CrowdMGR: Interactive Visual Analytics to Interpret Crowdsourced Data

Abon Chaudhuri*
Intel, Hillsboro, USA
abon.chaudhuri@intel.com

Mahashweta Das†
HP Labs, Palo Alto, USA
mahashweta.das@hp.com

ABSTRACT

Crowdsourcing is popularly defined as a paradigm that utilizes human processing power to solve problems that computers cannot yet solve. While recent research has been dedicated to improve the problem-solving potential of crowdsourcing activities, not much has been done to help a user quickly extract the valuable knowledge from crowdsourced solutions to a problem, without having to spend a lot of time examining all content in details. Online knowledge-sharing forums (Y! Answers, Quora, and StackOverflow), review aggregation platforms (Amazon, Yelp, and IMDB), etc. are all instances of crowdsourcing sites which users visit to find out solutions to problems. In this paper, we build a system CrowdMGR that performs visual analytics to help users manage and interpret crowdsourced data, and find relevant nuggets of information. Given a user query (i.e., a problem), CrowdMGR returns the solution, referred to as the SolutionGraph, to the problem as an interactive canvas of linked visualizations. The SolutionGraph allows a user to systematically explore, visualize and extract the knowledge in the crowdsourced data. It not only summarizes content directly linked to a user's query, but also enables her to explore related topics within the temporal and topical scope of the query and discover answers to questions which she did not even ask. In the demonstration, participants are invited to manage and interpret crowdsourced data in StackOverflow and Computer Science Stack Exchange, question and answer site for students, researchers and practitioners of computer science.

1. INTRODUCTION

Crowdsourcing is the practice of soliciting services, ideas, solutions, or content from an undefined, generally large group of people in the form of an open call. It is also popularly defined as *a paradigm that utilizes human processing power to solve problems that computers cannot yet solve* [7]. Crowdsourcing has received a lot of attention lately from researchers for its potential in solving problems, often unsolvable by computers, by tapping in to the collective intelligence of the crowd. Efforts have been dedicated to designing the optimal task and workflow, recruiting people by studying behavioral and cognitive biases, incentivizing the crowd, processing crowdsourced data to sift value, etc. However, not much has been done to help a user quickly extract the valuable knowledge from crowdsourced solutions to a problem, without having to spend a lot of time examining them in details. Online knowledge-sharing forums (Y! Answers, Quora, and StackOverflow), review aggregation platforms (Amazon, Yelp, and IMDB), etc. are all instances of crowdsourcing sites which users visit to find out solutions to problems. For example, a user may visit Stack Overflow[1] to find the answer to the problem *Is Java "pass-by-reference" or "pass-by-value"?*. Stack Overflow has 47 solutions to the problem and it would not be possible for the user to find the pertinent answer without examining the detailed textual information, often conflicting, at her disposal. Similarly, a user may visit Yelp[2] to find the answer to the problem *Is "B Patisserie" a healthy bakery to visit in the San Francisco neighborhood?*. Yelp has over 500 solutions to the problem (i.e., reviews for the bakery) that the user needs to go through in order to make her decision. Note that, the crowdsourced data in Stack Overflow concerns facts and information while that in Yelp is more about opinion and judgment. However, the task of eliciting the "solution" for the "problem" from the crowdsourced data remains the same across both the applications.

In this paper, we develop a framework that addresses this need. Our system, called CrowdMGR[3] performs analytics to help users *manage* and *interpret* crowdsourced data, and find *relevant* nuggets of information. Given a user query (i.e., a problem), CrowdMGR returns the solution, referred to as the SolutionGraph[4], to the problem as an interactive canvas of linked visualizations. The purpose of SolutionGraph is to enable a user to quickly access the knowledge in the crowdsourced data, in addition to the information that current crowdsourcing sites showcase. Over the past decade, researchers have developed techniques to summarize user-generated content in review sites, internet forums, blogs, etc. [3][5][6]. SolutionGraph not only summarizes content directly linked to a user's query, it also enables her to explore related topics within the *topical* and *temporal* scope of the query and discover answers to questions which she did not even ask. We present our SolutionGraph as an intuitively intelligible visual form, that incorporates graph drawing methods and geometric techniques for high dimensional data visualization, in order to communicate complex analytical information effectively. Its interactive exploration feature allows a user to seamlessly navigate from the high-level overview to the desired levels of granularity and back.

