Bulletin of the Technical Committee on

Data Engineering

December 2016 Vol. 39 No. 4

IEEE Computer Society

Letters

Special Issue on Human in the Loop in Data Management

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

The Current Issue

Human participation in data management has gone from being esoteric to florishing research area and seems headed for commercial success. One aspect of this is crowdsourcing. The explosion of interest in crowdsourcing is confirmation that the data management community recognizes two important things in our current technical environment: (1) there are questions that can sometimes be easily answered by the right people that "machines" find difficult; and (2) it is possible to bring human input into data management to exploit this human capability. The technology in back of (2) not only brings "humans into the loop" but further, makes their participation a part of the process that can be managed and even optimized. Data analytics also benefits from human (analyst) involvement with machine learning in making sense of data. This work recognizes that, as with crowd sourcing, humans working with machines can frequently produce better results than either working alone.

The current issue reflects the excitement for the field and the progress of the technology. Tova Milo served as editor for this issue and she has succeeded in bringing together many of the diverse technical threads that are undergoing such rapid evolution. I want to thank Tova for bringing this to Bulletin readers. This "snapshot in time" for the field provides a valuable resource for both young researchers thinking about pursuing this as well as for experienced researchers trying to stay abreast of the field.

TCDE Chair Election

Every two years, the Technical Committee on Data Engineering holds an election for TC chair. That election is going on now. There are two candidates running, Erich Neuhold, a past TCDE chair, and Xiaofang Zhou, current TC chair. The next paragraph is information from Brookes Little of the Computer Society explaining the mechanics of voting in this election. I would urge you to participate.

The TCDE Chair election is now open. The Chair's term will run from January1, 2017 through December 31, 2018. The poll will close December 22, 2016 at 5:00pmPacific StandardTime. Before entering the poll, please take a few moments to review candidate Biosketches and Position Statements, which can be found here: `http://www.computer.org/web/tandc/tcde`

*Please note: Only CS Members who are also TCDE Members can vote. **To cast your vote, you will need your IEEE CS Member number. To obtain a misplaced number, please visit:* `https://supportcenter.ieee.org/app/answers/detail/a_id/353/session/L3RpbWUvMTM3NTc0NDYxMi9zaWQvdXIxcVAxeGw%3D`.

To ensure your TCDE membership is valid and up to date, please log in to your IEEE Web Account and check your membership status:`https://www.ieee.org/profile/public/login/publiclogin.html`

You may vote only one time. VOTE here:`https://www.surveymonkey.com/r/87KPGPJ` *If you have trouble voting, please contact T&C Program Manager, Brookes Little (eblittle@computer.org).*

Computer Society News

The proposed amendment to the IEEE constitution changing the way that the IEEE is governed was defeated. The Computer Society had joined with nine other societies in formally opposing the amendment.

The Computer Society also had an election for governing officers. The election results are at `https://www.computer.org/web/election/`. Hironori Kashahara was elected as "President-Elect", an office he will hold for one year before becoming Computer Society President.

I want to thank TCDE members who voted for me as First Vice President. I am happy to report that I was elected. The vote was very close, so I am particularly grateful to those who selected me for this office.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

Over the past years there has been a growing recognition of the increasing role of people in the data life cycle. People massively contribute data and share opinions, people also extensively analyze data and consume the derived information. In this issue, we have a slate of very interesting articles discussing the different roles of human in data management. In particular, we discuss three complementary aspects. First, we consider the role of people as crowdsourcing workers, assisting in data generation and in data-centric computation and analysis. The focus here is on devising methods for cost-effective, meaningful usage of human capabilities and knowledge. Second, we consider the role of people as data analysts. The focus here is on interactive data analysis and mining and the development of tools that facilitate such effective interaction. Finally, we consider the role of people as data consumers and acknowledge the importance of considering ethics in many aspects of data creation, access, and usage. The focus here is on finding new ways for maximizing the benefits of massive data while nevertheless safeguarding the privacy and integrity of citizens and societies.

We start with three papers that investigate different aspects of crowdsourcing. In "Toward Worker-Centric Crowdsourcing", Amer-Yahia and Roy argue that accounting for human factors, in particular workers' characteristics, in task assignment benefits both workers and requesters, and discuss new opportunities raised by worker-centric crowdsourcing. In the second paper, "Spatial Crowdsourcing: Challenges and Opportunities", Chen and Shahabi consider spatial aspects of crowdsourcing, in particular those related to the usage of mobile devices and users, and discuss the challenges and opportunities related to this important setting. In "Optimizing Open-Ended Crowdsourcing: The Next Frontier in Crowdsourced Data Management", Parameswaran, Sarma and Venkataraman examine open-ended crowdsourcing and survey existing work on formally reasoning about and optimizing this important, but relatively understudied class of crowdsourcing.

The next three papers focus on interactive data analysis. In "Interactive Data Exploration via Machine Learning Models", Papaemmanouil, Diao, Dimitriadou and Peng examine how learning-based exploration techniques can automatically steer the user towards interesting data areas, based on relevance feedback on database samples. In "Runtime Support for Human-in-the-Loop Feature Engineering Systems", Anderson, Antenucci and Cafarella take a closer look at feature selection for machine learning and discuss two projects that accelerate feature engineering by applying domain insights to engineer high-impact features. Finally, in "Towards a Benchmark for Interactive Data Exploration", Eichmann, Zgraggen, Zhao, Binnig and Kraska discuss the metrics that should be used for evaluating interactive data exploration systems and present ideas towards a new benchmark that simulates typical user behavior and allows such systems to be compared in a reproducible way.

Next, in "The Values Challenge for Big Data", Jagadish explores the characteristics of human involvement in Big Data management and proposes a research agenda to address associated challenges. Finally, we conclude by "HILDA 2016 Workshop: A Report", by Nandi, Fekete and Binnig that overview the Human-in-the-Loop Data Analytics (HILDA) workshop, that was held in association with ACM SIGMOD 2016.

I hope that you enjoy the issue as much as I enjoyed putting it together!

Tova Milo
Tel Aviv University

Toward Worker-Centric Crowdsourcing

Sihem Amer-Yahia
Univ. Grenoble Alpes, CNRS, LIG
F-38000 Grenoble, France
sihem.amer-yahia@imag.fr

Senjuti Basu Roy
New Jersey Institute of Technology
Newark, NJ, USA
senjutib@njit.edu

Abstract

Today, crowdsourcing is used to "taskify" any job ranging from simple receipt transcription to collaborative editing, fan-subbing, and citizen science. Existing work has mainly focused on improving the processes of task assignment and task completion in a requester-centric *way by optimizing for outcome quality under budget constraints. In this paper, we advocate that accounting for workers' characteristics, i.e.,* human factors *in task assignment and task completion benefits* both *workers and requesters, and discuss new opportunities raised by* worker-centric *crowdsourcing. This survey is based on a tutorial that was given recently at PVLDB [2].*

1 A Case for Worker-Centric Crowdsourcing

As more jobs are being "taskified" and executed on crowdsourcing platforms, the role of human workers online is gaining importance. On virtual marketplaces such as Amazon Mechanical Turk, PyBossa and Crowd4U, the crowd is volatile, its arrival and departure asynchronous, and its levels of attention and accuracy diverse. Tasks differ in complexity and necessitate the participation of workers with varying degrees of expertise. As workers continue to get involved in crowdsourcing, a legitimate question is how to improve both their performance and their experience. Existing proposals have been mostly concerned with the development of requester-centric algorithms to match tasks and workers and with preemptive approaches to improve task completion. We believe that new opportunities in developing models and algorithms are yet to be explored in bridging the gap between Social Science studies and Computer Science. Naturally, *understanding the characteristics of workers, here referred to as human factors*, that directly impact their performance and their experience on the platform, is a necessary step toward achieving that goal. We advocate a re-focus of research in crowdsourcing on how to best leverage human factors at all stages that will widen the scope and impact of crowdsourcing and make it beneficial to both requesters and workers. Several other complementary surveys could be found in the literature [3, 6].

Common tasks such as labeling images or determining the sentiment of a piece of text, can be completed by each worker independently. These types of crowdsourcing tasks are known as *micro-tasks*. An emerging area of interest is *collaborative crowdsourcing* where workers complete a task together. Examples include fan-subbing, where workers with complementary skills collaborate to generate movie subtitles in various languages, just hours after movies are made available. Disaster reporting is another example where geographically close people with diverse and complementary skills work together to report the aftermath of an earthquake. Section 2

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Worker-specific	Micro-tasks: *Skill, Reputation/Trust, Expected Pay, Acceptance Ratio* Collaborative tasks: *Affinity, Critical Mass, Interaction model*
Task-specific	*Feedback, Incentives, Skill Variety, Task Identity, Task Autonomy, Expected Quality, Budget, Desired Expertise*
Workers and Tasks	*Motivation*

Table 1: A rough characterization of human factors

reviews human factors for micro-tasks and collaborative tasks in conjunction with psychology studies conducted in the 70's in physical workplaces. It then reports on more recent empirical evaluation of human factors in virtual marketplaces. The outcome of that section is a review of existing approaches to model, acquire and learn human factors.

The two main processes that leverage human factors in virtual marketplaces are task assignment and task completion. Section 3 reviews algorithms and approaches for assigning tasks to workers and various approaches to intervene during task completion and improve overall performance. We review task assignment for both micro-tasks and collaborative tasks and draw a connection with findings in psychology. This connection brings an understanding of which factors are most likely to affect workers' choice of tasks and their performance during task completion.

The review we provide in Section 3 naturally leads to the second half of this paper. Section 4 is dedicated to new opportunities raised by leveraging human factors in crowdsourcing. This section exposes a number of promising directions that contribute to worker-centric crowdsourcing, a paradigm shift that we believe will lead to sustainable crowdsourcing.

2 Human Factors at Work

A variety of human factors characterize workers and their environment at work. Their genesis goes back to the 70's when "organization studies" and "work theory" were developing models to understand motivation in physical workplaces. A flagship study is that of Hackman and Oldham in 1976 whose goal was to determine which psychological states are stimulated by which job characteristics. The authors ran experiments on 658 employees in 62 heterogeneous jobs (white collar, blue collar, industry, services, urban and rural settings) in 7 organizations. The study showed that modeling extrinsic motivation such as how much a job pays, and intrinsic motivation such as whether a job provides feedback to workers, are critical for measuring workers' psychological state and hence their satisfaction and performance in the workplace.

We gathered the most common human factors from the literature and characterized them as *worker-specific*, *task-specific*, or specific to *both workers and tasks.* Table 1 contains a summary of the most common factors identified in both physical and virtual marketplaces.

In practice, human factors are mostly acquired via questionnaires and qualification tests. They can also be learned from workers' previous performance in completing tasks. We review the literature on modeling and acquiring human factors.

2.1 Worker-Specific Human Factors

In this subsection, we discuss the human factors that are related to workers. Only Skill and Reputation/Trust are discussed because Expected Pay is acquired directly from workers via a questionnaire, and Acceptance Ratio is computed as a proportion of tasks for which the worker's contribution has been accepted (out of all tasks the worker completed).

2.1.1 Skill and Reputation/Trust

Existing research has investigated the skill and trust estimation problem in several ways, primarily in the context of micro-tasks. For example, for labeling tasks, a probabilistic model was proposed to infer the true label of each image, the expertise of each labeler, and the difficulty level of each task [44]. For annotation tasks, Bayesian solutions were used to iteratively establish and refine a particular golden standard, measure the performance of annotators with respect to that standard, or eliminate spammers [34, 35]. For the same kind of tasks, a more recent work focused on determining worker confidence intervals and worker error rates [20]. A follow-up work [21] designed solutions for the case where not all workers have attempted every task, tasks have non-Boolean responses, and workers have different biases for making false positive and false negative errors for Boolean tasks. Additional work focused on the problem of identifying workers who systematically disagree both with the majority and with the rest of co-workers [41, 42].

In contrast to micro-tasks, there exists only one effort in estimating human factors in team-based tasks [33]. In that work, skill estimation is based on modeling task quality as an aggregation of individual worker skills and their collaboration effectiveness, and on solving an optimization problem under different skill aggregation functions, *sum, max, and min*. The optimization problem reverse-engineers worker skills and collaboration effectiveness from observed outcome quality.

2.2 Task-Specific Human Factors

In this subsection, we describe human factors that are pertinent to tasks, namely Feedback and Incentives. Skill Variety represents the number of different skills a task requires from a worker. Task Identity represents whether a task is part of a bigger task or not. Task Autonomy indicates if a worker depends on others. Expected Quality, Desired Expertise and Budget are used to set a minimum threshold on workers' contributions.

2.2.1 Feedback

The importance of task feedback is studied in CrowdFlower [4]. It was shown that both immediate and long-term feedback helps to improve the quality of completed tasks. This study also indicates that workers expect they should be provided with a meaningful explanation of why their work is rejected, or in case their work is accepted, they expect reasonable turnaround time between submitting the work and receiving payment for it.

2.2.2 Incentives

Incentives have been studied using qualitative and quantitative approaches. In [40], two different types of incentives are studied - social incentives and financial incentives. That work empirically shows that both types can improve worker productivity. A recent work [17] focuses on performance-based payments (PBP) through financial incentives. It empirically tests the effect of varying outcome quality threshold in order for workers to receive a bonus and the effect of varying the bonus amount on task quality. It also recommends running a pilot experiment to determine whether a task is effort-responsive and then design PBP schemes.

A quantitative study [8] presents algorithms for dynamic pricing to meet (a) a user-specified deadline while minimizing total monetary cost, or (b) a user-specified budget constraint while minimizing total elapsed time.

2.3 Worker- and Task-Specific Human Factors

Finally, we describe human factors that are pertinent to both workers and tasks. In this context, studying workers' motivation in completing tasks has been the center of attention. One of the earliest studies of motivation in virtual marketplaces was conducted on Amazon Mechanical Turk [24]. The goal of that study was to empirically verify which of several intrinsic and extrinsic motivation factors were considered important to workers. Figure 1

summarizes the results of an offline evaluation of 13 human factors related to motivation. The results were obtained by asking workers to fill out questionnaires after completing tasks. The goal of the questionnaires was to determine which task-specific factors, Skill Variety, Task Identity, etc., and which other factors, Social Contact, Human Capital Advancement, etc., affect motivation. While Payment remains a highly motivating factor, the study also points out the cumulative importance of "Enjoyment-Based Motivation" factors when compared to Payment. One other highlight is that a large proportion of workers declared that "Human Capital Advancement" was an important motivation in completing tasks.

A later study [37] found a synergistic interaction between intrinsic and extrinsic motivators and demonstrated that increasing levels of payment increases task throughput regardless of other factors. However, increasing task throughput does not necessarily mean that workers do a good job at completing tasks. It was indeed shown that increasing pay does not increase the quality of workers' contributions [26].

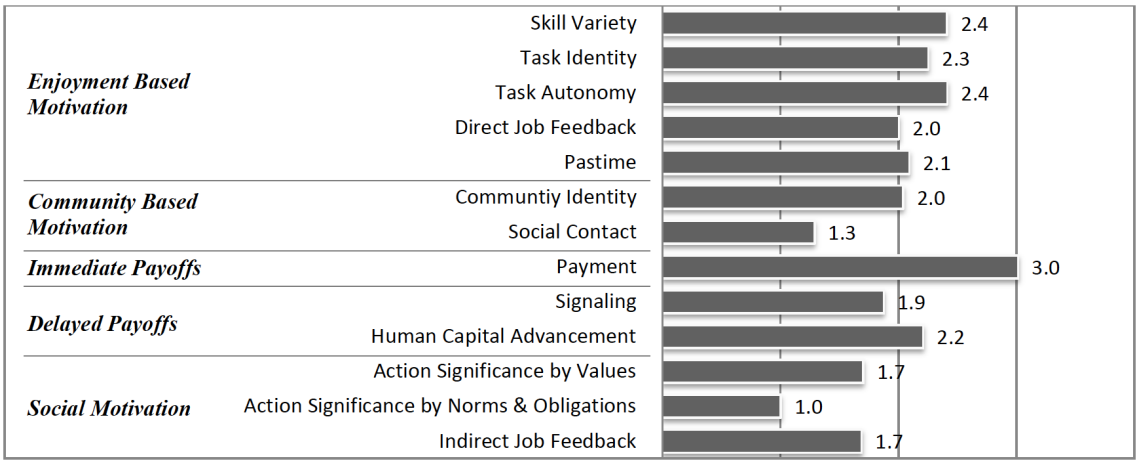


Figure 1: Results of an offline evaluation of human factors [24]

3 Human Factors in Task Assignment and Task Completion

Human factors are mostly recognized in a requester-centric fashion primarily for the purpose of task assignment and in a limited way for task completion. We present existing work in two parts - first, in the context of micro-tasks, then for collaborative tasks.

3.1 Micro-Task Assignment

Amazon Mechanical Turk only allows self-assignment to tasks - although task designers could specify desired worker qualifications for a given set of tasks via system-centric qualifications (e.g., HIT Approval Rate, also referred to as Acceptance Ratio) and platform-centric qualifications (e.g., marital status, education level, political affiliation, etc.).

Related work performs task assignment for micro-tasks, acknowledging primarily worker skill and budget. This body of work belongs to one of the following two kinds.

Researchers in Artificial Intelligence, Machine Learning, and Operations Research primarily assume that workers are distinguishable. Each individual worker has an associated id with a known skill and cost. Given a task with a budget and desired accuracy threshold, the task assignment problem is studied as a *matching* problem between workers and tasks [9, 15, 16, 60].

On the other hand, researchers in Databases have primarily assumed that workers are indistinguishable. A generic skill and cost model is assumed for all workers and a given set of tasks have an associated budget. Different types of datasourcing applications [5] are considered as human-machine intelligence tasks - such as filtering [28], sorting and join [25], deduplication and clustering [43], categorization [36], evaluating order queries [10], or even mining [1] involving human workers. Under this setting, the task assignment problem becomes that of selecting the best subset of tasks for the worker pool that satisfies the budget and is most likely to maximize quality.

3.2 Collaborative Task Assignment

Collaborative tasks need to be performed by a set of workers together as a team (e.g., collaboratively editing an article where each article has a minimum quality and a maximum cost requirement, as well as the need for complementary skills) and it is assumed that the workers' profiles are known (skill per domain, requested wage). In a recent work [38], the objective function is formalized so as to guarantee that each task surpasses its quality threshold, stays below its cost limit, and that workers are not over-utilized or under-utilized. Given the innate uncertainty induced by human involvement, a third human factor, Acceptance Ratio (e.g., computed as the probability that a worker accepts a recommended task) is also used in the problem formulation.

The experimental results presented in this paper indicate that despite assigning tasks to highly-qualified workers, for some cases, quality of completed tasks was low due to conflicting opinions, edit wars, or proliferation of edits. A follow-up work [33] proposes additional human factors to ensure that *teams are not too large* - in particular, it introduces *Upper Critical Mass* as a human factor that constrains the size of a team. For a given task, if the required number of workers are too many, then the objective is to form sub-teams, where both intra-team and inter-team collaboration is allowed. By leveraging related research in Psychology [14], the authors model collaborative aspects between workers as *Affinity*. Affinity is formalized using socio-demographic attributes, such as region, age, or psychological characteristics [27]. The objective function intends to form a set of teams for a collaborative task, where each sub-team must satisfy the critical mass constraint, their intra- and inter-team affinity is maximized, while satisfying a minimum quality requirement and a maximum cost budget. The problem is proved to be NP-hard and the authors propose a staged solution by designing approximation algorithms, where each stage individually admits an approximation factor.

3.3 Micro-Task Completion

Improving micro-task completion has mainly relied on monitoring worker motivation and taking appropriate actions to improve completion. Some efforts [7, 39] have shown that including a diversion or a break, such as showing an entertaining video, improves workers' motivation. More recently, pre-emption was exercised to interrupt workers who have not completed tasks on time [12].

