checkpoint Tracking application. The goal will be to assemble a large dataset of how long it takes to traverse security checkpoints at different times of day in different airports. The system will work as follows:

1. Using the twitter streaming API, we will monitor Twitter for tweets about being in airports or having just gone through security. We plan to initially use a human filter to identify relevant tweets, but we may offload this task to Mechanical Turk. A particular focus will be conservatively identifying relevant tweets, so that the questions sent in the next step will not be considered spam.
2. For tweets that are deemed relevant, we will send an @user reply to the sender of the tweet asking if they will share how long it took to get through security and which airport they are at. This reply will also include a link that explains the goal of the project and hopefully helps motivate users to respond with data.
3. We will collect any responses and add them to a public visualization available on the web.

There are several questions that we hope to answer through this prototype. First, how difficult is it to identify relevant tweets, and can it done by a machine, the crowd, or only an individual? Second, how can we motivate users to respond positively to our questions? It seems that communicating the value of a response will be key, but may be difficult in the limited 140 characters that are available.

Assuming that we can make this prototype successful, we are interested to apply it other domains, such as customer service, where an interested company might scan twitter for complaints about its service and request additional details automatically from people who complain. This might also be a reasonable approach to help potential users of existing explicit tools, such as CreekWatch, discover that those tools exist.

We are also interested exploring how this technique compares to ESM, another similar though time-intensive method of learning about user experiences. Understanding the similarities and differences between these two approaches will be interesting.

SHORT BIO

Jeffrey Nichols is a Research Staff Member at IBM Research – Almaden. His research interests are in the field of human-computer interaction, with a specific focus on crowdsourcing, automated design, mobile computing, and end-user programming. He received his Ph.D. in December 2006 from the Human-Computer Interaction Institute in Carnegie Mellon University's School of Computer Science. His thesis described the first system to automatically generate interfaces that are consistent with a user's previous experience and provided the first evidence from user studies that automatically generated interfaces can be more usable

than human-designed interfaces in certain situations. He received a BS degree in computer engineering from the University of Washington in 2000.

REFERENCES

1. Bigham, J.P., Lau, T. and Nichols, J., TrailBlazer: Enabling Blind Users to Blaze Trails Through the Web. in *Intelligent User Interfaces*, (Sanibel Island, Florida, USA, 2009), 177-186.
2. Kim, S., Robson, C., Zimmerman, T., Pierce, J. and Haber, E.M., Creek Watch: Pairing Usefulness and Usability for Successful Citizen Science. in *Proceedings of CHI'2011*, (Vancouver, British Columbia, Canada, 2011).
3. Lau, T., Drews, C. and Nichols, J., Interpreting Written How-To Instructions. in *Proceedings of IJCAI 2009*, (Pasadena, CA, 2009).
4. Little, G., Lau, T., Cypher, A., Lin, J., Haber, E.M. and Kandogan, E., Koala: Capture, Share, Automate, Personalize Business Processes on the Web. in *CHI*, (San Jose, CA, 2007), 943-946.
5. Nichols, J. Mechanical Turk and the NCAA Tournament, 2010.
6. Shamma, D.A., Kennedy, L. and Churchill, E.F., Peaks and Persistence: Modeling the Shape of Microblog Conversations. in *Proceedings of CSCW 2011*, (Hangzhou, China, 2011).
7. Sharifi, B., Hutton, M.-A. and Kalita, J., Summarizing Microblogs Automatically. in *Proceedings of Human Language Technologies*, (Los Angeles, CA, 2010), 685-688.
8. Starbird, K. and Stamberger, J., Tweak the Tweet: Leveraging Microblogging Proliferation with a Prescriptive Grammar to Support Citizen Reporting. in *Information Systems for Crisis Response and Management Conference*, (Seattle, WA, 2010).