*Authors are listed alphabetically
†Majoriy of work done while at University of Texas, Arlington

[1]http://stackoverflow.com/
[2]http://www.yelp.com/
[3]Abbreviated from Crowd Manager
[4]Name draws inspiration from Knowledge Graph

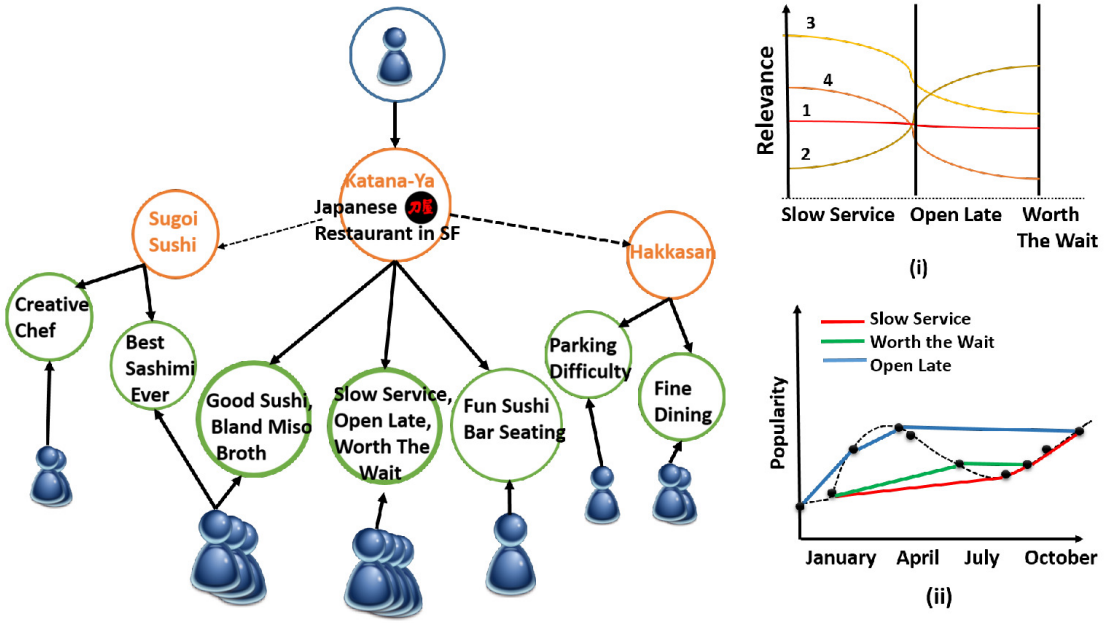

Figure 1: Example SolutionGraph for a query about a Japanese Restaurant in San Francisco; (i) Topical Analysis and (ii) Temporal Analysis of the Answer Node {Slow Service, Open Late, Worth the Wait}

Let us explain our system CrowdMGR and SolutionGraph with a simple illustrative example, presented in Figure 1. Suppose, a user wants to find out *if she will like to visit a particular Japanese restaurant "Katana-Ya" in San Francisco.* Given her query, the SolutionGraph in Figure 1 not only returns the reviews for the restaurant, but also returns two other related restaurants - one Japanese, "Sugoi Sasha" and one Asian Fusion, "Hakkasan" - in the neighborhood and the reviews for them. Since the number of reviews for a restaurant is huge, we aggregate the reviews based on content similarity. If the user is interested in eating Sashimi at a Japanese restaurant in San Francisco at that time, the graph helps her discover a restaurant that meets her preferences better than the one she is querying. Note that, there exists a user who has reviewed both the query restaurant under consideration and the related Japanese restaurant, the former for Sushi and the latter for Sashimi. If the user is interested in eating only at the particular restaurant she is querying, the graph helps her quickly access the broad summary of the feedback it has received. SolutionGraph also allows the user to interact with the system and obtain a detailed insight of the temporal and topical trends of each aggregate answer, as shown in Figure 1-(i) and Figure 1-(ii). Topical analysis of the answer node {Slow Service, Open Late, Worth the Wait} in Figure 1-(i) helps the user access the relevance of the keywords in each of the reviews, that are aggregated. If the user is interested in reading a review about the keyword 'Slow Service', she may read Review 3 in details. Moreover, since the plot suggests that Review 3 and Review 4 are similar in their content, she can readily filter out Review 4. Figure 1-(i) reveals the temporal trend of the answer node {Slow Service, Open Late, Worth the Wait}. The user may note that the keyword 'Open Late' has been frequently mentioned in the reviews in the summer months, while the frequency of the keyword 'Slow Service' has steadily increased over time.