3.4 Collaborative Task Completion

Failing to complete a micro-task has only "local" impact since it does not prevent other workers to complete that same task in parallel. Failing from completing a collaborative task is however more damaging since it leads to an uncompleted task for all. Despite its importance, monitoring task completion for collaborative tasks is still in its infancy. There exists one piece of work to date that addresses this problem [38]. This work proposes an adaptive task completion scheme that handles scenarios such as the arrival of new workers and tasks, and the departure of workers without finalizing tasks. The proposed solution is based on formulating a marginal IP problem that re-assigns tasks to workers when new workers or tasks arrive, or when workers leave. The assignment is based on the same objective functions described in Section 3.2.

4 Challenges and Opportunities

So far, we have discussed how human factors are modeled and monitored to optimize the main processes of a crowdsourcing platform, namely task assignment and task completion. In this section, we aim to widen the scope and impact of human factors in crowdsourcing and discuss how they can enable worker-centricity. In our opinion, it is essential to shift from requester-centric optimizations to an approach that integrates what workers want from a crowdsourcing platform. This shift will induce a tighter integration between human factors and the processes of a crowdsourcing system. In this section, we discuss some essential elements reailzing this shift. We start with the need for declarative tools that let workers benefit from crowdsourcing. We then examine the evolving nature of human factors and its impact on workers' performance. We move on to discuss evaluation and other factors that affect workforce organization and enabling experimental repeatability in crowdsourcing.

4.1 Declarativity

There have been several efforts in the database community to develop declarative languages that let requesters decompose complex tasks or specify task assignment criteria. Designing a declarative language that helps workers exploit the potential of a crowdcourcing platform appears as a natural goal. Workers should be able to express a number of desiderata such as acquiring or improving a specific skill, or being entertained for a specific period of time. The development of worker-centric primitives such as finding tasks of interest to a given worker or being notified when a particular requester posts tasks, opens new modeling and algorithmic opportunities that complement existing solutions for task assignment and task completion. Such primitives should be designed with an understanding of how worker-centric and task-centric human factors interleave and affect performance. Moreover, they should not hinder requester-centric optimizations. Rather, they should complement them. The overarching goal should be to bring together all the components of a crowdsourcing platform and benefit from an adaptive framework within which workers are observed and provided tasks that serve them and serve requesters. Such an integrative approach would close the loop between different crowdsourcing processes as shown in Figure 2.

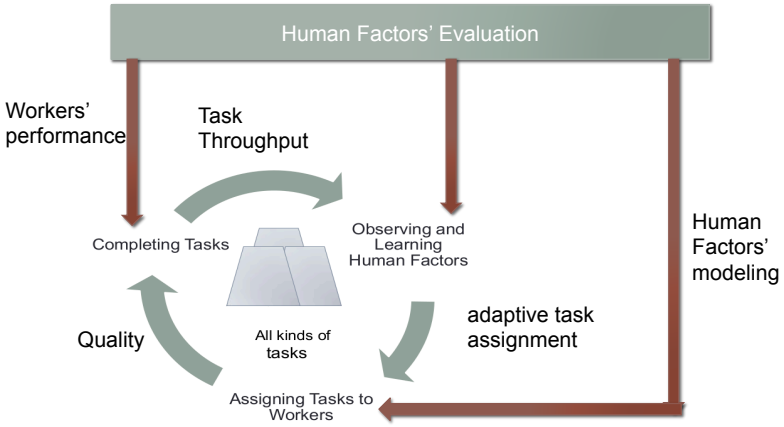
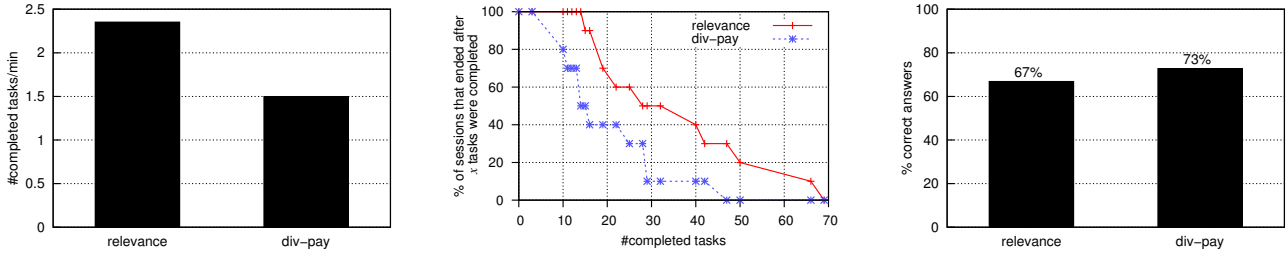


Figure 2: Adaptive task assignment and completion

4.2 Adaptivity

In practice, the evolving nature of human factors requires to re-think how they are integrated into crowdsourcing processes. The accuracy of such factors depends on the strategy used to acquire and refine them. Intuitively,

observing workers for a longer period of time should result in more accurate skill values. On the contrary, workers' motivation is task- and context-dependent, i.e., how long a task takes, what other tasks are present, or who else is involved in case of collaborative tasks. In short, motivation is more ephemeral than skills and the length of time required to learn a worker's motivation should be "shorter" than the length of time required to learn a worker's skill. Moreover, while workers' skills increase monotonically as they complete more tasks, motivation varies with time.



(a) Task throughput (b) Worker retention (c) Outcome quality

Figure 3: Workers' performance for relevance-based and diversity/payment-based task assignments

In a recent effort [29], we proposed observing workers as they completed tasks and learning their motivation as a combination between task diversity (intrinsic motivation) and payment (extrinsic motivation). The premise of this work was that workers' motivation evolves over time and that task assignment could be improved from one session to the next by monitoring workers and capturing their motivation as they choose and complete tasks. We compared two strategies: relevance-based, that provides workers tasks matching their skills, and diversity/payment-based, that provides them tasks achieving a combination of diversity and payment. We found that when measuring task throughput, akin to the number of tasks completed per time unit, and worker retention, akin to the likelihood of workers completing many tasks, the relevance-based strategy was superior (see Figures 3a and 3b). However, as shown in Figure 3c, the diversity/payment strategy resulted in contributions of higher quality when compared to a ground-truth. As a result, we could draw two conclusions: workers were faster at picking and completing tasks when no context switching was required, i.e., in the case where tasks were relevant and similar to each other. However, they generated higher quality contributions for tasks that optimized their motivation, i.e., the observed balance between task diversity and payment. Additional experiments can be found in [29].

These preliminary results allow us to argue for the need for adaptive crowdsourcing processes that incorporate human factors as they evolve.

4.3 Evaluation, Deployment Strategies, and Repeatability

Evaluating the performance of a crowdsourcing platform is a major concern that poses a number of challenges. The variety of profiles workers have and the diversity of tasks made available on a crowdsourcing platform raise the need for a careful evaluation. So, what is being evaluated and what are the approaches used for that? Evaluation in crowdsourcing has mostly focused on measuring two kinds of indicators. Requester-centric indicators are task throughput, the number of tasks completed per time unit, worker retention, the likelihood of workers completing many tasks, and payment. Worker-centric indicators are motivation and satisfaction.

Table 2 summarizes the evaluation protocols used in crowdsourcing and the performance indicators that are measured. Evaluation is usually performed in an offline or an online manner. Offline evaluation relies on questionnaires deployed before task completion to measure expected workers' performance, or after task completion, to measure their satisfaction. Online evaluation on the other hand, relies on actually deploying tasks to workers and measuring requester- or worker-centric performance indicators.

Protocols	Performance indicators
Offline: use questionnaires to measure worker-centric indicators *Online:* observe workers during task completion and measure performance	Worker-Centric: *Worker Expected Throughput, Worker Satisfaction* Requester-Centric: *Worker Retention, Task Throughput, Task Payment, Task Quality*

Table 2: Evaluation protocols and performance indicators in crowdsourcing

Longer worker retention does not necessarily imply higher outcome quality. The same could be said about all other indicators (task throughput, payment, workers' psychological state). For collaborative tasks, while it has been shown that team size affects outcome quality in the case of collaborative editing, there is no study on how different group interaction models affect quality. In practice, quality is evaluated in one of two ways: against a known ground-truth as in Information Retrieval, or using crowdsourcing. Note that a golden standard is not always available or possible. For example, in the case of text creation tasks, there does not exist a ground-truth and one has to resort to text evaluation criteria such as word error rate, clarity and completeness [38]. A more general approach is to crowdsource quality evaluation by asking another set of workers to evaluate potential ground truth answers. In the case of text creation for example, the accuracy of a text translation and the quality of the output text could be evaluated using traditional independent and comparative approaches.

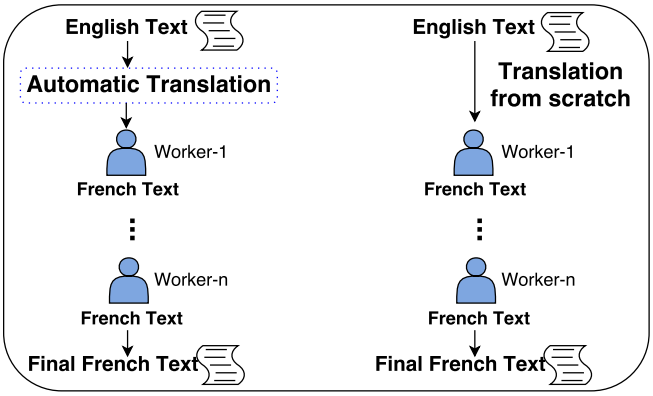


Figure 4: SEQ-IND-HYB & SEQ-IND-CRO

Much work is still needed in evaluating outcome quality for collaborative tasks. In Section 3.2, we discussed recent work that leverages group size and affinity between group members in the assignment of collaborative tasks to workers under quality and budget constraints [33]. In a recent piece of work [31], we also examined the usefulness of different skill aggregation functions in practice and validated them for a variety of tasks. A promising direction is the study of how different group dynamics and team interaction models [14] affect worker performance and outcome quality in crowdsourcing.

A deployment strategy defines the choices made to deploy a task. In worker-centric crowdsourcing, the question of how to organize the workforce becomes essential - thus deployment should become a center stage activity. A requester wishing to deploy a task makes a choice of how to combine algorithms and humans (crowd-only, or CRO vs hybrid, or HYB), a choice between sequential and simultaneous work structures (SEQ vs SIM), and a choice between an independent and a collaborative workforce organization (IND vs COL). In recent work [18, 30], we characterized different deployment strategies for text creation tasks such as translation and summarization. Figures 4 and 5 show some deployment options for English-to-French translation tasks. Our experiments measured outcome quality for different strategies and resulted in a set of guidelines to deploy

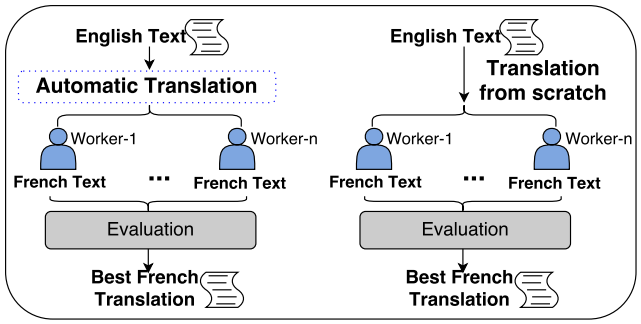


Figure 5: SIM-IND-HYB & SIM-IND-CRO

text translation and text summarization. For example, SEQ-IND-HYB (Figure 4), produces the best quality translation for long texts because workers prefer to start with an automatically translated text and are effective at improving each others' contributions in a sequential manner. The study of deployment strategies for other kinds of tasks and their impact on human factors, remains an open question.

Last but not least, *experimental repeatability* is a major concern in crowdsourcing and to the best of our knowledge, it has not received much attention. Repeatability in crowdsourcing is complex given the volatility of the crowd. Different times of day attract workers with different socio-demographics and in different time zones. Different platforms will also attract different workers. Different compensation schemes result in different behaviors [8]. Given this diversity, a pressing question is to define statistical significance in crowdsourcing. Some recent work [20,21] studied how to evaluate workers' quality with confidence interval - but using these proposed methods to obtain statistically significant output from the crowd remains an open question.

As a concluding remark, we envision that for the long term sustainability of crowdsourcing. In particular, we believe that deployment strategies should be made a primary focus of interest, followed by other worker-centric directions of research, such as collaboration, adaptability, and repeatability.

References

[1] Yael Amsterdamer and Tova Milo. Foundations of crowd data sourcing. In *SIGMOD Record*, 43(4):5–14, 2014.

[2] Sihem Amer-Yahia and Senjuti Basu Roy. Human Factors in Crowdsourcing. In *PVLDB*, Tutorial, 9(13):1615–1618, 2016.

[3] Anand Inasu Chittilappilly and Lei Chen and Sihem Amer-Yahia. A Survey of General-Purpose Crowdsourcing Techniques. In *IEEE Trans. Knowl. Data Eng.*, 28(9), pages 2246–2266, 2016.

[4] B. B. Bederson and A. J. Quinn. Web workers unite! addressing challenges of online laborers. In *CHI*, pages 97–106, 2011.

[5] L. Chen, D. Lee, and T. Milo. Data-driven crowdsourcing: Management, mining, and applications. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 1527–1529, 2015.

[6] Peng Dai and Mausam and Daniel S. Weld. Artificial Intelligence for Artificial Artificial Intelligence. In *AAAI*, 2011.

[7] P. Dai, J. M. Rzeszotarski, P. Paritosh, and E. H. Chi. And now for something completely different: Improving crowdsourcing workflows with micro-diversions. In *ACM CSCW*, pages 628–638, 2015.

[8] Y. Gao and A. G. Parameswaran. Finish them!: Pricing algorithms for human computation. *PVLDB*, 7(14):1965–1976, 2014.

[9] D. Geiger. Personalized task recommendation in crowdsourcing-Current state of the art. In *Decision Support Systems* 2014.

[10] B. Groz, T. Milo, and S. Roy. On the complexity of evaluating order queries with the crowd. *IEEE Data Eng. Bull.*, 38(3):44–58, 2015.

[11] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD Conference*, pages 385–396, 2012.

[12] Daniel Haas and Jiannan Wang and Eugene Wu and Michael Franklin. CLAMShell: Speeding up Crowds for Low-latency Data Labeling In *PVLDB* 2016.

[13] J. Hackman and G. R. Oldham. Motivation through the design of work: Test of a theory. *Organizational Behavior and Human Performance*, 16(22):250–279, 1976.

[14] G. Hertel and G. Hertel. Synergetic effects in working teams. *Journal of Managerial Psychology*, 2011.

[15] C. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, 2012.

[16] C. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *ICML*, pages 534–542, 2013.

[17] Ho, Chien-Ju and Slivkins, Aleksandrs and Suri, Siddharth and Vaughan, Jennifer Wortman. Incentivizing High Quality Crowdwork. In *WWW*, 2015.

[18] K. Ikeda, A. Morishima, H. Rahman, S. B. Roy, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Collaborative crowdsourcing with crowd4u. *Proceedings of the VLDB Endowment*, 9(13), 2016.

[19] J. J. Horton and L. B. Chilton. The labor economics of paid crowdsourcing. In *ACM EC*, pages 209–218, 2010.

[20] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Evaluating the crowd with confidence. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 686–694, New York, NY, USA, 2013. ACM.

[21] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Comprehensive and reliable crowd assessment algorithms. In *2015 IEEE 31st International Conference on Data Engineering*, pages 195–206. IEEE, 2015.

[22] R. Kenna and B. Berche. Managing research quality: critical mass and optimal academic research group size. In *IMA Journal of Management Mathematics*.

[23] D. Karger et. al. Budget-optimal task allocation for reliable crowdsourcing systems. In *Operations Research* 2014.

[24] N. Kaufmann, T. Schulze, and D. Veit. More than fun and money. worker motivation in crowdsourcing - A study on mechanical turk. In *AMCIS*, 2011.

[25] A. Marcus. et. al Human-powered sorts and joins. In *VLDB* 2011.

[26] Winter A. Mason and Duncan J. Watts. Financial incentives and the "performance of crowds". In *SIGKDD Exploration*, 11(2), pages 100–108, 2009.

[27] I. B. Myers and M. H. McCaulley. *Myers-Briggs Type Indicator: MBTI.* Consulting Psychologists Press, 1988.

[28] A. Parameshwaran et. Al CrowdScreen: Algorithms for filtering data with humans In *VLDB* 2010.

[29] Julien Pilourdault, Sihem Amer-Yahia, Dongwon Lee and Senjuti Basu Roy. Motivation-Aware Task Assignment in Crowdsourcing. In *Technical Report*, University of Grenoble-Alps, 2016.

[30] Ria Mae Borromeo, Maha Alsayasneh, Sihem Amer-Yahia and Vincent Leroy. Hybrid Deployment Strategies for Crowdsourced Text Creation Tasks. In *Technical Report*, University of Grenoble-Alps, 2016.

[31] Dalila Koulougli, Julien Pilourdault, Sihem Amer-Yahia, Allel Hadjali. Effect of Human Factors and Work Structure on Aggregating Workers Contributions in Crowdsourcing. In *Technical Report*, University of Grenoble-Alps, 2016.

[32] H. Rahman, S. Thirumuruganathan, S. B. Roy, S. Amer-Yahia, and G. Das. Worker skill estimation in team-based tasks. *Proceedings of the VLDB Endowment*, 8(11):1142–1153, 2015.

[33] H. Rahman, S. B. Roy, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Task assignment optimization in collaborative crowdsourcing. In *IEEE ICDM*, pages 949–954, 2015.

[34] V. Raykar, C. Vikas C. and Y. Shipeng. Eliminating Spammers and Ranking Annotators for Crowdsourced Labeling Tasks. In *J. Mach. Learn. Res.*, pages 491–518, 13(1), 2012.

[35] V. Raykar, S. Yu, L. Zhao, A. Jerebko, C. Florin, G. Valadez, L. Bogoni, and L. Moy. Supervised Learning from Multiple Experts: Whom to trust when everyone lies a bit. In *ICML*, pages 889–896. ACM, 2009.

[36] R. Rebeiro et al. Crowdsourcing subjective image quality evaluation *Int Conference on Image Processing 2011*.

[37] J. Rogstadius, V. Kostakos, A. Kittur, B. Smus, J. Laredo, and M. Vukovic. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

[38] Senjuti Basu Roy and Ioanna Lykourentzou and Saravanan Thirumuruganathan and Sihem Amer-Yahia and Gautam Das. Task assignment optimization in knowledge-intensive crowdsourcing. In *VLDB J.*, 24(4): 467–491, 2015.

[39] J. M. Rzeszotarski, E. H. Chi, P. Paritosh, P. Dai. Inserting Micro-Breaks into Crowdsourcing Workflows. In *AAAI* 2013.

[40] Shaw, Aaron D. and Horton, John J. and Chen, Daniel L. Designing Incentives for Inexpert Human Raters. In *CSCW* 2012.

[41] Soberón, Guillermo and Aroyo, Lora and Welty, Chris and Inel, Oana and Lin, Hui and Overmeen, Manfred. Measuring Crowd Truth: Disagreement Metrics Combined with Worker Behavior Filters. In *CrowdSem Workshop* 2013.

[42] Beatrice Valeri and Shady Elbassuoni and Sihem Amer-Yahia. Crowdsourcing Reliable Ratings for Underexposed Items. In *WEBIST* 2016.

[43] J. Wang et al. Crowder: Crowdsourcing entity resolution In *VLDB* 2012.

[44] Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., and Movellan, J. R. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.

[45] R. Yan, M. Gao, E. Pavlick, and C. Callison-Burch. Are two heads better than one? crowdsourced translation via a two-step collaboration of non-professional translators and editors.