The two main technical challenges in achieving the objective of our system is: (i) how to select the solution nodes for the user query, i.e., problem node in the SolutionGraph; and (ii) how to discover the problem nodes related to the user query in the SolutionGraph. Both the goals are wedded to the definition of *relevance* measure that decides what CrowdMGR intends to show to a user. In this study, our goal is to not advocate one particular measure over another. Rather, we focus on defining the problem framework and demonstrate the utility of our system for managing and interpreting crowdsourced data. Online sites today usually sort user reviews, answers, etc. by decreasing order of popularity (i.e., how many people found the review useful), recency in activity, etc. Ghose et.al [2] has designed review ranking strategies that orders reviews based on their expected helpfulness and expected effect on sale. In this work, we transform each solution to a multi-dimensional weighted feature vector of keywords and employ K Means Clustering that associates similar feature vectors and dissociates dissimilar vectors, where the extent of association (or, dissociation) is measured by the Euclidean distance between the vectors. Several online sites employ natural language processing techniques and machine learning approaches to identify and return list of content related to user query. In this work, we represent each problem as a boolean vector of keywords and employ Jaccard similarity coefficient to identify the related problems. We use force-directed graph drawing algorithm to visualize the graph. Interactive visual analysis of the SolutionGraph to cater to a user's cognitive needs and aid further explorations possess additional challenges. We use parallel-coordinate plots to visually capture and present the topical diversity among similar answers and a 3D point-based plot enhanced by novel visual cues to visually highlight the temporal trend of topics in the SolutionGraph. Note that, CrowdMGR is a real-time system and hence the task of building the SolutionGraph for a user query incurs computational challenges too.

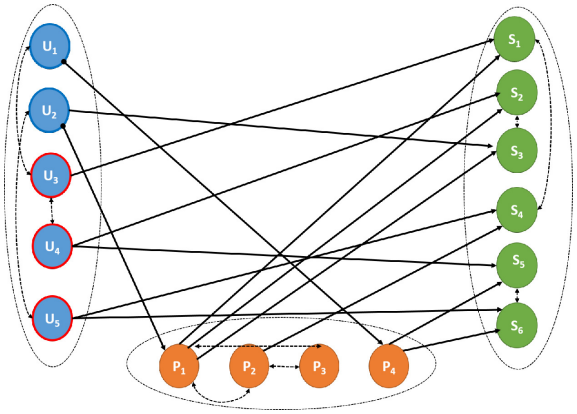

Figure 2: CROWDMGR **Data Model**

2. CrowdMGR DESIGN

The focus of our work is to provide a framework for organizing crowdsourced data in order to help a user access relevant content effectively and efficiently. We first introduce our data model, then discuss our mining problem and algorithmic solution, and finally present our analytics and interactivity features.