Spatial Crowdsourcing: Challenges and Opportunities

Lei Chen [#], Cyrus Shahabi [$]
[#]Dept. of Computer Science & Engineering, HKUST
Hong Kong, P.R. China
leichen@cse.ust.hk
[$]Department of Computer Science, USC
California, U.S.A.
shahabi@usc.edu

Abstract

As one of the successful forms of using Wisdom of Crowd, crowdsourcing, has been widely used for many human intrinsic tasks, such as image labeling, natural language understanding, market predication and opinion mining. Meanwhile, with advances in pervasive technology, mobile devices, such as mobile phones, tablets, and PDA, have become extremely popular. These mobile devices can work as sensors to collect various types of data, such as pictures, videos, audios and texts. Therefore, in crowdsourcing, a requester can unitize power of mobile devices and their location information to ask for data related a specific location, subsequently, the mobile users who would like to perform the task will travel to the target location and collect the data (videos, audios, or pictures), which is then sent to the requester. This type of crowdsourcing is called spatial crowdsourcing. Due to the pervasiveness of mobile devices and their superb functionality, spatial crowdsourcing is gaining more attention than general crowdsourcing platforms, such as Amazon Turk(http://www.mturk.com) and Crowdflower (http://crowdflower.com/). However, to develop a spatial crowdsourcing platform, effective and efficient solutions for motivating workers, mining workers' profiles, assigning tasks, aggregating results and controlling data quality must be developed. Therefore, in this paper, we will discuss the challenges and opportunities related to these key techniques, including 1) effective incentive mechanisms to encourage mobile device users to participate in crowdsourcing tasks; 2) automatic user profile mining methods; 3) optimal task assignment solutions; 4) novel answer aggregation models; 5) intelligent data quality control mechanisms.

1 Introduction

Recent advances in Internet and pervasive computing technology make tasks such as image tagging, language translation, and speech recognition easier to achieve by humans than by machines. Crowdsourcing, as one of the successful forms of utilizing human intelligences, has become quite popular recently. Basically, there are three components in crowdsourcing, a requester, an open call platform and workers. The requester will submit his/her tasks through the open calls and workers who are ready are assigned to perform the task. The

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

results returned from the workers are aggregated and returned to the requester. In fact, crowdsourcing is human computation [32]and social computing [25] powered by a crowdsourced workforce. Crowdsourcing overlaps with several other topics like social computing, human computing, collective/collaborative intelligence, etc. It provides a new problem-solving paradigm [2, 38] and has branched into several areas.

Recently, smart phones have become quite popular and they have become people's lifemate. People can easily identify and participate in some location-based tasks that are close to their current positions, such as taking photos/videos, repairing houses, or preparing a gathering at a specific location. Therefore, a new framework, namely spatial crowdsourcing [23], for employing workers to conduct spatial tasks, has emerged in both academia (e.g., gMission [7] and MediaQ [24]) and industry (e.g., TaskRabbit). A typical spatial crowdsourcing platform assigns a number of moving workers to perform spatial tasks nearby, which requires workers to physically move to some specified locations and accomplish these tasks. With spatial crowdsourcing, people can well utilize the human intelligence in any places at any time.

Though many benefits can be brought through spatial crowdsourcing, building such a platform is not trival. In this paper, we will present challenges and opportunities related to the core techniques for spatial crowdsourcing along four dimensions: worker and task registration, task assignment, result aggregation, and data quality control. Specifically, we will discuss effective incentive mechanisms to motivate registered workers, automatic user profile mining tools, optimal task assignment mechanisms, market-driven data aggregation solutions, and movement pattern-based data quality control methods. Finally, we will present some privacy-related issues related to spatial crowdsourcing.

2 Overview of Spatial Crowdsouring Platform

Figure 1 shows a general overview of a spatial crowdsourcing system, such as gMission [7] and MediaQ [24]. Basically, there are four phases: 1) Registration (requesters and workers), 2) Task Assignment, 3) Answer Aggregation, and 4) Response & Quality Control. In the registration phase, the task requesters submit their spatial crowdsourcing tasks to the spatial crowdsourcing server (SCS) and workers will submit their profile information, including their locations and expertise, to the SCS and indicate their willingness to work on the tasks. The challenging issues in this phase are how to motivate workers' participation and how to adjust system configuration dynamically based on the current setting. In the task assignment phase, the SCS will assign the tasks to the workers based on the budget, location and other constraints. The task assignment is very important, and directly affects the system throughput, i.e., how many tasks can be well handled simultaneously. In this phase, we have to design solutions to address the challenges of insufficient knowledge about workers' profiles and expensive cost (NP-hard) to finding the optimal assignment. In the answer aggregation phase, the SCS will use different data integration models to aggregate the answers collected from the workers. Since the workers have different backgrounds, abilities, and different understanding of the tasks, the results returned by them may vary much. Therefore, we need to find a suitable model and design efficient solutions to aggregate the results. In addition, we need to address the challenges to handle fake results returned by malicious works, such as images or videos downloaded from Internet not captured by the worker himself/herself. In the response and quality control phase, the SCS will send the final answers to the requesters and estimate the quality of the returned results using various methods. In the same phase, the SCS will determine whether more workers should be hired to perform the same task to improve the quality. The challenging issue in this phase is that we often do not have the ground truth (gold standard) for the crowdsourcing tasks. Although there are a few previous works on spatial crowdsourcing [1, 23, 28] all of them focused only on task assignment. In the rest of this paper, we highlight the research problems and possible solutions to the challenges for each phase of the SCS.

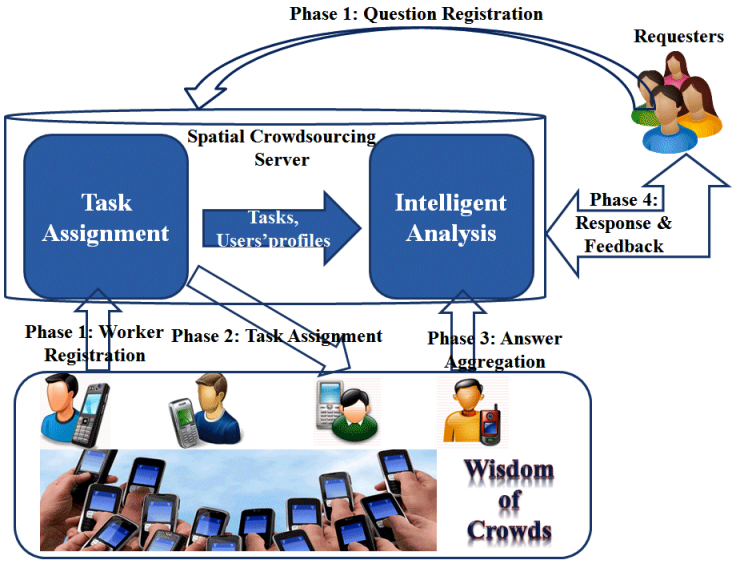


Figure 1: Overview of the Spatial Crowdsourcing System

3 Task and Worker Registration

For requesters, it is often not an issue for them to register their tasks in the SCS as long as a user friendly interface is provided. However, for normal mobile users, coming up with effective incentives to encourage their participation is quite challenging and an interesting problem, since participation often means consumption of their own resources, such as computation and communication resources. Generally, the existing incentive mechanisms can be divided into four categories: entertainment-based, service-based, social facilitation-based and monetary-based. For entertainment-based mechanisms [15, 20, 33], the system tries to gamify the tasks, such that users can contribute computation or sensing abilities of their smartphones when they play games. Here "gamify" means using game elements in non-game contexts [20]. This paradigm allows users to enjoy performing tasks, but the challenge lies in making sure that the gamified tasks are interesting enough. The service-based mechanisms are rooted in the mutual beneficial principle, where service consumers are also service providers. In other words, if a user wants to benefit from the service system, he/she also has to contribute to the service system [19, 27]. Social facilitation-based mechanisms take advantage of people's tendency to perform better in simple tasks if they are being watched. Thus to encourage people to participate in certain tasks, the effort of each participant should be easily observed by others [11, 30]. The last category is monetary incentives [18, 29, 31]. In this case, the system has to offer a certain amount of money to motivate potential participants such that the participants will use their resources, usually smartphone cameras or sensors, to complete the tasks. However, the existing monetary systems mostly assume static settings, i.e., when the interaction between a platform and users starts, there must exist a certain number of users, such that the current strategies can be applied. This setting is referred to as an offline setting. However, for most location-based services, an online setting is more practical. Therefore, it is necessary to investigate different incentive mechanisms in the online setting. Specifically, the following two issues should be further investigated:

a) Assigning different types of incentives for different tasks.

Spatial crowdsourcing tasks may involve various different tasks, ranging from simple text-based voting (decision making), image capturing, video or audio recording, to sophisticated product ranking and recommendation. Based on the characteristics of different tasks, especially the cost of performing such tasks (image or video uploading), it is necessary to design different incentives from which the requester can choose. Based on the collected log data, we can compare the similarity between the tasks in the same category and offer different

incentives. The challenge in performing such assignment is how to measure task difficulty. Only when the SCS knows the difficulty of the task can it offer suggestions regarding the type and amount of incentives that the requester should offer. Offering a small amount of incentives for a very difficult task may result in no workers taking up the task. On the contrary, giving out a large amount of incentives for a rather easy task may attract malicious workers. Therefore, how to measure the difficulty of the task is quite important. Some previous works consider the difficulty of a crowdsourcing task as a parameter in their models [26, 39]. However, it is still an open question as to how to quantify the difficulty of given crowdsourcing tasks. Even for the same class of tasks, the difficulty of individual tasks may be different. Given a set of crowdsourcing tasks, we first investigate the following measures for capturing the semantic of "difficulty" of text-based tasks.

- Entropy - If we consider the answer of a crowdsourcing task as a random variable, then each crowdsourced answer is simply the outcome of one experiment on the random variable. Under this model, difficult crowdsourcing tasks would lead to high randomness of the answer. Since entropy is a typical measure of randomness, it is a fair tool to estimate the magnitude of difficulty.

- Inter-rater reliability - Inter-rater reliability is the degree of agreement among workers. With the increase in difficulty, the degree of agreement among workers significantly decreases. Hence, the inter-rater reliability is also a reasonable measure of the difficulty. Given these two difficulty measures and their values computed from historical trace of tasks, we can estimate the difficulty of the current task by combining the difficulties of a few very similar historical tasks, The similarity function between two tasks can be a simple hamming distance or a sophisticated earth mover's distance. After completing the text-based tasks, we study how to measure the difficulty of capturing multimedia content (image, video and audio). Our initial idea is to measure the similarity between the current task and historical tasks, with the consideration of the size and resolution of the data.

b). Adjusting the online setting.

Since both tasks and workers in the system change dynamically, we need to propose solutions to address the data dynamics. Most of the workers in spatial crowdsourcing are mobile users. The characteristics of mobile devices (e.g. where a device is located) lead to the uniqueness of the spatial crowdsourcing workers. In [9], we have proposed a grid-based predication method to estimate spatial distributions of workers/tasks in the future. In particular, we consider a grid index, $\mathcal{I}$, over tasks and workers, which divides the data space $\mathcal{U}$ into γ^2 cells, each with the side length $1/\gamma$, where the selection of the best γ value can be guided by a cost model in [10]. Then, we estimate possible workers/tasks that may fall into each cell at a future timestamp, which can be inferred from historical data within the most recent sliding window of size w. After that, we assign workers with spatial tasks for multiple rounds (within a time interval P), based on the predicted location/quality distributions of workers/tasks.

Specifically, for each round at timestamp p, we first retrieve a set, T_p, of available spatial tasks, and a set, W_p, of available workers. Here, the task set T_p (or worker set W_p) contains both tasks (or workers) that have not been assigned in the last round and the ones that newly arrive at the system after the last round. In order to avoid the local optimality at the current timestamp p, we need to predict location/quality distributions of workers and tasks at a future timestamp $(p+1)$ that newly join the system, and obtain two sets W_{p+1} and T_{p+1}, respectively.

In particular, our grid-based prediction algorithm first predicts the future numbers of workers/tasks from the latest w worker/task sets in cells, and then generates possible worker (or task) samples in W_{p+1} (or T_{p+1}) for cells of the grid index $\mathcal{I}$. First, we initialize a worker set W_{p+1} and a task set T_{p+1} in the future round with empty sets. Then, within each cell $cell_i$, we can obtain its w latest worker counts, $|W_{p-w+1}^{(i)}|$, $|W_{p-w+2}^{(i)}|$, ..., and $|W_p^{(i)}|$, which form a *sliding window* of a time series (with size w). Due to the temporal correlation of worker counts in the sliding window, in this paper, we utilize *linear regression* over these w worker counts to predict the future number, $|W_{p+1}^{(i)}|$, of workers in cell, $cell_i$, that newly join the system at timestamp $(p+1)$. Note that,

other prediction methods can be also plugged into our grid-based prediction framework, and we would like to leave it as our future work. Similarly, we can estimate the number, $|T_{p+1}^{(i)}|$, of tasks in $cell_i$ for the next round at timestamp $(p+1)$. According to the predicted numbers of workers/tasks, we can uniformly generate $|W_{p+1}^{(i)}|$ worker samples (or $|T_{p+1}^{(i)}|$ task samples) within each cell $cell_i$, and add them to the predicted worker set W_{p+1} (or task set T_{p+1}). Finally, we return the two sets W_{p+1} and T_{p+1}.

Note that, random samples of workers/tasks in cells can approximately capture future location distributions of workers/tasks. However, in each cell, discrete samples may be of small sample sizes, which may lead to low prediction accuracy. Therefore, we alternatively consider continuous *probability density function* (pdf) for location distributions of workers/tasks.

4 Task Assignment

In task assignment, we need to design solutions to assign spatial crowd sourcing tasks to workers. Like traditional crowdsourcing, spatial crowdsourcing also aims to assign the tasks to workers economically and efficiently. By economically we mean that all the tasks are to be achieved within some budget, and by efficiently we mean that all the assigned tasks should be accomplished as soon as possible. To achieve these two goals, we need to address the following two key issues.

a) Building mobile users' (workers') profiles.

After the mobile users register with the SCS as workers, other than the registration data that they provide, we need to enhance their profiles by mining the tasks they have performed. This mining step is critical since the workers often do not express explicitly what type of questions (domains) they can answer during the registration. Furthermore, without feedback from the requesters, we cannot know whether the workers will perform a good job. In addition, based on the time that workers spend to answer various tasks, we can derive a latency model for each worker. Therefore, in this step, for each user, we can conduct mining on both tasks that he/she has performed and the evaluation reports (satisfactory/unsatisfactory and time cost to solve the tasks) to generate a preferred task domain and a time latency model of the worker. In our previous work [4]. we have used data mine methods to mining social users' profile. A similar idea can be applied to build worker's profile for spatial crowdousrcing platform. In gMission [7], we treat each well-performed task as a sample data point and conduct an EM algorithm to derive the task (topic) distribution of the worker. We can also infer the user's ability to conduct various event capturing tasks such as movement speed of the worker, resolution of the camera and precision of GPS devices [9].

For the latency model, we develop a two-phase model [6]. The first phase is the period from the task being published to the task being chosen by a worker; the second phase is the period from the task being accepted by a worker to the answer being returned. Statistical research has been conducted on several crowdsourcing platforms to capture the traits of such latencies [17, 29, 42], such as the one defined as follows.

Definition 1 (Latency): The On-hold Latency L_o of a task (or a batch of tasks) is the clock time from when the task is published to the time when it is accepted by a worker. The Processing Latency L_p of a task (or a batch of tasks) is the clock time from when the task is accepted to the time when the answer is returned and collected by the system. The Overall Latency L is the sum of L_o and L_p: $L = L_o + L_p$.

b) Task Assignment

Based on publishing modes of spatial tasks, the spatial crowdsourcing problems can be partitioned into another two classes: *worker selected tasks* (WST) and *server assigned tasks* (SAT) [23]. In particular, WST publishes spatial tasks on the server side, and workers can choose any tasks without contacting the server; SAT collects location information of all workers to the server, and directly assigns workers with tasks. For example, in the WST mode, some existing works [1,14] allowed users to browse and accept available spatial tasks. However,

there often exist the cases that tasks with low reward or far from the workers will not be served. On the other hand, in the SAT mode, our previous works [23] assumed that the server decides how to assign spatial tasks to workers and the solutions only consider simple metrics such as maximizing the number of assigned tasks on the server side. Compared with WST, SAT is more popular and can control the system throughput and offer balanced workload for each worker.

In SAT, to assign a task to the right crowd, the first question to address is what makes the answers from the crowd wise. Fortunately, a clear solution is provided in [23]. As indicated in [23], four key elements separate wise crowds from irrational ones: Diversity - Each person should have private information even if it's just an eccentric interpretation of the known facts. Independence - People's opinions are not determined by the opinions around them. Decentralization - People are able to specialize and draw on local knowledge. Aggregation- Some mechanism for turning private judgments into a collective decision. Intuitively, independence, decentralization and aggregation can be easily achieved with an appropriate question-answering framework. For instance, a requester may push a yes-no task to a deterministic group of users. Without publishing any answers during the crowdsourcing process, the users would work on the tasks with independence and decentralization. Therefore, the major challenge left is to find an appropriate group of users with high diversity. To achieve this, we need to measure the similarity among users, which again relates to the profiles of users. Considering user's profile as a multi-attribute vector, we can apply different similarity measures for different attributes and combine them by assigning different weights. For example, we can use the Euclidean distance to measure the similarity of users' positions and Cosine similarity for users' expertise which are represented by a set of keywords.

Based on users' profiles and measures about users' diversity, we can work on solving the issue of task assignment. Specifically, given a set of spatial crowdsourcing tasks, we want a strategy to assign tasks to workers to achieve the maximum number of task assignment in real time and meanwhile reduce the cost of each participating workers in terms of their travel distances to the specified crowdsourcing location. Clearly, due to the dynamic properties of workers, we cannot obtain a global picture of all workers at a specific point in time. Therefore, we try to obtain local optimal assignment solutions. Moreover, there are many constraints to assigning tasks, such as the locations (specifying where the task should be performed), the expertise (the domains that a worker knows well), the latency (the time taken to complete the task), reliability (the success rate or the confidence with which a worker can perform certain tasks) and diversity of the workers (dissimilar and independent workers). [23] studied the spatial crowdsourcing with the goal of static maximum task assignment, and proposed several heuristics approaches to enable fast assignment of workers to tasks. Similarly, Deng et al. [14] tackled the problem of scheduling spatial tasks for a single worker such that the number of completed tasks by this worker is maximized. As proved in [23], the task assignment problem in SAT is NP-hard and we can focus on designing various heuristic and approximate solutions to conduct the optimal assignment.

We further study the spatial-temporal diversity and reliability in task assignment [10]. The reliability indicates the confidence that at least some worker can successfully complete the task, whereas the spatial/temporal diversity reflects the quality of the task accomplishment by a group of workers, in both spatial and temporal dimensions (e.g., taking photos from diverse angles and at diverse timestamps). We prove that task assignment with the consideration of spatial-temporal diversity and reliability is also NP-hard, and thus we propose three approximation algorithms (i.e., greedy, sampling, and divide-and-conquer). In addition to handle tasks which require single-skilled workers, we investigate solutions to handle complex tasks requiring multi-skilled workers [8]. For such a complex task assignment, we want to maximize an assignment score (i.e., flexible budget, given by the total budget of the completed tasks minus the total travelling cost of workers). Moreover, we also need to consider several constraints, such as skill-covering, budget, time, and distance constraints, which make the complex assignment problem more challenging. Again, we prove that the multi-skilled complex task assignment problem is NP-hard and intractable. Therefore, we propose three effective heuristic approaches, including greedy, g-divide-and-conquer and cost-model-based adaptive algorithms to get worker-and-task assignments.