2.1 Data Model

A crowdsourcing site, as shown in Figure 2, contains heterogeneous information and can be modeled as a directed tri-partite graph $\mathtt{G}$:

- Nodes: Users ($\mathtt{U}$), Problems ($\mathtt{P}$), and Solutions ($\mathtt{S}$) coexist in the graph. Note that, there are two kinds of users: Problem giver ($\mathtt{U_P}$) and Solution giver ($\mathtt{U_S}$).
- Inter-relational edges: Edges between user nodes, problem nodes and solution nodes can be derived from the explicit interactions in the crowdsourced data. For a user $u \in \mathtt{U}$, there exists an edge from u to a problem $p \in \mathtt{P}$ or to a solution $s \in \mathtt{S}$, depending on $u \in \mathtt{U_P}$ or $u \in \mathtt{U_S}$. There also exists an edge from a node $p \in \mathtt{P}$ to a node $s \in \mathtt{S}$ if s is a solution for the problem p.
- Intra-relational edges: The set of nodes in the partite $\mathtt{P}$ share edges based on content similarities. The set of nodes in the partite $\mathtt{U}$ share edges based on social network ties, demographic profile information overlap, etc. The set of nodes in the partite $\mathtt{S}$ share edges based on semantic relatedness.
- Node weight: User nodes are weighted by their qualification score, problem and solution nodes are weighted by the aggregated count of votes (up, down) they have received. Instead of scalars, the weights can be vectors, e.g., weighted vector of relevance score of keywords in the solution nodes.

For example in Figure 2, $\mathtt{U} = \{u_1, u_2, u_3, u_4, u_5\}$ where $u_1, u_2 \in \mathtt{U_P}$ and $u_3, u_4, u_5 \in \mathtt{U_S}$; $\mathtt{P} = \{p_1, p_2, p_3, p_4\}$; $\mathtt{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$. s_1, s_2, s_3 are solutions to user u_1's problem p_1 provided by users u_2, u_3, u_4 respectively.

2.2 Problem Overview

Given crowdsourced data as a tripartite graph $\mathtt{G}$, a user u_i ($u_i \in \mathtt{U_P}, \mathtt{U_P} \subseteq \mathtt{U}$) and her query p_j ($p_j \in \mathtt{P}$), CROWDMGR identifies the subgraph $\mathtt{G}'$ from $\mathtt{G}$ that contains:

- Set $\mathtt{P}'$ of k_p nodes ($\mathtt{P}' \subseteq \mathtt{P}$) related to p_j by measure $\mathcal{A}$
- Set $\mathtt{S}'$ of k_s nodes ($\mathtt{S}' \subseteq \mathtt{S}$) having directed edges from $\{p_j \cup \mathtt{P}'\}$ and aggregated by measure $\mathcal{B}$
- Set $\mathtt{U}'$ of nodes ($\mathtt{U}' \subseteq \mathtt{U}, u_i \in \mathtt{U_P}, \mathtt{U}' - u_i \subseteq \mathtt{U_S}$) having directed edges from $\{p_j \cup \mathtt{P}'\}$

The subgraph $\mathtt{G}'$ is the SOLUTIONGRAPH for the query. It can also be classified as a semantic graph [1] consisting of heterogeneous nodes and links that carry semantic information in them.

Measure $\mathcal{A}$: The objective of measure $\mathcal{A}$ is to select the top-k_p of neighboring problems in $\mathtt{P}$ that are related to user query p_j. In our system, we consider the popular Jaccard similarity coefficient. We use the keyword extraction toolkit Alchemy API[5] to extract keywords from the problems in $\mathtt{P}$. Thus, each problem is represented as a boolean vector of size n_p, where n_p is the the total number of distinct keywords extracted from $\mathtt{P}$. The Jaccard measure help us determine the top-k_p related problem nodes. Thus, we only leverage the intra-relational edge information between the problem nodes. We may employ the intra-relational edge information in all three partites for this purpose, e.g., people who viewed this restaurant also viewed feature in Yelp.

Measure $\mathcal{B}$: The objective of measure $\mathcal{B}$ is to determine the k_s solution nodes in $\mathtt{S}$ that are to presented in $\mathtt{G}'$ as solution nodes to user query p_j. If the number of solutions for a problem is not large ($\leq n_s$, $\mathtt{G}'$ may just comprise of the solution nodes. However, the number of answers per question is usually large, and answers often receive multiple comments, e.g., in Stack Overflow. Hence, we aggregate the set of all solutions for a problem to determine k_s nodes. In this work, we consider Euclidean distance and employ K Means clustering to group similar vectors together.