The task assignment solutions that we discussed so far focus on offline scenarios, where the server knows all the spatiotemporal information of tasks and workers in advance. However, this offline scenario is quite un-

realistic since tasks and workers in real applications appear dynamically and their spatiotemporal information are often unknown in advance. Thus, in our recent work [41, 44], we have investigated solutions for online task assignment. In [41], to address the shortcomings of existing offline approaches, we first identify a more practical micro-task allocation problem, called the Global Online Microtask Allocation in spatial crowdsourcing (GOMA) problem, then, we extend the state-of-art algorithm for the online maximum weighted bipartite matching problem to the GOMA problem as the baseline algorithm. We further develop a two-phase-based framework targeting on the average performance of online assignment algorithms. Based on the proposed framework, we present an efficient algorithm with $1/4$-competitive ratio under the online random order model. To improve its efficiency, we further design the TGOA-Greedy algorithm, which runs faster than the TGOA algorithm but has lower competitive ratio of $1/8$. In [44], aiming at offering mutual benefit to both workers and tasks, we propose a task assignment framework, called task assignment with mutual benefit awareness (TAMBA). TAMBA captures both sides preferences based on the historical data. With the awareness of both sides preferences, the tasks are assigned to the workers in order to achieve the maximum mutual benefit. Specifically, TAMBA is built on top of a crowdsourcing platform (e.g. gMission), and consisted of two core components: the preference estimator and the AMMB assignor. The preference estimator is built based on a probabilistic graphical model, which is trained offline with historical data (the training of the graphical model is offline conducted and NOT a part of the assignment process). When tasks are submitted, it generates the preference scores (presented in the next section) between the submitted tasks and all the workers. With the preference scores, the AMMB assignor allocate the tasks to the workers based on the output of AMMB assignor. To optimize the performance, we have developed algorithms for the offline and online settings of the same assignment, respectively.

5 Answer Aggregation

After collecting the data from the workers, our next task is to aggregate the results from the workers. Depending on the types of crowdsourcing tasks, different data aggregation models should be investigated. Markets have been proven to be an effective institution for aggregating beliefs of users and yielding reliable answers. For example, in racetrack betting [16], investors are only allowed to choose from two options, and the promised rewards are only given to the investors whose investment meets the market opinion (the majority of all investors). We apply this idea to the aggregation model of workers' answers [5]. Here, workers are considered as investors in this market, thus a wise market includes a set of investors, each of which is associated with a probability (individual confidence) that he or she will give the correct answer. This confidence can be mined from the historical results contributed by the workers. The SCS sever maintains a pool of candidate workers while evaluating the worker' confidence simultaneously. When a particular task comes along, the SCS server uses the solutions proposed in the task assignment step to find an optimal subset of workers and releases the task to them with promise of a reward. After receiving their choices, the answer preferred by a majority of the workers will be given as the market opinion and the ones who make the same choice as the market are granted a reward. Note that the output is the majority of the market, which is not necessarily the same as the ground truth. Compared to simple majority voting, only the workers who make the same choice as the majority will get the reward. This mechanism helps prevent malicious workers, who might return random answers for the reward. With this market aggregation model, the key issue is how to compute the result confidence efficiently. The market confidence is the probability that the aggregation result of C investors (C is larger than half of the selected crowd) is the same as the ground truth. We can use divide-and-conquer strategy to recursively divide the crowd into two groups and compute the confidence for each group. Note that the above discussed aggregation model is designed for simple decision tasks. In [3], we extend the idea of changing "workers" to "traders" to a more complex problem, opinion elicitation. We use Bayesian updating, beginning with our initial guess as the prior, to obtain a posterior distribution that reflects the weighted opinions of all the traders in the market. The payoff for each trader is proportional to her contributed modification from the prior to the posterior.

However, for spatial crowdsourcing, there may exist needs to perform more complicated tasks, such as ranking or capturing events. Therefore, we need to investigate the following issues:

1. Aggregation module for the ranking tasks: since each worker will only return partial ranks, we can apply different voting rules, such as Copeland, maximin, Bucklin, and ranked pairs, or even considering the distance from the worker to the task as the weighting factor [21], to aggregate the answers. However, it was proved that aggregation under these rules is NP-complete [43]. Thus we need to investigate efficient heuristic or sampling solutions.

2. Aggregation module for capturing some event or scenery: since malicious workers may post fake images or videos possibly downloaded from Internet to obtain the reward, we need to design solutions to aggregate them effectively and efficiently. To find such images or videos, we can extend our solutions in detecting near duplicated images or videos [45] to remove those that have appeared a few times.

6 Responses and Quality Control

In the last step, the aggregated results will be returned to the requester. However, there is a big and challenging problem about the quality of the returned results. From the database point of view, crowdsourcing applications break the Closed World assumption that data not available in the database can be considered as non-existing. Data quality measurement for Collected Contents varies according to the specific usage. To deal with the data quality challenge, many data-driven solutions have been proposed. The data-driven approaches are mostly compelled by the data collected so far or the data to be collected. The philosophies of designing such solutions vary according not only to specific tasks they are tackling, but also the form of the quality measurement. Amazingly, success in developing such solutions depends on the coherent matching between the reasonable understanding of the uncertainty, and the proper mathematical (probabilistic) model that agrees with the semantics. We can investigate these quality issues along the following two dimensions:

(a). Probabilistic Data Fusion

Probabilistic Data Fusion approaches are mostly adopted in the crowdsourcing scenario when the workers are asked to choose one or multiple answers from a set of candidates [26]. Two intrinsic observations of such scenario are:

1) There exists the ground truth (or real correct answer) for such decision-making problem, no matter whether the truth is latent or obvious;

2) There are workers with low quality, but there are no spam or biased workers. The two observations lead to a fundamental mathematical description of the uncertainty: The correctness of each worker (w_i)'s answer is a Bernoulli random variable, with identifiable accuracy ϵ_i. Given a voting scheme (e.g. the simplest one, majority voting) and a set of n workers, $W_n = (w_1, w_2, , w_n)$, the overall answer is:

$$OA(W_n) = \begin{cases} 1 & \text{if } \sum w_i \geq \lceil \frac{n}{2} \rceil \\ 0 & \text{if } \sum w_i < \lfloor \frac{n}{2} \rfloor \end{cases}$$

Not surprisingly, the aggregator does not rule out any error-prone ingredients, but installs a straightforward synthesizer. If the answers from different workers are i.i.d. the overall data quality (OQ) can be measured as the reliability of the outcome from Majority Voting:

$$\begin{aligned} OQ(W_n) &= \Pr(OA(W_n) = G) \\ &= \Pr(\big|C\big| \geq \lceil \frac{n}{2} \rceil) \end{aligned}$$

where C is the number of correct workers and G stands for the ground truth. Despite the rigorous definition of data quality, the challenge remains in a practical way. The number of correct workers is essentially a random variable following Poisson Binomial Distribution, on which even the probability density function is intractable.

(b). Data Cleaning

Data cleaning is another way to deal with conflicting crowdsourced data. Compared to data fusion approaches, the data cleaning methods rely relatively less on the probabilistic semantic, internal consistency and dependency receive more consideration,. In one word, the conflict among the data is due to the existence of wrong tuples, and the correct answers are just self-justified. The goal of data cleaning is thus to eliminate wrong items via the internal consistencies or dependencies. We can employ probabilistic graphical models [12, 13] to capture the data quality issues derived from intricate correlation and apply our previous work on matching dependencies [34–37] to clean the results.

Neither approach above takes into consideration the movement patterns of mobile users. Very often, the users follow the same movement pattern in weekdays (commuting from home to the office). Based on these movement patterns, we can easily tell whether the data provided by the worker are correct or not according to her usual location at that time. Thus, we can incorporate movement patterns (along spatio-temporal domain) into quality control solutions.

7 Privacy Issues

Other than the four topics that we have discussed above, privacy is an important issue that needs to be addressed along all the steps in spatial crowdsourcing applications. During the process of spatial crosscutting, users, especially workers' position information need to be released in order to assign the tasks to the nearby workers. However, workers' position and their moving trajectory information are very sensitive information, which can be used to identify individual workers. Therefore, some of our previous works [22,40] investigate how to tackle the privacy leakage problem in spatial crowdsourcing. In [22], we first formally define the problem of privacy leakage in spatial crowdsourincg system and identify its unique challenges due to the existence of an un-trusted central data server. We propose a privacy-aware framework by submitting group and independent queries instead of individual and correlated ones, which enables participation of users without compromising their privacy. In [40], again we assume the existence of a trusted server named *cellular service provider*, we propose a spatial crowdsourcing framework that can achieve an acceptable assignment success rate and total worker travel distance without privacy leakage. Workers first submit their accurate locations to the cellular service provider, then the provider desensitize these locations to a private spatial decomposition for the crowdsourcing server to access. When a new task comes, the crowdsourcing server issues some region queries to the Private Spatial Decomposition (PSD) to determine a specific region to notice. And at last we use a technique named geocast to send the task information to this region.

8 Conclusion

Spatial Crowdsourcing, as a new direction for crowdsourcing has become more and more popular and has been applied in many real applications. In this paper, we first give an overview of the spatial crowdsourcing framework and then illustrate the challenges and opportunities of the spatial crowdsourcing step by step, including incentive design, task assignment, answer aggregation and quality control. Finally, we have presented some works on privacy-preserved spatial crowdsourcing. Given the popular usage of smart phones and advanced development of AI, we believe there will be many challenges and opportunities for spatial crowdsourcing applications.

References

[1] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. Location-based crowdsourcing: Extending crowdsourcing to the real world. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 13–22, 2010.

[2] D. C. Brabham. Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, 14(1):75–90, 2008.

[3] Caleb Chen Cao, Lei Chen, and Hosagrahar Visvesvaraya Jagadish. From labor to trader: Opinion elicitation via online crowds as a market. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1067–1076, 2014.

[4] Caleb Chen Cao, Jieying She, Yongxin Tong, and Lei Chen. Whom to ask?: Jury selection for decision making tasks on micro-blog services. *Proc. VLDB Endow.*, 5(11):1495–1506, July 2012.

[5] Caleb Chen Cao, Yongxin Tong, Lei Chen, and H. V. Jagadish. Wisemarket: A new paradigm for managing wisdom of online social users. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 455–463, 2013.

[6] Chen Cao, Jiayang Tu, Zheng Liu, Lei Chen, and H. V. Jagadish. Tuning crowdsourced human computation. *Arxiv.org Technical Report*, page https://arxiv.org/submit/1693767, 2016.

[7] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. gmission: A general spatial crowdsourcing platform. *Proc. VLDB Endow.*, 7(13):1629–1632, August 2014.

[8] Peng Cheng, Xiang Lian, Lei Chen, Jinsong Han, and Jizhong Zhao. Task assignment on multi-skill oriented spatial crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2201–2215, 2016.

[9] Peng Cheng, Xiang Lian, Lei Chen, and Cyrus Shahabi. Prediction-based task assignment on spatial crowdsourcing. *Arxiv.org Technical Report*, page https://arxiv.org/abs/1512.08518, 2016.

[10] Peng Cheng, Xiang Lian, Zhao Chen, Rui Fu, Lei Chen, Jinsong Han, and Jizhong Zhao. Reliable diversity-based spatial crowdsourcing by moving workers. *Proc. VLDB Endow.*, 8(10):1022–1033, June 2015.

[11] Antin J. Cheshire, C. The social psychological effects of feedback on the production of internet information pools. *Comput. Mediat. Commun*, 13:705–727, 2008.

[12] Mausam Christopher H. Lin and Daniel S. Weld. Crowdsourcing control: Moving beyond multiple choice. In *Proceedings of the 2012 Uncertainty in Artificial Intelligence*, UAI '12, pages 491–500, 2012.

[13] Peng Dai, Mausam, and Daniel S. Weld. Decision-theoretic control of crowd-sourced workflows. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 1168–1174, 2010.

[14] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 324–333, 2013.

[15] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. using game-design elements in non-gaming contexts. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 2425–2428, 2011.

[16] David A. Easley and Jon M. Kleinberg. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[17] Siamak Faridani, Björn Hartmann, and Panagiotis G Ipeirotis. What's the right price? pricing tasks for finishing on time. In *Hcomp 2011*.

[18] David Geer. E-micropayments sweat the small stuff. *Computer*, 37(8):19–22, August 2004.

[19] B. Hoh, T. Yan, D. Ganesan, K. Tracton, T. Iwuchukwu, and J.-S. Lee. Trucentive: A game-theoretic incentive platform for trustworthy mobile crowdsourcing parking services. In *Proceedings of International IEEE Conference on Intelligent Transportation Systems*, ITSC12, pages 160–166, 2012.

[20] Jeff Howe. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Crown Publishing Group, New York, NY, USA, 1 edition, 2008.

[21] Huiqi Hu, Yudian Zheng, Zhifeng Bao, Guoliang Li, Jianhua Feng, and Reynold Cheng. Crowdsourced POI labelling: Location-aware result inference and task assignment. In *Proceedings of the 32nd IEEE International Conference on Data Engineering*, pages 61–72, 2016.

[22] Leyla Kazemi and Cyrus Shahabi. A privacy-aware framework for participatory sensing. *SIGKDD Explor. Newsl.*, 13(1):43–51, August 2011.

[23] Leyla Kazemi and Cyrus Shahabi. Geocrowd: Enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 189–198, 2012.

[24] Seon Ho Kim, Ying Lu, Giorgos Constantinou, Cyrus Shahabi, Guanfeng Wang, and Roger Zimmermann. Mediaq: Mobile multimedia management system. In *Proceedings of the 5th ACM Multimedia Systems Conference*, MMSys '14, pages 224–235, 2014.

[25] Irwin King, Jiexing Li, and Kam Tong Chan. A brief survey of computational approaches in social computing. In *Proceedings of the 2009 International Joint Conference on Neural Networks*, IJCNN'09, pages 2699–2706, 2009.

[26] Qiang Liu, Jian Peng, and Alexander Ihler. Variational inference for crowdsourcing. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 692–700, 2012.

[27] Tie Luo and Chen-Khong Tham. Fairness and social welfare in incentivizing participatory sensing. In *Proceedings of IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks*, SECON '12, pages 425–433, 2012.

[28] Y. Yilmaz M. Bulut and M. Demirbas. Crowdsourcing location-based queries. In *Proceedings of Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Percom'011, pages 513–518, 2011.

[29] Winter Mason and Duncan J. Watts. Financial incentives and the "performance of crowds". In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 77–85, 2009.

[30] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 323–336, 2008.

[31] Mohamed Musthag, Andrew Raij, Deepak Ganesan, Santosh Kumar, and Saul Shiffman. Exploring micro-incentive strategies for participant compensation in high-burden studies. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 435–444, 2011.

[32] Alexander J. Quinn and Benjamin B. Bederson. Human computation: A survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1403–1412, 2011.

[33] Christoph Schlieder, Peter Kiefer, and Sebastian Matyas. Geogames: Designing location-based games from classic board games. *IEEE Intelligent Systems*, 21(5):40–46, September 2006.

[34] Shaoxu Song and Lei Chen. Differential dependencies: Reasoning and discovery. *ACM Trans. Database Syst.*, 36(3):16:1–16:41, August 2011.

[35] Shaoxu Song, Lei Chen, and Hong Cheng. Parameter-free determination of distance thresholds for metric distance constraints. In *Proceedings of the 28th IEEE International Conference on Data Engineering*, ICDE '12, pages 846–857, 2012.

[36] Shaoxu Song, Lei Chen, and Philip S. Yu. On data dependencies in dataspaces. In *Proceedings of the 27th IEEE International Conference on Data Engineering*, ICDE '11, pages 470–481, 2011.

[37] Shaoxu Song, Lei Chen, and Philip S. Yu. Comparable dependencies over heterogeneous data. *The VLDB Journal*, 22(2):253–274, April 2013.

[38] R. Laubacher T. W. Malone and C. Dellarocas. Harnessing crowds: mapping the genome of collective intelligence. *Technical Report, MIT Sloan Research Paper*, 4732(09), 2009.

[39] Yuandong Tian and Jun Zhu. Learning from crowds in the presence of schools of thought. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 226–234, 2012.

[40] Hien To, Gabriel Ghinita, and Cyrus Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *PVLDB*, 7(10):919–930, 2014.

[41] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *Proceedings of the 32nd IEEE International Conference on Data Engineering*, ICDE'16, pages 49–60, 2016.

[42] Jing Wang, Siamak Faridani, and P Ipeirotis. Estimating the completion time of crowdsourced tasks using survival analysis models. *CSDM 2011*.

[43] Lirong Xia and Vincent Conitzer. Determining possible and necessary winners under common voting rules given partial orders. In *AAAI*, 2008.

[44] Liu Zheng and Lei Chen. Mutual benefit aware task assignment in a bipartite labor market. In *Proceedings of the 32nd IEEE International Conference on Data Engineering*, ICDE'16, pages 73–84, 2016.

[45] Xiangmin Zhou, Lei Chen, and Xiaofang Zhou. Structure tensor series-based large scale near-duplicate video retrieval. *IEEE Trans. Multimedia*, 14(4):1220–1233, 2012.

Optimizing Open-Ended Crowdsourcing: The Next Frontier in Crowdsourced Data Management

Aditya Parameswaran[†], Akash Das Sarma[⋆], Vipul Venkataraman[†]
[†]University of Illinois [⋆]Stanford University

Abstract

Crowdsourcing is the primary means to generate training data at scale*, and when combined with sophisticated machine learning algorithms, crowdsourcing is an enabler for a variety of emergent automated applications impacting all spheres of our lives. This paper surveys the emerging field of formally reasoning about and optimizing* open-ended crowdsourcing*, a* popular and crucially important, but severely understudied *class of crowdsourcing—the next frontier in crowdsourced data management. The underlying challenges include distilling the right answer when none of the workers agree with each other, teasing apart the various perspectives adopted by workers when answering tasks, and effectively selecting between the many open-ended operators appropriate for a problem. We describe the approaches that we've found to be effective for open-ended crowdsourcing, drawing from our experiences in this space.*

1 Introduction

We are on the cusp of a new data-enabled era, where machine learning algorithms, trained on large volumes of labeled training data, are enabling increasing automation in our daily lives—from driving, robotics, and manufacturing, to surveillance, medicine, and science; a recent New York Times article calls this *"a transformation that many believe will have a payoff on the scale of the personal computing industry or the commercial internet"* [1]. Although considerable effort has gone into the development of machine learning algorithms for these applications, the generation of labeled training datasets, done at scale using *crowdsourcing* [2]— while equally important [3, 4]—is often overlooked. Recent work has demonstrated that *optimizing crowdsourcing* can often yield *orders of magnitude more high-quality labeled data at the same cost*, spurring the development of increasingly sophisticated machine learning algorithms, and providing immediate benefits via substantial increases in accuracy for existing algorithms.

From Boolean Crowdsourcing to Open-Ended Crowdsourcing. However, most of the past research on optimizing crowdsourcing has focused on what we call *boolean crowdsourcing*, e.g., [5–9, 11, 47], where human involvement can be abstracted as tasks or operators whose answers come from a small, finite domain, e.g., evaluating a predicate, comparing a pair of items, or rating an item on a scale from 1–5. In the former two cases, the domain of possible answers is boolean, while in the last case, the domain is finite and has cardinality five. Boolean crowdsourcing operators have natural analogs in computer operators, and are easier to reason about and develop algorithms with. This research has largely ignored *open-ended crowdsourcing*, where the answers to

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

the tasks have no similar restriction. Anecdotal evidence indicates that open-ended crowdsourcing tasks are at least as popular as traditional boolean crowdsourcing tasks on crowdsourcing marketplaces. As one data point, an analysis of Mechanical Turk's log data [12] reported that *content creation*—including open-ended tasks like transcription—was by far the single largest category of tasks in the period from 2009 to 2014. As another example, a survey of industry users of crowdsourcing reports similar findings [2].

Examples of Open-Ended Crowdsourcing. We now describe some canonical open-ended crowdsourcing problems that we use as examples for this paper. One example is *transcription*: the goal of transcription is to transform a piece of audio or video to text. This could be done, for example, using unit tasks where we provide workers a portion of the audio or video, along with a text box where they can type a sequence of words to match the contents of the portion. Another example is *clustering*: the goal of clustering is to subdivide a collection of items (images, pieces of text) into clusters. This could be done, for example, using unit tasks where workers place items into an arbitrary number of buckets determined by them. Yet another example is *detection*: the goal of detection is to identify where in an image an object is located. This could be done, for example, using unit tasks where workers draw a bounding rectangle or polygon around the location of the object of interest.

Open-Ended Crowdsourcing: A Pressing Need. Beyond the popularity, there are several reasons why open-ended crowdsourcing is crucially important. First, some tasks are *near-impossible with just boolean crowdsourcing*. For example, using boolean crowdsourcing to locate the position of an object in an image is near-impossible. If we use a task like: "is the object in this portion of the image (yes/no)", we may be able to get close to the actual location of the object by repeatedly using this task on various portions of the image, but getting an accurate bounding box around the object may require hundreds or thousands of such tasks. On the other hand, asking workers to simply draw a bounding box is a lot more effective in terms of time, cost, and accuracy. Second, open-ended crowdsourcing lets us *get more fine-grained data*, since workers provide answers from a potentially unbounded set. If we were to use an entropy argument, the number of bits provided by workers for an open-ended question or task is considerably larger than that provided for a boolean question. Third, recent computer vision and text processing papers argue that *fine-grained training data is essential* for developing sophisticated machine learning models [13–16]. Specifically, our best hope for improving the accuracy of present machine learning models is by using training data that reveals more information about how humans think, as opposed to training data that is more akin to how computers operate (i.e., via boolean operations).

Despite these compelling reasons, research on optimizing open-ended crowdsourcing is still in its infancy. Open-ended crowdsourcing is presently *leveraged in an unoptimized fashion, with resources wasted and inaccurate data collected, or even worse, not at all,* leading to severe impediments to machine learning.

Challenges. The reason why open-ended crowdsourcing is leveraged in an unoptimized fashion is that optimizing open-ended crowdsourcing is substantially harder than optimizing boolean crowdsourcing:

1. Hard to aggregate. Due to the many possible answers to open-ended tasks, distilling the 'right' answer from this set is non-trivial. For example, when drawing a box around an object in an image, or transcribing audio into text, no two workers will provide the same box or transcription. This is because the number of possible answers is very large, and there are many ways of making mistakes. In boolean crowdsourcing, we could simply resort to the majority opinion, but those techniques do not apply, especially when all of the answers are different from each other.

2. Sparsity of quality measurements. In trying to characterize the error rates of workers, due to the large number of possible answers, it is hard to get reliable estimates of the probability that a worker provides an answer a, when the true answer is b. In order to estimate these probabilities, it would take a lot of tasks to be issued on crowdsourcing marketplaces to ensure adequate coverage (and therefore accurate estimates) for every (a, b) pair.

3. Many right answers. To further complicate matters, open-ended tasks often have many 'right' answers, due to different underlying perspectives or beliefs, making it challenging to distinguish between the case when a worker is making mistakes, or the case when the worker is simply adopting a different perspective. For instance,

when clustering items—say a collection of images of everyday objects, workers may use different criteria—size, color, geometric shape—to cluster the items, while also inadvertently introducing errors.

4. Multiple scales. While boolean crowdsourcing typically operates on items (images, text) as a whole, open-ended crowdsourcing can additionally operate on portions of items. For example, when counting objects in an image, we may ask workers to count within portions of the image for less error-prone counts. There are an unbounded number of such portions that can be counted, making it hard to pick between them when selecting tasks to be assigned to workers.

5. Many open-ended operators. Unlike boolean crowdsourcing which is limited to a small number of operator types, there is a wide variety of open-ended operators, even for the same problem (including boolean ones as a subset). For example, for detecting where an object is present in an image, workers could draw a box around an object, fix a box, or compare boxes. The large number of alternatives makes it hard to design algorithms.

In short, these issues (1–3) lead to an increased complexity in reasoning about the underlying algorithms, and (4–5) an overwhelming number of design choices when it comes to designing algorithms.

This Paper. While there has been a large body of work on open-ended crowdsourcing, primarily from the HCI (Human-Computer Interaction) community, most of this work has been on creatively using open-ended crowdsourcing operators in workflows as opposed to understanding how to model and reason about them, and develop optimal algorithms. While there has been a deep, interesting, and important body of work from the database community (and to a certain extent from the AI and machine learning communities) on optimizing boolean crowdsourcing, our hope is to bring open-ended crowdsourcing the same kind of attention, especially given its importance as articulated above.

Therefore, in this paper, we aim to outline an emerging body of work in optimizing open-ended crowdsourcing from us and from other groups by developing a set of *design principles* that we have found to be effective for algorithm development, and by describing how these design principles were applied for a few papers that we have been working on in this space. Note that our survey of the emergent work on open-ended crowdsourcing will necessarily be biased by our own work that we're most familiar with; this is not to indicate that the other work is not as important or as interesting, but merely indicates our lack of familiarity with them. We will attempt to categorize all of the open-ended crowdsourcing work from the database community that we are aware of, along with work from other communities in Section 4.

2 Design Principles for Open-Ended Crowdsourcing

We now describe some approaches that we've found to be successful in dealing with the increased modeling complexity (issues 1–3 above) and increased number of design choices (issues 4–5 above), followed by a solution scaffold or recipe for open-ended crowdsourcing.

2.1 Dealing with the Modeling Complexity

The challenges in modeling worker performance stems from the fact that there are far too many possible answers that workers can provide even for a single question or task. This means that we do not have a clear mechanism to aggregate worker answers, and nor do we have the ability to estimate error rates of the form $\Pr[\text{worker answers } a|\text{true answer is } b]$ for all (a, b) pairs. We have identified two ways of dealing with this modeling complexity, both of which essentially allow us to *transform* the worker answer into one or more boolean crowdsourcing answers, following which we can apply standard techniques from boolean crowdsourcing.

The first approach is to *project the answer down* to a finite set of choices. For example, if the worker provides an answer that is any rational number in a range, we can project this answer down to a finite set of integers; yet another way is to project it down into a binary choice: $\leq a$ or $> a$. Note that this approach is wasteful in that we lose some of the fine-grained information that workers are providing, and begs the question of why we

didn't simply ask the boolean crowdsourcing question in the first place. Nevertheless, this simple approach is commonly used: we provide an example of this approach in Section 3.1.

The second approach is to *decompose the answer* down to answers to a collection of boolean crowdsourcing questions. As an example, if a worker is providing a transcription to a piece of audio, instead of treating the entire transcription as one open-ended crowdsourcing task, we can break it down into the sequence of individual words, each of which can be considered a response or a non-response to a word transcription task, which is much more manageable. By doing so, we can now model and reason about workers making mistakes at the level of words, rather than complete transcriptions. In transcription, at least, an additional challenge remains, which is to identify which words across workers were meant to be provided as output for the same portion of the audio piece. As another example, if a worker is providing a bounding box as an answer to a detection task, instead of treating the bounding box as a whole, we can decompose it down into boolean crowdsourcing answers for individual pixels: where if a pixel is part of the box drawn by a worker, the pixel gets a "yes" answer for the corresponding boolean crowdsourcing question, while it gets a "no" answer if it is not. This approach has the downside that the answers to the "pixel-level" boolean crowdsourcing questions are in fact not independent—if the answer to a specific pixel is "yes", then it is more likely than the answer to a neighboring pixel is "yes" rather than no. By decomposing the open-ended crowdsourcing task to boolean crowdsourcing ones, we lose this information.

We also employ a third approach which is fundamentally different from the previous two. This last way of dealing with open-ended crowdsourcing tasks is a bit more ad-hoc, and problem dependent, but does not require us to discard any information. Here, we operate on the answers provided to the open-ended crowdsourcing tasks directly. Consider the answers to a single open-ended crowdsourcing task. We can represent each of these task answers as nodes in a graph, and connect nodes that are similar to each other (on some similarity metric, such as overlap) with an edge annotated by the degree of similarity. Once this graph of answers is constructed, we can apply standard graph clustering algorithms to identify various view-points among the open-ended task answers: the largest cluster or clique may represent the "consensus" answer. What can be done with these clusters is dependent on the problem. We provide an example of this approach in Section 3.2.

2.2 Dealing with the Increased Design Choices

As described previously, open-ended crowdsourcing brings with it a considerable increase in the number of alternative tasks that can be issued at each step. The increase is due to the large number of open-ended crowdsourcing task types available, and also because these tasks can be applied to portions of items and not just the items directly. We now describe our approaches to deal with these increased design choices.

Our first approach is one that has been applied in the past for boolean crowdsourcing, which is to estimate the *information gain* for issuing a specific task: here, an additional challenge is that the information gain may be hard to estimate because we do not have a good model to reason about worker performance. Nevertheless, we may be able to use proxies for information gain, e.g., prioritizing items or portions of items for which we have fewer answers than others, or more ambiguity, perhaps measured by projecting or decomposing the answer down to boolean tasks (as described in the previous section).

The second approach is to start at a coarser granularity and then drill-down only when needed. For instance, in transcription—we can have workers first transcribe the entire audio piece, and then have additional workers transcribe the portions that are were found to be ambiguous. We will describe another example of this approach in Section 3.1.

A last approach is to incorporate information from primitive machine learning algorithms to "direct" attention to specific items or portions of them. For example, if we know for certain that an object does not lie in a given portion of the image, we can remove those portions when providing workers the open-ended task where they are asked to draw a bounding box about the object of interest. Once again, we describe how this approach is used for counting in Section 3.1.

2.3 Solution Scaffold

We have developed an *open-ended crowdsourcing problem-solving approach* drawing on our mechanisms to deal with additional modeling and design choice complexity. While the specific instantiation may differ across problems, the overall principles still apply, and the insights transfer across. For each problem, we apply **MARQED**, short for ***Model-Aggregate-Reason-Quantify-Estimate-DrillDown***: the first three (MAR) are tailored to managing modeling complexity, while the last three (QED) are tailored to managing design choice complexity.

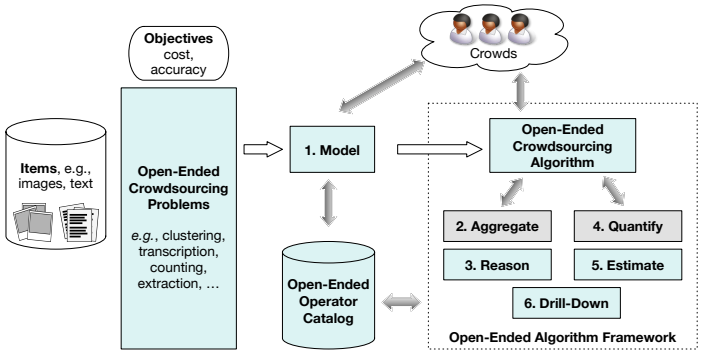


Figure 1: Algorithm Flow: Boxes in blue did not exist in boolean crowdsourcing, while boxes in gray are substantially different

In particular, **MARQED** stands for the following: (1) ***Model*** the performance of workers on open-ended operators; (2) Develop methods to ***Aggregate*** across their responses to identify one or more 'right answers'; (3) Identify techniques to ***Reason*** about whether workers are generating their answers based on the same or different underlying perspective; (4) Develop procedures to ***Quantify*** the information gain of different open-ended operators, and the same operator operating on different items; (5) Design schemes to incorporate prior ***Estimates*** from automated algorithms to reduce costs; and (6) Develop ***Drill-Down*** techniques on the items to enable workers to examine an item more closely. Then, we ***design the open-ended crowdsourcing algorithm*** that can leverage (1–6) (if available)—see Figure 1; boxes in blue did not exist in boolean crowdsourcing, while boxes in gray are substantially different. Note that it is not necessary to develop solutions for all of (2–6) before we can reap considerable benefits in practice. Next, we describe how we applied **MARQED** in practice.

3 Example Problems in Open-Ended Crowdsourcing

In this section, we describe our results for two problems that we believe are representative of the challenges in open-ended crowdsourcing, along with our solution approaches, in addition to other problems in this space.

3.1 Counting

The goal of *counting* is to estimate the number of objects of a given type in an image at low cost; it is a basic computer vision primitive, with applications in security, medicine, and biology. Counting is a hard problem due to occlusion—the partial obscuring of objects behind other objects, with state-of-the-art automated techniques having accuracies less than 50% [17, 18].

In our paper [19], we develop cost-effective techniques to use crowds to accurately count the number of objects in images from two completely different domains—cell colonies in microscope photos, and people in Flickr photos. We now describe the components of our solution, drawing parallels to the **MARQED** methodology.

Model. Our open-ended operator is simple—we display an image or a portion of an image, and ask workers to provide the count of the number of objects in that portion. We find that workers do not make mistakes in counting when the number of objects in the image is less than a certain number k (20 in our experiments); after that, workers start introducing errors, with the errors growing superlinearly as the number of objects increases. This may be because workers are not able to keep track of the objects they have already counted, or because they get fatigued beyond a certain point and start introducing errors. Using this insight, we model worker error by *projecting down* worker answers to boolean ones: if the answer is $< k$, then it is assumed to be correct; if the

answer is $\geq k$, then it is assumed to be incorrect i.e., given a worker provides a count $\geq k$, the only information we can deduce is that it is $\geq k$, but no additional information can be inferred.

While this simple model provides reasonable results, we are indeed wasting information by ignoring worker answers if they are $\geq k$. Using a more fine-grained error model along with maximum-likelihood estimation can help identify a current best estimate for an image or a portion of an image, allowing us to "skip ahead" to the portions that need more attention. That said, this fine-grained error model requires more expensive training data to estimate accurately.

Drill-down. Based on the simple error model, we can already develop a strategy for counting objects in images, by repeatedly splitting images and *drilling-down*. We model this process as a *segmentation-tree*, where the root node represents the original, complete image, and children of any node represent the segments obtained by splitting the parent image (using some splitting scheme, horizontal or vertical). Figure 2 shows one such example segmentation tree where the original image, V_0, is split into segments $\{V_1, V_2\}$, which are respectively split into $\{V_3, V_4\}$ and $\{V_5, V_6, V_7\}$.

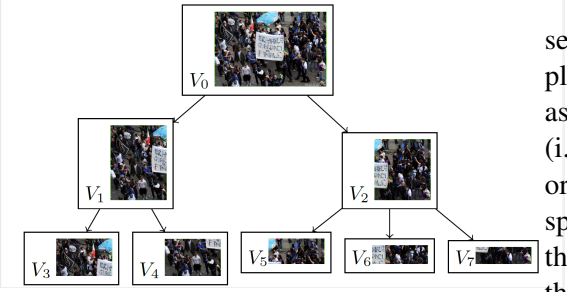


Figure 2: Segmentation Tree

We refer to any set of mutually exclusive nodes (i.e. image segments) that when put together reconstruct the original complete image as a *frontier* of the segmentation-tree. We start by asking workers to count the number of images in the root node (i.e. the complete image). If they reply with a count greater than or equal to k, then we traverse down the segmentation tree by splitting the image and asking workers to provide the count for the children segments. If the count on all segments is smaller than k, then we are done. If not, we again split and repeat this process for every segment that still has a count $\geq k$, until we reach a frontier of image segments such that every one of them has a smaller count than k. This simple strategy has optimal *competitive ratio* for any given segmentation tree, and also returns counts with reasonable accuracies of 93% when counting people on a standard dataset [20]. This approach does come with one challenge: objects often span multiple sibling segments in the segmentation tree, and therefore have a danger of being double-counted. In our paper, we take the easy-way-out by asking workers to only count an object if more than half of it appears in the image segment. Further improvements may be possible.

Reason and Aggregate. Using our simple error model, reasoning about perspectives, and aggregation is not necessary—workers are expected to agree and provide correct answers as long as the true count is $< k$—thus we can simply ask one worker to count each image or portion of the image, and additional answers are not necessary. In practice, however, we find that sometimes workers may still introduce errors: to handle this, in our paper, we take three worker answers per question and take the median—this simple aggregation scheme is sufficient to resolve worker differences and obtain high accuracy count estimates. Finally, we sum up all the aggregated counts from the different constituent image segments to obtain the count for the original, whole image. Changes to the worker error model, for instance, using a fine-grained probabilistic model for maximum-likelihood count estimation, will open the road to more interesting aggregation challenges.

Quantify. While we did not implement other operators for this problem, there are a number of interesting alternatives that yield more information, at the expense of a little extra worker effort. For instance, we could ask workers to tag objects that they have already counted, say with a dot, to help eliminate double-counting as well as avoid missing objects. Tagged objects could serve as references for other workers, who could then mark additional objects missed by previous workers, or eliminate redundant instances.

Estimate. Even if the number of objects is in the hundreds, our algorithm on the segmentation tree may end up asking several "useless" questions at the higher levels all with count $\geq k$, while the "useful" questions (whose answers are actually used to compute the final count at the root) are at the frontier of the tree where the count is just k. However, it may be possible to use feedback from a primitive automatic segmentation algorithm to craft

a segmentation tree where there are no useless levels, and where objects do not span multiple sibling segments. Consider the problem of counting cells in biological images. Even though automated counting may be hard due to occlusion, we can partition the image into non-overlapping portions using the watershed algorithm [21] and learn prior counts for each partition using an SVM [22]. Note that these prior counts may be much smaller than expected (due to occlusion), but it suffices as a starting point.

Given these partitions and prior counts, we can construct a segmentation tree that groups multiple contiguous partitions together until they hit up to k objects each (based on the prior counts, which may be underestimating). This allows us to construct a segmentation tree where all levels are "useful" given our prior information. Since merging partitions together optimally is NP-HARD via a reduction from planar partitioning [23], we employ other heuristic techniques in our paper, such as first-fit [24]. We then traverse the tree asking workers as before. By using the prior machine-learned estimates in this fashion, we are able to skip several "useless" questions providing a 2x reduction in cost. It should be noted that the images of cell colonies are much more amenable to automated prior estimation techniques. It is an open challenge to explore whether similar techniques could be applied to other kinds of objects, where it is a lot harder to get accurate prior counts.

3.2 Clustering

Given a collection of items (e.g., images, documents), the goal of clustering, is to organize them into coherent groups. Collections of images or documents are commonplace; organizing them is essential before one can understand the themes in the collection, or improve search or browsing. Clustering embodies all the challenges faced by open-ended crowdsourcing: workers can have different perspectives leading to multiple "right" answers making worker responses hard to aggregate. The space of possible responses is also extremely large, since workers operate on multiple items simultaneously. The open-endedness of the problem also means that there are a number of different interfaces and operators that could conceivably be used. We shall now see how applying the **MARQED** approach allows us to reason about this challenging open problem in a principled fashion.

Model. Prior work has considered crowdsourced clustering, limited to the boolean operator where pairs of items are compared [25, 26]—as a result, crowd workers do not have any context to compare items. Also, the eventual clusterings end up having "mixed" perspectives, resulting in low accuracies. Instead we used a basic open-ended clustering operator, where a set of items are provided to workers, and they are asked to group them into an arbitrary number of disjoint clusters. Using this operator, we had multiple workers cluster a stylized image dataset with each image containing a shape with different sizes and colors, as a running example. First, we introduce the notion of *concept hierarchies* to capture the notion of worker cluster perspectives. One concept hierarchy for the concept of shapes, could be to have {*All Shapes*} divided into {*Quadrilaterals*, *non-Quadrilaterals*}, the latter of which is subdivided into {*Triangles*, *Circular Shapes*}. For any given dataset, the concept hierarchy need not be unique. For instance, another hierarchy would have {*All Shapes*} divided into {*Circular Shapes*, *Straight-Edged Shapes*}. When clustering, each worker answer can be seen to draw from one or more inherent concept hierarchies.