2.3 Visualization

Our system presents the analytics report of the crowdsourced data in a visually engaging way:

SolutionGraph: The graph is presented as a node-link style visualization. It is generated using Kamada-Kawai layout algorithm [4]. As discussed in Section 2, there are three types of entities in the graph: user nodes, problem nodes, and solution nodes. Since this is a semantic graph, we use different colors and shapes for representing the different entities and links. Some design choices regarding encoding information in the graph have been made very carefully to facilitate analytics (to be explained in Section 2.4). The visualization of SolutionGraph delivers information on demand to avoid clutter. Before generating the graph, the user can control its size by tuning two parameters: maximum number of related problems and maximum number of solutions for each problem. However, it is also possible to generate a content-rich graph and then control the amount of information to show by applying filters on graph based on node type, degree, popularity and so on. For example, the user may want to see only the highly voted answers for each problem for further analysis. The graph view is associated with two other linked visualizations.

Topical Analysis: The solution nodes are the key entities of interest in the graph. They contain weighted vector of relevance score of keywords extracted from the solutions and

[5]http://http://www.alchemyapi.com/api/keyword-extraction/

broadly summarizes the content in the nodes. Recall that each solution node in the SolutionGraph is an aggregated group of similar answers in the crowdsourced data obtained by K Means clustering. The individual answers belonging to a solution node, though similar, are not identical. To help the user access answer(s) based on keywords, we employ parallel coordinates (PC) plot - a technique known for visualizing high dimensional data - as shown in as shown in Figure 1-(i). Each vertical axis in the PC plot denotes a keyword. The relevance is denoted as a point on the axis. Hence, a feature vector of keywords is represented by a line connecting the points on each axis. PC plot can effectively highlight how similar two answers are even when they belong to the same solution node. To accommodate large number of axes (keywords), we use zoomable PC plot which focuses on a few keywords at a time.

Temporal Analysis: In crowdsourcing sites, the answers to a question are usually posted and voted over a long period of time. A topically relevant answer may actually have lost relevance over time. For example, if the restaurant in Figure 1 was being praised for 'Good Sushi' 3 years back but could not maintain its reputation, a temporal analysis of the keyword 'Good Sushi' should reflect that. To capture these temporal characteristics, we present all the individual answers aggregated in a solution node on a 2D scatterplot enhanced with visual cues, as shown in Figure 1-(ii). The answers are temporally ordered along the horizontal axis, the y-axis captures the popularity of each answer (number of upvotes - number of downvotes). A selectable list of keywords (a subset containing the frequent ones) is presented alongside the plot. As the user selects a keyword, a spline curve connects the answers that contains that keyword. This overlaid curve on top of the scatterplot clearly highlights many facts, e.g., if that keyword has appeared consistently over time, if there is a correlation between the popularity of an answer and existence of that keyword, etc.

2.4 Interaction and Analytics

CrowdMGR allows a user to perform analytics by easy and effective interaction with the system in order to help her seamlessly navigate from the high-level overview to the desired levels of granularity. Our system favors analytics in two ways:

By driving user interaction: Each of the three visualizations in Section 2.3 above has information encoded in such a way that it can channel the user's attention to the meaningful components of the SolutionGraph. For example, the sizes of the user nodes in the graph are determined by the user's reputation (measured by how much the community trusts the user, how actively the user participates, etc.) in the crowdsourcing site. Hence, while looking at the creator of a post (problem node or a solution node), the demo participant can get some idea about the creator's credibility which may help her choose or skip a node. Again, the size of a solution node is proportional to the number of individual answers it is aggregating, thereby conveying the solution highlights and content distribution to the user effortlessly.

By responding to user interaction: As the user views the visual analytics result returned by our system, she is presented with opportunities to drive the analytic process forward. For example, as the user clicks on a solution node on the SolutionGraph, the topical and temporal analysis plots are populated for further analysis. Again, clicking on a curve in the topical analysis plot brings out the actual text of the answer with keyword highlighted. Thus, our interaction framework enables a user to navigate through various levels of detail, otherwise unmanageable. At any point of time, the user can restart the analysis, or step back without having to click through a series of browser back buttons, or having to scroll a long way up.