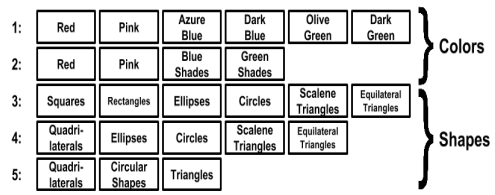


Figure 3: Examples of real worker clusterings

Figure 3 shows examples of real worker clusterings on our dataset. While workers 1 and 2 clustered based on color alone, workers 3, 4 and 5 clustered based on shape. We focus on the latter for the time being. One conceptual hierarchy, C, "consistent" with workers 3, 4, and 5 is {*All Shapes*} divided into {*Quadrilaterals*, *Triangles*, *Circular Shapes*}, which are respectively subdivided into {*Squares*, *Rectangles*}, {*Scalene Triangles*, *Equilateral Triangles*}, and {*Circles*, *Ellipses*}. We additionally introduce the notion of *frontiers* on a given concept hierarchy to capture the notion of granularities: a frontier in a hierarchy is a set of nodes that do not have any ancestor-descendent relationship between them, and together cover all paths to the leaves. Each frontier corresponds to one valid granularity of clustering consistent

with the concept hierarchy. Representing the concept hierarchy, C, as a tree, we have worker 3 operating at the leaf nodes, or at the finest granularity of the tree, corresponding to the frontier {*Squares*, *Rectangles*, *Scalene Triangles*, *Equilateral Triangles*, *Circles*, *Ellipses*}. Similarly, worker 5 is operating at a depth of one in the tree, which is the coarsest non-trivial granularity and corresponds to the frontier {*Quadrilaterals*, *Triangles*, *Circular Shapes*}.

Reasoning and Aggregation. From our experiments on the stylized dataset, we make the following observations: (a) Workers cluster using different *perspectives*, e.g., some workers clustered using shape, and others clustered using color. That said, there was a dominant, popular clustering perspective, in this case, shape. (b) Even within a perspective, workers cluster at various *granularities*, e.g., some workers clustered shapes into rectangles and non-rectangles, while others broke up non-rectangles into fine-grained clusters. (c) Sometimes, there are *confusing items* that end up being placed in different clusters by different workers even if they agree on the perspective and the granularity.

To reason about workers' perspectives, we develop a notion of *consistency*: two worker clusterings (i.e., a set of clusters formed by each worker) are consistent if for any pair of clusters C, C', one from each worker, either $C \subseteq C', C' \subseteq C$, or $C \cap C' = \emptyset$, i.e., each cluster from one worker generalizes, specializes, or does not overlap with another worker's clusters. This definition allows consistent workers to cluster at different granularities.

Given the notion of consistency, we can now directly operate on worker responses to identify whether there are any "consensus" clusterings, i.e., consensus concept hierarchies (perspectives), and frontiers within them (granularities), that emerge. This allows us to have a starting point to cluster the rest of the items. To do this, we generate a *clustering graph*, with one node per worker, with consistent pairs having an edge between them.

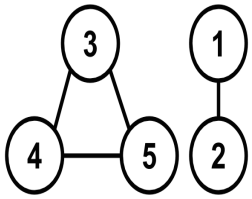


Figure 4: Clustering graph

Figure 4 shows the clustering graph corresponding to the workers from Figure 3. Workers 1 and 2 both clustered based on color alone and were consistent with each other, while workers 3, 4, and 5 clustered by shape alone with no inconsistencies among themselves. We can show that under multinomial worker perspective selection models, the maximum likelihood worker perspective is the MAX-CLIQUE in the graph. Despite MAX-CLIQUE being NP-HARD, the clustering graph is often not large, making the problem tractable. In our paper, we additionally describe how we can incorporate worker error models, which introduce additional complexity to the problem.

Quantify and Drill-Down. So far, we have described how to use a single open-ended clustering operator and aggregate responses from it for a small number of items. However, if we have a large number of items, workers might not be able to cluster all of them in one go. This suggests the need for drilling-down, or splitting the set of items, and asking workers to cluster the resulting subsets. This raises the challenge of aggregating partial clusterings. For this purpose, we maintain a *kernel* of items across clustering tasks, akin to *pivots* in Polychronopoulos et al. [9], to be able to relate partial clusterings across each other, for aggregation. We design techniques to extend the current maximum likelihood hierarchy by merging worker responses on new items to the existing hierarchy—we leverage these techniques to design a merging algorithm which aggregates clusterings from separate subsets together to output a consensus hierarchy on the original, complete set of items.

We additionally incorporate a categorization-based operator, once the consensus clustering is identified, to provide additional cost benefits, by categorizing the remaining items into the discovered clusters [27]. Instead of having workers repeatedly cluster many items (implicitly a many-to-many comparison), they only need to identify which cluster or category is appropriate for one item at a time, with the clusters being fixed.

Overall, our techniques lead to up to $3\times$ better recall and $1.9\times$ better accuracy than boolean clustering schemes using pairwise judgments [25, 26] on their datasets, for the same crowdsourcing cost.

3.3 Other Problems

We've applied the **MARQED** approach to other open-ended crowdsourcing problems. We describe some of these problems briefly here, followed by work done by others.

Extraction. The goal of *extraction* is to use crowdsourcing to gather entities of a specific type [28]. We considered a broad space of open-ended operators for this problem, leveraging the attributes of the entity set [29]. For example, musicians in Chicago can be categorized as guitarists, drummers, and so on, and may play music of various types. By considering these attributes, we can ask workers to answer more *fine-grained* open-ended questions: e.g., provide a jazz drummer in Chicago. This allows us to target the questions at the attribute combinations where we lack entities. However, the number of possible open-ended questions increases exponentially in the attributes, making the problem challenging. We showed that picking the best questions is NP-HARD, and found that a *drill-down* based technique works well, where we ask generic questions first (e.g., provide a musician), and then drill down and ask more specific questions later (e.g., provide a drummer in Lincoln Park).

Searching. The goal of crowd-assisted search [30, 31], is to return relevant results from a corpus given a search query with embedded images, video, and/or text. Here, we once again found that the following open-ended problem solving aspects are helpful: (a) multiple open-ended operators; (b) starting at a coarser granularity and drilling down into more promising candidates; and (c) incorporating prior information – in this case from regular text search engines.

Batching. Boolean tasks are typically grouped into batches of 10s to 100s of tasks that are attempted by workers together—to reduce cost and effort—making it essentially one large open-ended task. However, these tasks are assumed to be answered independently, which is not the case. We developed a probabilistic model to reason about the answers incorporating two forms of judging: independent, where workers answer each question independently, or correlated, where workers implicitly rank the items and then pick the top-k to be positive, with the rest negative (i.e., the Plackett Luce model). We found that this model, led to substantial accuracy improvements over schemes that ignored these correlations [32].

In addition, there has been work by others on optimizing other open-ended crowdsourcing problems. We describe some of them below.

The goal of *transcription* is to transcribe a sequence of words into text; it is a very challenging problem, with automated techniques performing very poorly [33, 34]. By decomposing worker answers down into individual words, one can apply standard multiple sequence alignment algorithms from the bioinformatics literature to align worker answers and identify consensus answers, leading to substantial improvements [34]; other work tries combining crowd answers with automated techniques [35, 36]. Some prior work has also looked at the problem of *detection* — finding the location of objects in images, applying computer vision models and human input via a Markov Decision Process [37].

Other open-ended crowdsourcing work exists as well, for various purposes, including designing plans [38], rule mining [39], and pattern matching [40].

4 Related Work

The work on open-ended crowdsourcing is related to work in many areas.

Area 1: Optimized Boolean Crowdsourcing. There has been quite a bit of work on optimizing boolean crowdsourcing, targeting basic algorithms, including filtering [5,6,41], sorting [7], max [11,42,43], categorization [27], top-K [8, 9, 44], spatial crowdsourcing [45, 46], and entity resolution (ER) [47, 47–53].

Area 2: Crowdsourcing Systems and Toolkits. Many groups have been building crowdsourcing systems and toolkits to harness crowdsourcing in a "declarative" manner [54–57], as well as several domain-specific toolkits [30, 31, 58, 59]. All these systems and toolkits could benefit from the design of optimized algorithms as building blocks.

Area 3: Quality Estimation: A number of papers perform simultaneous worker quality estimation and most accurate answer estimation, typically using the EM algorithm, sometimes providing probabilistic or partial guarantees, and sometimes modeling difficulty, bias, and adversarial behavior, e.g., [60–67]. To our knowledge, there is no work on applying EM to open-ended crowdsourcing tasks.

Area 4: Decision Theory: Recent work has leveraged decision theory for improving cost and quality in simple workflows, typically using POMDPs (Partially Observable MDPs), to dynamically choose the best decision to make at any step, e.g.. [68, 69]. While some of this work could be applicable to some open-ended tasks, there are no optimality guarantees associated with any of these techniques.

5 Conclusion

Open ended crowdsourcing is not only challenging and interesting, but also necessary to meet the demands of the new generation of data-hungry applications. We hope that the next wave of crowdsourcing research from the database community will tackle more problems in this space, expanding the frontier of our understanding.

Acknowledgements: A. P. acknowledges support from grants IIS-1513407 and IIS-1633755 awarded by the National Science Foundation, grant 1U54GM114838 awarded by NIGMS and 3U54EB020406-02S1 awarded by NIBIB through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov), and funds from Adobe, Google, and the Siebel Energy Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies and organizations.

References

[1] "Artificial Intelligence Swarms Silicon Valley on Wings and Wheels," *http://www.nytimes.com/2016/07/18/technology/on-wheels-and-wings-artificial-intelligence-swarms-silicon-valley.html*.

[2] A. Marcus and A. Parameswaran, "Crowdsourced data management: Industry and academic perspectives," *Foundations and Trends in Databases Series*, 2016, http://data-people.cs.illinois.edu/crowd-book.html.

[3] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *Intelligent Systems, IEEE*, 2009.

[4] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, 2012.

[5] A. G. Parameswaran, et al., "Crowdscreen: algorithms for filtering data with humans," in *SIGMOD*, 2012.

[6] A. G. Parameswaran, S. Boyd, et al., "Optimal crowd-powered rating and filtering algorithms," in *VLDB*, 2014.

[7] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller, "Human-powered Sorts and Joins," *PVLDB*, 2011.

[8] A. D. Sarma, A. Parameswaran, H. Garcia-Molina, and A. Halevy, "Crowd-powered find algorithms," in *ICDE*, 2014.

[9] V. Polychronopoulos, L. de Alfaro, et al., "Human-powered top-k lists," in *WebDB*, 2013.

[10] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "CrowdER: Crowdsourcing Entity Resolution," *PVLDB*, 2012.

[11] S. Guo, A. Parameswaran, et al., "So who won? dynamic max discovery with the crowd," in *SIGMOD*, 2012.

[12] D. E. Difallah, et al., "The dynamics of micro-task crowdsourcing: The case of amazon mturk," in *WWW*, 2015.

[13] E. Pavlick, J. Bos, M. Nissim, C. Beller, B. Van, and D. C. Callison-burch, "Adding semantics to data-driven paraphrasing," in *In ACL*, Citeseer, 2015.

[14] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *EMNLP*, Association for Computational Linguistics, 2015.

[15] A. Vedaldi, S. Mahendran, et al., "Understanding objects in detail with fine-grained attributes," in *CVPR*, 2014.

[16] S. Bell, K. Bala, and N. Snavely, "Intrinsic images in the wild," *ACM Transactions on Graphics (TOG)*, 2014.

[17] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *CVPR, 2012*.

[18] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2014.

[19] A. D. Sarma, A. Jain, A. Nandi, A. Parameswaran, and J. Widom, "Surpassing humans and computers with jellybean: Crowd-vision-hybrid counting algorithms," in *HCOMP*, 2015.

[20] B. Plummer, L. Wang, C. Cervantes, J. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," *CoRR*, vol. abs/1505.04870, 2015.

[21] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," *OPTICAL ENGINEERING-NEW YORK-MARCEL DEKKER INCORPORATED-*, vol. 34, pp. 433–433, 1992.

[22] T. W. Nattkemper, H. Wersing, W. Schubert, and H. Ritter, "A neural network architecture for automatic segmentation of fluorescence micrographs," *Neurocomputing*, vol. 48, no. 1, pp. 357–367, 2002.

[23] M. Dyer and A. Frieze, "On the complexity of partitioning graphs into connected subgraphs," *Discrete Applied Mathematics*, vol. 10, no. 2, pp. 139–153, 1985.

[24] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: a survey," in *Approximation algorithms for NP-hard problems*, pp. 46–93, PWS Publishing Co., 1996.

[25] R. Gomes, P. Welinder, A. Krause, and P. Perona, "Crowdclustering," in *NIPS*, pp. 558–566, 2011.

[26] J. Yi, R. Jin, A. K. Jain, and S. Jain, "Crowdclustering with sparse pairwise labels: A matrix completion approach," in *In AAAI Workshop on Human Computation*, 2012.

[27] A. G. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom, "Human-assisted graph search: it's okay to ask questions," *PVLDB*, vol. 4, no. 5, pp. 267–278, 2011.

[28] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar, "Crowdsourced enumeration queries," in *ICDE*, 2013.

[29] T. Rekatsinas, A. Deshpande, and A. Parameswaran, "CrowdGather: Entity Extraction over Structured Domains," *ArXiv e-prints*, Feb. 2015.

[30] A. Parameswaran, M. H. Teh, H. Garcia-Molina, and J. Widom, "DataSift: An Expressive and Accurate Crowd-Powered Search Toolkit," in *HCOMP*, 2013.

[31] A. G. Parameswaran, M. H. Teh, H. Garcia-Molina, and J. Widom, "Datasift: a crowd-powered search toolkit," in *International Conference on Management of Data, SIGMOD*, 2014.

[32] H. Zhuang, A. G. Parameswaran, D. Roth, and J. Han, "Debiasing crowdsourced batches," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[33] D. Yu and L. Deng, *Automatic Speech Recognition*. Springer, 2012.

[34] W. Lasecki, C. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. Kushalnagar, and J. Bigham, "Real-time captioning by groups of non-experts," in *UIST*, 2012.

[35] J. D. Williams, I. D. Melamed, T. Alonso, B. Hollister, and J. Wilpon, "Crowd-sourcing for difficult transcription of speech," in *Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[36] R. Van Dalen, K. Knill, P. Tsiakoulis, and M. Gales, "Improving multiple-crowd-sourced transcriptions using a speech recogniser," in *ICASSP*, 2015.

[37] O. Russakovsky, L.-J. Li, and L. Fei-Fei, "Best of both worlds: human-machine collaboration for object annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2121–2131, 2015.

[38] I. Lotosh, T. Milo, and S. Novgorodov, "Crowdplanr: Planning made easy with crowd," in *ICDE*, 2013.

[39] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart, "Crowd mining," in *SIGMOD Conference*, 2013.

[40] G. Demartini, B. Trushkowsky, T. Kraska, and M. J. Franklin, "Crowdq: Crowdsourced query understanding," in *CIDR*, 2013.

[41] Y. Gao and A. G. Parameswaran, "Finish them!: Pricing algorithms for human computation," *PVLDB*, 2014.

[42] P. Venetis and H. Garcia-Molina, "Dynamic max algorithms in crowdsourcing environments," technical report, Stanford University, August 2012.

[43] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis, "Max algorithms in crowdsourcing environments," in *Proceedings of the 21st World Wide Web Conference*, 2012.

[44] S. B. Davidson, S. Khanna, T. Milo, and S. Roy, "Using the crowd for top-k and group-by queries," in *ICDT*, 2013.

[45] D. Deng, C. Shahabi, U. Demiryurek, and L. Zhu, "Task selection in spatial crowdsourcing from worker's perspective," *GeoInformatica*, vol. 20, no. 3, pp. 529–568, 2016.

[46] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Trans. Spatial Algorithms and Systems*, vol. 1, no. 1, p. 2, 2015.

[47] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking," in *WWW*, 2012.

[48] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye, "Katara: A data cleaning system powered by knowledge bases and crowdsourcing," in *SIGMOD*, 2015.

[49] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 229–240, ACM, 2013.

[50] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu, "Corleone: Hands-Off Crowdsourcing for Entity Matching," *SIGMOD 2014*, Mar. 2014.

[51] S. Wang, X. Xiao, and C.-H. Lee, "Crowd-based deduplication: An adaptive approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1263–1277, ACM, 2015.

[52] K. Bellare, S. Iyengar, A. Parameswaran, and V. Rastogi, "Active sampling for entity matching with guarantees," in *ACM Transactions on Knowledge Discovery from Databases (TKDD)*, 2013.

[53] K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi, "Active sampling for entity matching," in *KDD*, 2012.

[54] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: answering queries with crowdsourcing," in *SIGMOD Conference*, pp. 61–72, 2011.

[55] A. Marcus, et al., "Crowdsourced databases: Query processing with people," in *CIDR*, 2011.

[56] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom, "Deco: declarative crowdsourcing," in *CIKM*, pp. 1203–1212, 2012.

[57] H. Park, A. Parameswaran, and J. Widom, "Query processing over crowdsourced data," technical report, Stanford University, September 2012.

[58] A. Bozzon, M. Brambilla, and S. Ceri, "Answering search queries with crowdsearcher," in *WWW*, 2012.

[59] D. Deutch, et al., "Using markov chain monte carlo to play trivia," in *ICDE*, 2011.

[60] D. R. Karger, S. Oh, and D. Shah, "Budget-optimal task allocation for reliable crowdsourcing systems," in *CoRR*, vol. abs/1110.3564, 2011.

[61] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. R. Movellan, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *NIPS*, pp. 2035–2043, 2009.

[62] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi, "Aggregating crowdsourced binary ratings," in *WWW*, 2013.

[63] Y. Zhang, et al., "Spectral methods meet em: A provably optimal algorithm for crowdsourcing", *arXiv preprint*, 2014.

[64] A. Ramesh, A. Parameswaran, H. Garcia-Molina, and N. Polyzotis, "Identifying reliable workers swiftly," technical report, Stanford University, September 2012.

[65] M. Joglekar, H. Garcia-Molina, and A. Parameswaran, "Evaluating the crowd with confidence," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 686–694, ACM, 2013.

[66] M. Joglekar, H. Garcia-Molina, and A. Parameswaran, "Comprehensive and reliable crowd assessment algorithms," in *ICDE*, 2015.

[67] A. Das Sarma, A. Parameswaran, and J. Widom, "Towards globally optimal crowdsourcing quality management: The uniform worker setting," in *SIGMOD*, 2016.

[68] C. H. Lin, Mausam, and D. S. Weld, "Crowdsourcing control: Moving beyond multiple choice," in *UAI*, 2012.

[69] E. Kamar, et al., "Combining human and machine intelligence in large-scale crowdsourcing," in *AAMAS*, 2012.

Interactive Data Exploration via Machine Learning Models

Olga Papaemmanouil*, Yanlei Diao±, Kyriaki Dimitriadou*, Liping Peng±
* Brandeis University, ±University of Massachusetts, Amherst
olga@cs.brandeis.edu, yanlei@cs.umass.edu, kiki@cs.brandeis.edu, lppeng@cs.umass.edu

Abstract

This article provides an overview of our research on data exploration. Our work aims to facilitate interactive exploration tasks in many big data applications in the scientific, biomedical and healthcare domains. We argue for a shift towards learning-based exploration techniques that automatically steer the user towards interesting data areas based on relevance feedback on database samples, aiming to achieve the goal of identifying all database objects that match the user interest with high efficiency. Our research realizes machine learning theory in the new setting of interactive data exploration and develops new optimizations to support "automated" data exploration with high performance over large databases. In this paper, we discuss a suite of techniques that draw insights from machine learning algorithms to guide the exploration of a big data space and leverage the knowledge of exploration patterns to optimize query processing inside the database.

1 Introduction

Today data is being generated at an unprecedented rate. Every day large data sets are collected from sensors and scientific instruments that monitor our environment. For instance, LSST [17], a leading effort in astronomical surveys, is expected to store 55 petabytes of raw imagery ultimately and the database catalog containing descriptions of these objects and their observations is expected to approach 50 petabytes in size. Although data volumes and the user community of big data sets continue to grow, the human ability to comprehend data remains as limited as before. Hence, in the "Big Data" era we are faced with an increasing gap between the growth of data and the limited human ability to comprehend the data. Our work on data exploration aims to deliver new software tools to bridge this increasing gap.

Database management systems (DBMSs) have been long used as standard tools to store data and query it to obtain valuable information. However, traditional DBMSs are suited for applications in which the structure and the content of the database, as well as the questions (queries) to be asked are already well understood by the user. Unfortunately, these fundamental assumptions made by the DBMSs are becoming less true as the volume and diversity of data grow. First, the structure and content of the database are hard to understand, even for database experts. Second, finding the right question to ask is a long running complex task by itself, often requiring a great deal of experimentation with queries, backtracking on the basis of query results and revision of results at various points in the process.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Therefore, we argue that fundamental new models and tools are needed to increase the usability of DBMSs. Towards this direction, our work proposes "interactive data exploration" as a new service of a database management system, and it offers a suite of new algorithms, methods and optimizations to support this service for a broad user community across science, healthcare, and business. Our approach leverages machine learning techniques and offers new data management optimization algorithms to provide effective data exploration results as well as high interactive performance over databases of big sizes.

2 Interactive Data Exploration: Overview & Challenges

To support the task of interactive data exploration, we introduce a new approach for system-aided exploration of big data spaces that relies on *automatically learning* user interests and infers "classification" models that retrieve data relevant to the user interests. To achieve this, we rely on an interactive learning approach that iteratively requests user feedback on strategically collected data samples. In a nutshell, the user engages in a "conversation" with the system by characterizing a set of data samples as relevant or irrelevant to his interest. The user feedback is incrementally incorporated into the system and used to gradually improve the effectiveness of the query steering process, that is, to lead the user towards interesting data areas and eventually generate a classification model that precisely characterizes the set of data matching the user interest.

This interactive query steering process is depicted in Figure 1. Initially, the user is presented with a sample set selected to capture the diversity of the overall data exploration space. The iterative query steering process starts when the user provides feedback on the relevance of these samples. Labeled samples are used as the training set of a classification model that characterizes the user interest, i.e., predicting the data objects relevant to the user based on the feedback collected so far (*Learning*). Subsequent iterations aim to refine the characterization of the user interest by exploring further the data space: it identifies promising data areas to be sampled further (*Space Exploration*) and it retrieves the next sample set to show to the user. To achieve that, we leverage current knowledge of the user interest as defined by the user model. New samples and the user feedback on them are incorporated with the already labeled samples and a new user model is built in the next iteration. The above steps are executed iteratively aiming to converge to a model that captures the user interest within acceptable accuracy. The steering process is terminated when the classifier's accuracy reaches a system-defined threshold (i.e., on the number of labeled objects or convergence rate) or the user terminates the process explicitly.

In our framework, users are asked for feedback on data objects. In the back-end, each object is mapped to a set of features collected through domain-specific feature extraction tools. Data objects in our target applications include many layers of features. We make a concrete distinction between the feature space one uses for visualizing data objects for annotation and the feature space used for exploration. Specifically, while the front-end should visualize high-level features which humans can understand and interact with effectively (i.e., pictures, maps, summarized values), the back-end exploration is performed on an extended set of dimensions that include also lower level features. This distinction allows users to review data objects while concealing the detailed feature (attribute) set of the underlying database. At the same time, it allows for more effective exploration as the system learns user interests based on a set of features that is wider from the one humans can perceive. Hence, our learning-based approach can provide more insightful descriptions of the user interests than the one humans could generate manually.

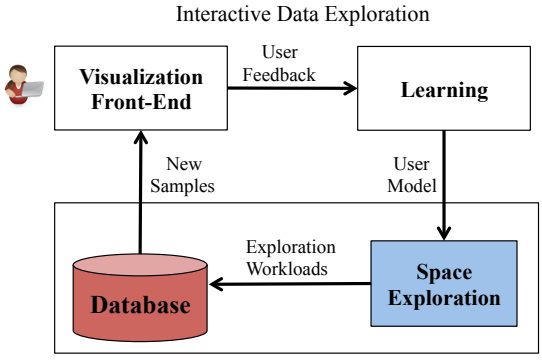


Figure 1: Interactive Data Exploration.

Research Challenges Towards realizing our vision for a learning-based data exploration system, we have identified two main research challenges:

1. *Fast Convergence*: A core research issue is to decide our exploration areas and how to sample them for objects to present to the user for further feedback. The challenge is to make such *exploration decisions* in a way that the user model converges to the true model with the minimum number of labeled samples.

2. *Interactive Performance*: The learning and space exploration steps of each iteration must offer interactive performance, e.g., complete within seconds, as the user may be waiting online. This challenge becomes even more apparent in large databases.

Existing DBMSs are not designed to support multi-step exploration tasks with interactive performance over large databases. At the same time, existing machine learning algorithms cannot solve our data exploration problem either. Classification algorithms (e.g., [3]) can build the user model but do not deal with the question of *which* data samples to show to the user and cannot be used to tackle the challenge for fast convergence. Active learning solutions [24] aim to identify the most promising data samples to label in order to maximize the learning outcome while minimizing the user effort. However, our recent results [9] revealed that those algorithms are not designed for interactive data exploration over large databases as they examine and rank *all* database objects before they identify the best sample to show to the user. Therefore, they cannot offer fast convergence nor interactive performance on big data sets.

In a nutshell, existing machine learning algorithms and database systems do not address the two main challenges of interactive data exploration. Hence, our research advocates a close synergy between machine learning and database algorithms and offers methods and tools that address the above research challenges.

3 Learning-based Exploration Strategies

We next discuss two exploration strategies that target diverse types of user interests. One uses decision tree classifiers for predicting linear interest patterns and the second one uses SVM (Support Vector Machine) models for capturing non-linear interest patterns.

3.1 Decision trees: learning linear patterns

Decision-tree classifiers can be very effective in discovering linear patterns of user interests, i.e., interests captured by conjunctive and/or disjunctive of linear (range) predicates. We refer to such interests as *relevant areas*. We designed an exploration framework [7–9] that leverages decision tree learning to efficiently produce highly accurate user models. Decision trees are easy-to-interpret prediction models that describe the features characterizing our relevant objects. Hence, the classification rules can be easily translated to simple boolean expressions and therefore to query expressions that retrieve all objects predicted to be relevant to the user.

The exploration is performed in a d-dimensional space where each tuple represents a d-dimensional object and the relevant areas are hyper-rectangles with up to d dimensions. The exploration space may include attributes both relevant and irrelevant to the final expression that captures the true user interest. The steering process starts when the user provides feedback on the relevance of the first set of retrieved samples. We assume a binary feedback model where the user indicates whether a data object is relevant or not to her. The labeled samples are used to train a decision tree which may use any subset of the d attributes of the exploration space to characterize user interests. Each iteration refines the characterization of the user interest by *exploring* further the data space trying to collect more insight on what the user likes as well as *exploiting* the knowledge we have collected.

Exploration. To explore our data space, our framework defines sampling areas over multiple *exploration levels*. For a given level, we divide each normalized attribute domain into width ranges that cover a given

percentage of the normalized domain, effectively creating a number of grid cells (Figure 2(a)). The width of each cell defines the granularity of the specific exploration level. A lower number leads to more grid cells of smaller width per dimension. Each cell in our grid covers a certain range of attribute values for each of the d exploration attributes. Therefore, each cell includes a set of unique attribute value combinations and it includes the data objects that match these attribute values. For the same exploration level we also construct k clusters, by clustering off line all data in the data space. By default our highest level creates a single cluster and each level doubles the number of clusters of its previous one.

Our exploration step starts by sampling at the highest exploration level (i.e., with one cluster and one grid cell) and moves on at each iteration to the next lower level until the user terminates the exploration. At each level it samples dense areas by collecting one random sample within each cluster of that level. Next, it samples sparse sub-areas by retrieving one random sample within specific grid cells of the same exploration level. These are the non-empty grid cells from within which no sample has been retrieved yet. The user is presented with the all collected samples and provides feedback. This approach adjusts the sample size to the skewness of our exploration space (i.e., we collect more samples from dense sub-areas) while it ensures that any sparse relevant areas will not be missed (i.e., sparse sub-areas are sufficiently explored).

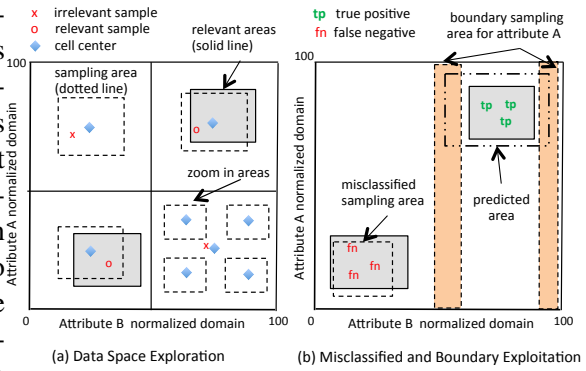


Figure 2: Exploration and Exploitation

Exploitation. The exploitation step aims to leverage prior knowledge. At each iteration it uses the feedback collect so far as well as the latest user model to identify promising data areas to be sampled further. Specifically, we exploit the following information:

- *Misclassified objects*: We randomly sample around each false negative (i.e., objects labeled as relevant but classified as non-relevant by the latest user model) (Figure 2(b)). To further reduce the sampling areas (and hence the sampling overhead), clustering techniques are used to identify close-by false negatives (which most likely belong to the same relevant area) and create a single sample area around each of the generated clusters. This technique increases both the precision and the recall of the user model (since it increases the relevant samples) while reducing the false negatives.

- *Decision boundaries*: Given a set of relevant areas identified by the decision tree classifier, we randomly sample around each boundary (Figure 2(b)) aiming to refine them and improve the accuracy of our user model. The approach is applied in parallel to all the boundaries of the relevant areas, allowing us to shrink/expand each area as we get more feedback from the user.

In Figure 3 we demonstrate the impact our exploration and exploitation techniques. Our evaluation metric is the number of samples we need to reach different accuracy levels of our user model. We measure accuracy by the F-measure (i.e., the harmonic mean of precision and recall) and we assume the user's relevant objects lie within one single area that covers 7-9% of the normalized domain of the `rowc` and `colc` attributes of the `PhotoObjAll` table in the SDSS database [26]. In this figure, *Explore* uses only the exploration step, *Explore/ExploitFN* uses the exploration and the exploitation of the false negatives and *Explore/ExploitAll* uses the exploration and all exploitation steps (i.e., false negatives and boundaries). The results show that combining all three steps gives the best results, i.e., better accuracy

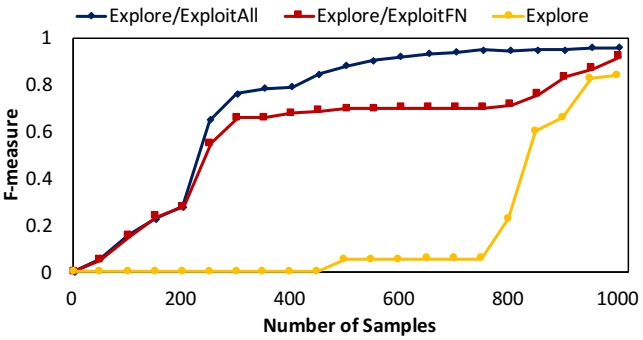


Figure 3: Exploration & Exploitation Impact

with fewer labeled samples. Specifically, using only the exploration step requires more than 800 labeled samples to reach an accuracy higher than 20%. Adding the false negative exploitation increases the accuracy by an average of 54%. Finally, adding the boundary exploitation further improves the accuracy by an average of 15%. Hence, all three phases are highly effective in predicting relevant areas while reducing the amount of user effort.

3.2 SVM: discovering non linear patterns

Non-linear patterns, that is, patterns that cannot be captured by range predicates, are prevalent in applications ranging from location-based searches to scientific exploration tasks using complex predicates. While our decision tree based approach can approximate non-linear patterns, it suffers from poor performance. For example, to predict a circle-shaped relevant area "$(rowc\text{–}742.76)^2+(colc\text{–}1022.18)^2 < 100^2$" on two location attributes $rowc$ and $colc$ in the SDSS data set [26], the decision tree model required over 1800 training samples and approximated the circle region with 95% accuracy using 71 range predicates combined through conjunction/disjunction, as illustrated in Figure 4. This motivated us to seek a more efficient approach to supporting non-linear patterns, reducing both the user labeling effort and the querying and sampling cost in the database.

Our exploration approach uses Support Vector Machines (SVMs) as the classification algorithm [7]. Here, the training set (i.e., labeled samples) in the *data space* is mapped, via a *kernel function*, to a higher-dimensional *feature space* where examples of different classes are linearly separable. Figure 5 shows a 3-dimensional feature space (manually selected by us) where the training points of the circle area in Figure 4 are linearly separated; in practice an SVM may need many more dimensions to see such linear separation. Then among the many hyperplanes that might classify the data in the feature space, SVM selects the one, called the *decision boundary* $\mathcal{L}$, with the largest distance to the nearest mapped samples of any class; this boundary $\mathcal{L}$ is used as the model of user interest as it separates relevant from irrelevant objects. The main challenge here is to identify at each iteration of the exploration process, the next to-be-labeled sample that can quickly improve the accuracy of the current user model $\mathcal{L}$.

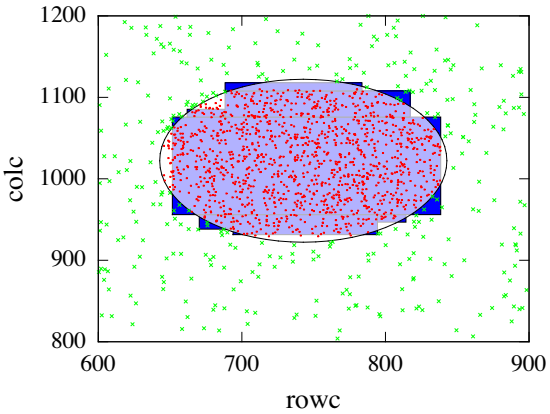


Figure 4: A circle area (green area) approximated by decision trees (blue area), with training points in 2-D data space (red: Y, green: N).

Exploration. Recent active learning theory [2] proposed to choose the example closest to the current decision boundary. However, they suggest a search through the entire data set in *each* iteration, which is prohibitively expensive. Precomputation to store the distance of each tuple to the current decision boundary is not possible either, because the decision boundary changes in each iteration. Our system puts active learning theory into practice: we find the unlabeled example closest to the current decision boundary $\mathcal{L}$ without retrieving all the tuples and evaluating their distances to $\mathcal{L}$. Our system uses two techniques for identifying samples to show to the user in each iteration.

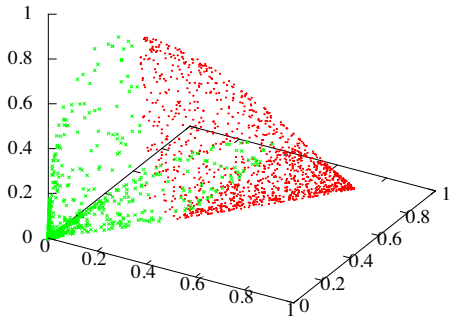


Figure 5: Linear separation of training points in a 3-D feature space using SVM (red: Y, green: N).

Bounding the search area using decision trees. We define a δ-region around the current SVM decision boundary $\mathcal{L}$ and form a two-class training data set such that points inside the δ-region are the relevant class and points outside the δ-region are not. Then a decision tree can be trained to approximate the δ-region and can be easily translated to an exploration query, Q, to send to the database $\mathcal{D}$.

Finally given the query result $Q(\mathcal{D}) \subseteq \mathcal{D}$, we iterate over this set and find the example closest to $\mathcal{L}$. Note that δ can be set to balance two trends: a too small δ can lead to too few training points in the relevant class while a too large δ may result in $Q(\mathcal{D}) = \mathcal{D}$.

Branch and bound search. Our system also builds indexes such as R-trees [1] and CF trees [29] over the database, and performs fast branch-and-bound search over these indexes. Take R-tree for example. Each R-tree node offers a hyper-rectangle, $[a_j, b_j],\ j = 1, \ldots, d$, as a minimum bounding box of all the data points reachable from this node. Given the current SVM decision boundary $\mathcal{L}$, we search the R-tree top-down in a depth-first fashion and always maintain the current closest tuple, $\boldsymbol{x}^*$, and its distance to $\mathcal{L}$, $f(\boldsymbol{x}^*, \mathcal{L}) \stackrel{\text{def}}{=} f^*$. Note that $f^* = +\infty$ before any leaf node is visited. For each intermediate node visited, we dynamically compute a lower bound of the distance from any point in its hyper-rectangle to $\mathcal{L}$ by calling a constrained optimization solver: $\min_{\boldsymbol{x}} f(\boldsymbol{x}, \mathcal{L})$ s.t. $a_j \leq \boldsymbol{x}^{(j)} \leq b_j,\ j = 1, \ldots, d$. If the lower bound is already higher than f^*, we can prune the entire subtree rooted at this node. Once we reach a leaf node, we can update $\boldsymbol{x}^*$ and f^* accordingly. Then the final $\boldsymbol{x}^*$ is the closest tuple to $\mathcal{L}$ in the entire database.

While the above optimizations may lead to a more efficient implementation of active learning theory, we also observe that existing learning theory falls short in two aspects.

Convergence rates. Although active learning theory aims to identify the best samples from input to expedite the convergence of the model, it offers only asymptotic results on the convergence rate [27]. To bring such theory to practice, it is crucial to know the accuracy of the user interest model at any point of the exploration process, so that the DBMS knows when the model is "good enough" and can return the rest of relevant objects from the database with high confidence. Our current research is performing an in-depth analysis of the convergence rate in order to provide useful runtime information on the effectiveness of a learned model.

Sparse or insufficient samples. Our initial results reveal that SVM-based exploration strategies suffer from slow convergence when the data space involves more than 4 or 5 dimensions (*high dimensionality*) and when the true user interest model amounts to a very small area in the total data space (*high selectivity*). In the case of high dimensionality, samples are sparsely distributed in the data space, which prevents SVM from learning the correct model efficiently even if the number of truly relevant dimensions is limited. In the case of high selectivity, the true user interest model may select less than 1% of the objects in the database. Existing active learning theory leads to a sample set that is strongly biased towards negative samples. Such imbalance between negative samples and positive samples also causes SVM to take long to learn the correct model. Our ongoing work is exploring new sampling techniques for the workloads where existing active theory does not work well.

4 Supporting Interactive Performance

Our research aims to advance the state-of-the-art of database system design by developing new query processing and optimization techniques for new data exploration workloads. Next we describe two optimizations that are part of our future work.

4.1 Sample Acquisition Optimizations

Our data exploration approach adds a big processing overhead on the underlying data processing engine due to large numbers of sample acquisition queries issued to retrieve additional samples. These queries represent new interesting workloads in the data processing back-end. Next we discuss the unique characteristics of these exploration queries and propose optimizations for these workloads.