3. DEMONSTRATION

The CrowdMGR system can work on any crowdsourcing site that provides data as descried in Section 2.1. For the purpose of the demo, we use publicly available Stack Overflow and Stack Exchange data[6]. As of August 2012, the Stack Exchange dump for Computer Science has 10,529 registered users; 4,926 questions of which 2,487 have accepted answers; 7,122 answers; 25,042 comments; and 60,035 votes. The Stack Exchange dump for Programmers has 96,744 registered users; 29,025 questions of which 17,451 have accepted answers; 116,491 answers; 282,421 comments; and 1,391,975 votes. The Stack Overflow dump is even bigger having over 1.3 million registered users and over 4 million questions.

3.1 Demo

Our demo allows the audience to use a standalone application as shown in Figure 3 and specify search query in the scope of the crowdsourcing site under consideration. Example queries include: *Why is quicksort better than other sorting algorithms in practice?*, *What are the text editors for large files?*, and so on. If the query entered by the user is not present in the crowdsourcing site, we identify the question that is most similar to the user query and proceed with it. The audience can specify other query settings such as: maximum number of solution nodes(i.e., k_s) and maximum number of related problems (i.e., k_p) they want to see in the SolutionGraph. They can select any one of the solution nodes and observe the topical and temporal trends of the keywords in it. They can drill down deeper to view the actual textual solution too. Such exploration will give the audience a deeper appreciation of our systemŠs utility to aid users extract the valuable knowledge from crowdsourced data quickly, and it's superiority over content displayed in existing crowdsourcing sites.

3.2 Use Case Study

Let us illustrate our demo with a detailed use case study. Suppose, a user selects the crowdsourcing site `http://cs.stackexchange.com/` and submits the query *Why is quicksort better than other sorting algorithms in practice?*. The maximum number of solution nodes and the maximum number of related problem nodes she submits as input are 2 and 3 respectively. On clicking Find Answer button, The SolutionGraph is generated in the Visualization panel. The problem nodes are in orange (with the user query node having a red border); the solution nodes are in green; and the user nodes are in blue. The size of the user node is proportional to the user's reputation score in the site. The size of the solution node is proportional to the number of individual answers aggregated in it. Note that in the Solution Graph, there exists a user (the node having a red border) who has submitted answer to the user query as well as to

[6]http://http://stackexchange.com/

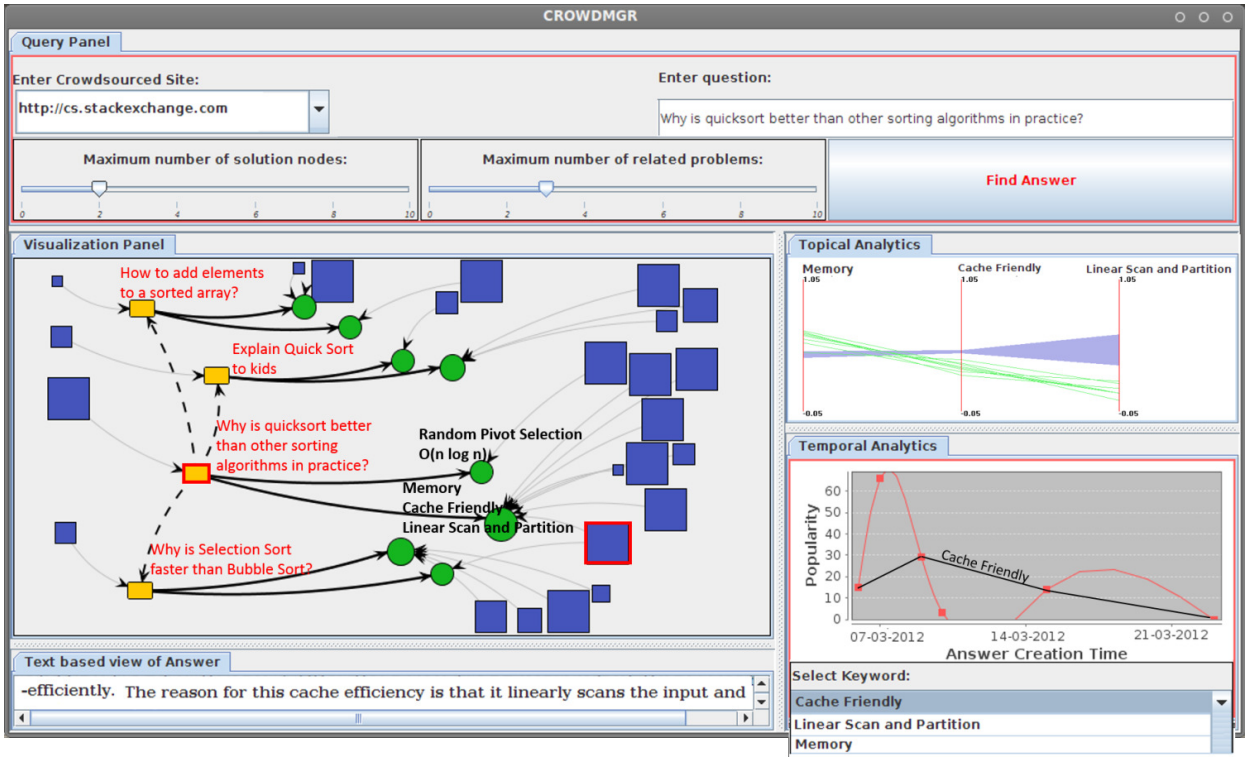

Figure 3: User Interface of CrowdMGR

a related problem, *Why is Selection Sort faster than Bubble Sort?*. Suppose, the user wants to explore the most popular solution node for the query that have 8 answers aggregated, i.e, the solution node mentioning 'Memory', 'Linear Scan and Partition', and 'Cache Friendly'. On clicking that node, the Topical Analytics and Temporal Analytics plots are displayed. On selecting one of the green curves in the Topical Analytics plot, the text of the actual answer is shown in the Text Based View of the Answer panel (bottom left). Also, the user can select a keyword from the drop down option in Temporal Analytics panel to observe the popularity of a keyword, overlaid on top of the popularity of all solutions over time.

4. CONCLUSION

Given a user query, i.e., a problem, CrowdMGR generates a SolutionGraph that helps user manage and interpret crowdsourced data and extract valuable nuggets, i.e., solutions, from it. It enables a user to conduct temporal and topical analysis of the solutions returned for the problem by the system, as well as discover answers to questions which she did not even an ask. Our demo allows users to generate and interactively explore interesting SolutionGraph-s for questions in Stack Overflow and Computer Science Stack Exchange.

Our work is a preliminary look at a very novel problem of research in crowdsourcing and there appear to be many exciting directions of future research. Our immediate goal is to improve the efficiency and effectiveness of our system by employing sophisticated techniques in order to conduct big data analytics and identify nodes to be displayed in the SolutionGraph. Since user-generated content is always on the rise, we plan to handle updates and insertions of new users, problems, and answers in our system. We also intend to investigate the applicability of our framework to other forms of crowdsourced data involving images and videos, as well as other novel applications, e.g., how SolutionGraph can improve the quality of recommendation, etc.

5. REFERENCES

[1] T. Coffman, S. Greenblatt, and S. Marcus. Graph-based technologies for intelligence analysis. *Communications ACM*, 47(3):45–47, 2004.

[2] A. Ghose and P. G. Ipeirotis. Designing novel review ranking systems: predicting the usefulness and impact of reviews. In *ICEC*, pages 303–310, 2007.

[3] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, pages 168–177, 2004.

[4] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.

[5] L.-W. Ku, Y.-T. Liang, and H.-H. Chen. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 100–107, 2006.

[6] Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu. Understanding and summarizing answers in community-based question answering services. In *COLING*, pages 497–504, 2008.

[7] L. Von Ahn. *Human Computation*. PhD thesis, 2005.