In the decision tree-based exploration, our results indicated that our data exploration workload consists mostly of numerous range queries on individual attributes. Specifically, for k d-dimensional data areas characterized as relevant by the classifier and for m misclassified objects we execute $(k + m) \times d$ range queries, or

hyper-rectangle queries in the d-dimensional space, in each iteration[1]. To illustrate the workload better, consider two attributes, A and B, used in a decision tree. The exploration queries may include Q1: $A \in [a_1, a_2]$ and $B \in (-\infty, \infty)$, where dense sampling is performed for $A \in [a_1, a_2]$ and $B \in [b_1, b_2]$, and sparse random sampling is performed for $B \in (-\infty, b_1) \cup (b_2, \infty)$ to check if the attribute B is irrelevant to the user interest. Similarly, we may also have a simultaneous exploration query Q2: $A \in (-\infty, \infty)$ and $B \in [b_3, b_4]$, where areas of $A \in [a_3, a_4]$ and $B \in [b_3, b_4]$ are densely sampled while $A \in (-\infty, a_3) \cup (a_4, \infty)$ is randomly sampled. If we have a covering index on (A, B, C), the access patterns of the two queries in the index are illustrated in Figure 6. As the dimensionality of the exploration queries increases, e.g., to 5 or 10 attributes, the number of simultaneous queries and the overlap among them increase significantly. Similarly, in the SVM-based exploration, the exploration queries may be k-nearest neighbor queries from a set of data points (support vectors), which can be viewed as a set of ball-shaped queries in a d-dimensional space where significant overlap among these queries exists.

Since each iteration of query steering may issue many simultaneous queries with possible overlap in the data space, separate evaluation of these queries wastes processing resources and leads to poor performance. Traditional multiple-query optimization [20, 21, 23] and continuous query processing (e.g., [5, 6]) focus on analyzing query expressions and finding the common sub-expressions for shared processing. Our exploration queries, however, have more clearly-defined access patterns, e.g., a set of hyper-rectangles or ball-shaped areas in a d-dimensional space. Hence more effective optimizations may be possible. For instance, given a covering index that includes all attributes in the exploration queries, a single scan of all the leaf nodes of the covering index offers an improvement because it avoids repeated scans of the overlapped regions among queries. When the index is large itself, better prediction of necessary regions to visit allow us to skip a fraction of leaf nodes of the index. Furthermore, some queries could mix dense sampling in focused regions and sparse random sampling in wide regions as shown in Figure 6. Hence, the multi-query optimization will also consider quick random sampling using indexes [19].

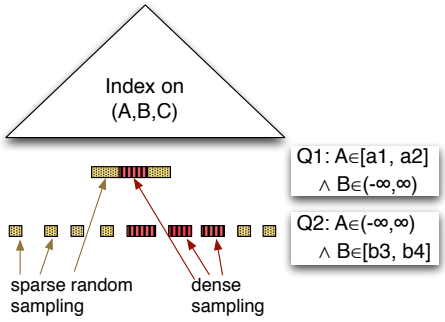


Figure 6: Access patterns of two exploration queries in a covering index.

4.2 Model-based Collaborative Filtering: Predicting exploration trajectories

Data exploration tasks can greatly benefit from information about past user profiles. To this end, we recently focused on how to leverage detailed information of user profiles with the end goal of improving the efficiency of interactive data exploration tasks. User profiles include feedback on data objects, the final classification model as well as its lineage, i.e., sequence of steering queries and samples that resulted from the exploration process.

To leverage past user exploration profiles, we employ a *model-based collaborative filtering* approach. Collaborative filtering (CF) uses the known preferences of a group of users to predict the unknown preferences of other users. Model-based CF systems involve learning a model based on data objects (aka items) ratings. This allows the system to learn to recognize complex patterns based on the training data, and then make intelligent *predictions* for collaborative filtering tasks based on the learned models without having to use the complete data set every time. This offers the benefits of speed and scalability making model-based CF techniques a good fit for real-time predictions on the basis of very large data sets.

In the context of our interactive data exploration framework clustering models can be used to predict relevant areas and recommend them to the back-end as promising sampling areas. Specifically, clustering algorithms can identify groups of users who appear to have similar relevance feedback. Once the clustering model is created, we can leverage it during an online exploration task as follows. Given the relevance feedback of the current

[1]In some cases, the data space can be reduced to include only relevant attributes however, in the worse case scenario, sampling queries are executed on all d dimensions of the exploration space.

user, the system predicts the cluster it belongs to, i.e., its neighbors with respect to their relevant objects. Then a traditional CF algorithm is applied to rate candidate areas to sample based on the interests of *only* these neighbors. At each iteration, more feedback is collected and our prediction of the "neighbors" improves helping the system to converge faster to the current user's classification model. Our clustering-based approach has better scalability than typical CF methods because it makes predictions within much smaller clusters rather than the entire user and object base. The complex and expensive clustering computation is run once offline and the resulting clusters can be used by any future user. Next we sketch the technique in more detail.

Cluster-based Exploration CF techniques use user ratings on data objects to calculate the similarity between users (or objects) and make predictions according to those calculated similarity values. However, similarity values are based on common data objects and therefore are unreliable when the common data objects rated by the users are therefore few. This is indeed the case of our interactive exploration framework: users provide feedback on few sample objects of the entire database and the intersection of labeled data sets of past users is often quite small, even when user interests highly overlap. To address this challenge we apply our clustering-based exploration on the level of the predicted relevant areas as opposed to labeled items by using past user models.

Each past user of our interactive exploration framework, $u_i \in \{u_1, u_2, ..., u_m\}$, is represented by its final classification model that characterizes the relevant areas for that user. Given a collection of user models one can identify a partitioning schema of the data space such that each partition $p_i \in \{p_1, p_2, ..., p_n\}$ involves items that are *all* relevant to the *same* set of users. Each user is represent by (partition, relevance) pairs which can be summarized in a user-partition table R. This table R contains the relevance score R_{ij} that is provided by the user u_i on the partition p_j. For a binary relevance feedback model R_{ij} is 1 if the partition p_j contains objects characterized as relevant by the model of user u_i and is 0 otherwise. Alternatively one can use a fixed partitioning schema of the exploration space in which case the relevance score R_{ij} is the degree of overlap between the user's u_i predicted relevant areas and the partition p_j.

We use a clustering algorithm on the user partition table R to form groups of users that appear to provide similar relevance feedback (identified the same partitions as relevant). We can then assign the current user to one of these clusters as follows. Given the active user's u_a latest decision tree, we calculate the overlap of the user's relevant areas with each data partition p_i. We create a vector with the overlap per partition R_i vector to characterize the current user. Full overlap indicates a relevance of 1 for that partition and no overlap a relevance of zero. Partial overlap can be calculated based on the size of the overlapping area. Using this vector the current user will be assigned to the cluster with the most relevant past users (i.e., users that explored similar partitions).

Once the neighborhood is obtained, a CF algorithm is used to predict the relevance score for all partitions. In particular, the relevance (prediction) score $R_{a,j}$ for the active user u_a on a partition p_j can be calculated by aggregating the neighbors relevance feedback on the partition p_j using a number of aggregation functions. Examples include averaging the relevance score of that partition over all the neighbors or calculating the weighted average of the neighbors relevance feedback where the weight is based on the Pearson similarity between the neighbor and the current user (i.e., the more similar two users are wrt to their feedback, the higher their weight on identifying promising sampling areas).

Our data exploration framework can leverage the output of the above process in multiple ways. For example, we can use the relevance score of each partition for the current user to apply a weighted sampling on them, i.e., more relevant partitions are sampled with higher ratio than less relevant ones. Diversification techniques [16] can also be combined with the relevance score in order to identify the most diversified set of samples with high relevance score to show to the current user. Collecting feedback on these samples will allow our system to collect more insight on the user's interests. All the above can be executed iteratively - each round improves the user model which improves also the relevance predictions on each partition increasing eventually the convergence rate of our exploration.

4.3 Dimensionality Reduction

Our initial results revealed that the exploration space is often characterized by high redundancy. As data sets become highly multi-dimensional data exploration suffers from slow convergence. This is due to the fact that many dimensions are correlated with others while some are irrelevant to the user's interest.

Offline Reduction Eliminating redundant feature combinations can be done offline by removing dimensions with low variation, since these dimensions will not be "useful" to our classifier. To demonstrate this with an extreme example, if all sky objects have the same brightness level, then the brightness attribute will not be a good separator of irrelevant/relevant objects. Variation along each dimension can be captured by the variance of its values and dimensions with relatively low variance can be removed. Principal component analysis (PCA) can also be used offline to identify only the uncorrelated dimensions our models should be using.

Online Reduction Online reduction techniques aim to identify dimensions irrelevant to the current user and hence eliminate them as early as possible from the exploration space. One approach we explore is based on the expectation that the distribution of relevant labels collected from the current user will be more scattered when projected on irrelevant dimensions and more clustered on specific areas of the relevant domains. Based on this, projecting the samples characterized as relevant on each dimension of our exploration area and comparing their distribution across each dimension independently could indicate the most relevant dimensions and allow us to adapt the number of samples to be higher for the relevant domains.

We also leverage past user models to identify dimensions irrelevant to the current user. Specifically, past user models (i.e., decision trees) reveal the relevant dimensions for past users. Hence, once the neighbors of the current user are identified we apply dense sampling on the relevant dimensions for these neighbors and sparse sampling on the rest of the exploration domains. The collected feedback is further used to identify the relevant dimensions for the current user as described above.

5 Prototyping and Demonstration

We have implemented a prototype of our interactive data exploration framework that includes the techniques we described in Section 3.1 and 3.2. The system is designed on top of a relational database system and its architecture (Figure 7) includes three software layers: (1) an interactive visualization front-end, (2) the AIDE middleware that implements our exploration techniques and (3) a database backend supporting our data exploration. An initial prototype of our system was demonstrated at VLDB 2015 [7].

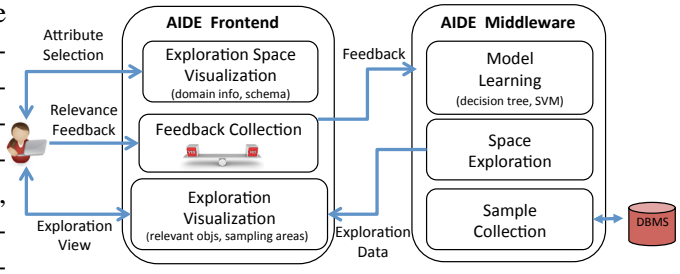
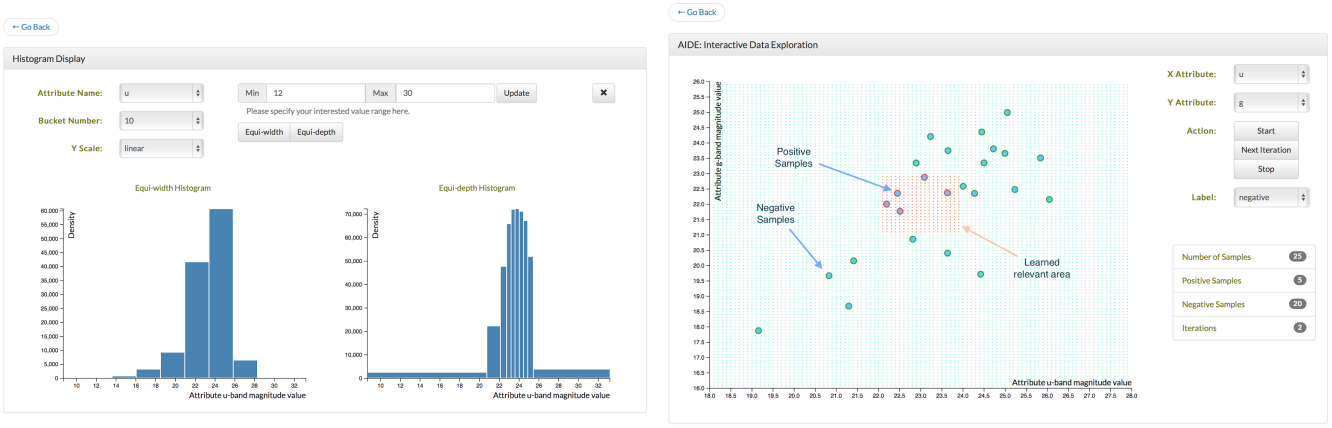


Figure 7: Architecture of our system

Our visualization front-end provides several functionalities. The user is initially presented with the database schema and he can select an initial subset of attributes of interest, which will be refined later by the data exploration. Our front-end can also visualize domain distributions of each attribute to further allow the user to filter attributes based on the domain characteristics and restrict the value ranges of the relevant attributes for consideration (e.g., focus on a dense region or a region close to a landmark). Users can select between different types of plots of the underlying data distributions, such as histograms and heat maps. Figure 8a shows a histogram example on an attribute in the SDSS [26] data set.

The system starts a series of iterations of sample labeling, model learning and space exploration. The front-end supports this process by visualizing various subspaces of the exploration attributes, presenting data samples to the user, collecting yes/no labels from the user regarding the relevance of the shown samples and showing the locations of labeled samples in the exploration space. Figure 8b shows an example of this interface.



(a) Histogram visualization for exploration attributes.

(b) Exploration visualization (learned areas, labeled samples).

Figure 8: Front-end visualization interface.

Sitting below the visualization front-end is the "automatic user steering" layer (middleware in Figure 7), which is the heart of our system. This component is implemented in Java, with a few machine learning libraries integrated in the system. At each iteration it incorporates the newly collected labeled samples and generates a new classification model. At any point the user can request a visualization of the current user model (i.e., decision tree or SVM decision boundary) which entails highlighting the objects classified as relevant to the user. The user can then decide to stop the exploration (if he is satisfied with the current set of identified objects) or to proceed to the next round of exploration.

The database backend uses PostgreSQL. The database engine includes various sampling techniques implemented as stored procedures. These techniques are designed to support the exploration approaches we discussed above. For example one procedure supports the decision tree approach to learning linear patterns (§3.1) by selecting a predefined number of random samples within a given distance from the center of a d-dimensional area, while other procedures support random and weighted sampling.

6 Related Work

Numerous recent research efforts focus on data exploration. The vision for automatic, interactive navigation in databases was first discussed in [4] and more recently in [28]. YMALDB [10] supports data exploration by recommending to the user data similar to his query results. DICE [15] supports exploration of data cubes using faceted search and in [12] they propose a new "drill-down" operator for exploring and summarizing groups of tuples. SciBORQ [25] relies on hierarchical database samples to support scientific exploration queries within strict query execution times. Idreos et al. [18] proposed a system for interactive data processing tasks aiming to reduce the time spent on data analysis. In [22] they interactively explore the space based on statistical properties of the data and provide query suggestions for further exploration. In [11] they propose a technique for providing feedback during the query specification and eventually guiding the user towards her intended query. In [13] users rely on prefetching and incremental online processing to offer interactive exploration times for window-based queries. SearchLight [14] offers fast searching, mining and exploration of multidimensional data based on constraint programming. All the above systems differ from our approach: we rely on the user's feedback on data samples to create progressively an effective training set for generating machine learning models that predict the user's data interests.

7 Conclusions

This article provided an overview of our on-going work on learning-based data exploration. We highlighted the research challenges and a set of solutions that attempt to address them. Our proposed techniques can be crucial for deriving insights from huge and complex data sets found in many discovery-oriented applications. Human exploration effort across large data sets will be significantly reduced, as users will be methodically steered through the data in a meaningful way. Such automated steering, fully exploiting user interests and application characteristics while grounded in rigorous learning theory, will assist users in discovering new interesting data patterns and eliminate expensive ad-hoc exploratory queries.

8 Acknowledgments

This work was funded by NSF grants IIS-1253196, IIS-1218524, and IIS-1218524 and a gift from HP Labs.

References

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. 1990.

[2] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *J. Mach. Learn. Res.*, 6:1579–1619, Dec. 2005.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[4] U. Çetintemel, M. Cherniack, J. DeBrabant, Y. Diao, K. Dimitriadou, A. Kalinin, O. Papaemmanouil, and S. Zdonik. Query steering for interactive data exploration. In *Proceedings of the 6th Biennial Conference in Innovative Data Systems Research (CIDR)*, 2013.

[5] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for internet databases. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD*, 2000.

[6] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer. Path sharing and predicate evaluation for high-performance XML filtering. *ACM Trans. Database Syst.*, 28(4):467–516, 2003.

[7] Y. Diao, K. Dimitriadou, Z. Li, W. Liu, O. Papaemmanouil, K. Peng, and L. Peng. AIDE: An Automatic User Navigation Service for Interactive Data Exploration (Demonstration). In *VLDB*, 2015.

[8] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-Example: An Automatic Query Steering Framework for Interactive Data Exploration. In *33rd ACM Special Interest Group in Data Management (SIGMOD)*, 2014.

[9] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. AIDE: An Active Learning-based Approach for Interactive Data Exploration. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28(11):2842 – 2856, 2016.

[10] M. Drosou and E. Pitoura. YMALDB: exploring relational databases via result-driven recommendations. *VLDB Journal*, 22:849–874, 2013.

[11] L. Jiang and A. Nandi. SnapToQuery: Providing Interactive Feedback During Exploratory Query Specification. *VLDB 2015*.

[12] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran. Smart drill-down: A new data exploration operator. *VLDB 2015*.

[13] A. Kalinin, U. Cetintemel, and S. Zdonic. Interactive data exploration using semantic windows. In *SIGMOD*, 2014.

[14] A. Kalinin, U. Çetintemel, and S. B. Zdonik. Searchlight: Enabling integrated search and exploration over large multidimensional data. *VLDB 2015*.

[15] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and Interactive Cube Exploration. In *ICDE*, 2014.

[16] H. Khan, M. Sharaf, and A. Albarrak. DivIDE: efficient diversification for interactive data exploration. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management (SSDBM' 14)*, 2014.

[17] Large synoptic survey telescope: the widest, fastest, deepest eye of the new digital age. `http://http:/www.lsst.org/`.

[18] Martin Kersten and Stratos Idreos and Stefan Manegold and Erietta Liarou. The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds. *International Conference of Very Large Databases (VLDB)*, 4(12), 2011.

[19] F. Olken. *Randomized sampling from databases*. PhD thesis, University of California, Berkeley, 1993.

[20] A. Rosenthal and U. S. Chakravarthy. Anatomy of a mudular multiple query optimizer. In *Proceedings of the 14th International Conference on Very Large Data Bases*, VLDB '88, pages 230–239, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[21] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. Efficient and extensible algorithms for multi query optimization. *SIGMOD Rec.*, 29(2):249–260, May 2000.

[22] T. Sellam and M. L. Kersten. Meet Charles, big data query advisor. In *biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[23] T. K. Sellis. Multiple-query optimization. *ACM Trans. Database Syst.*, 13(1):23–52, Mar. 1988.

[24] B. Settles. *Active Learning*. Morgan & Claypool, 2012.

[25] L. Sidirourgos, M. Kersten, and P. Boncz. SciBORQ: Scientific data management with Bounds On Runtime and Quality. In *CIDR*, 2011.

[26] Sloan digital sky survey. `http://www.sdss.org/`.

[27] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, Mar. 2002.

[28] A. Wasay, M. Athanassoulis, and S. Idreos. Queriosity: Automated Data Exploration. In *IEEE International Congress on Big Data*, 2015.

[29] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD*, 1996.

Towards a Benchmark for Interactive Data Exploration

Philipp Eichmann, Emanuel Zgraggen, Zheguang Zhao, Carsten Binnig, Tim Kraska
Brown University, Providence, RI, USA

Abstract

Existing benchmarks for analytical database systems such as TPC-DS and TPC-H are designed for static reporting scenarios. The main metric of these benchmarks is the performance of running different SQL queries over a predefined database. In this paper, we argue that such benchmarks are not suitable for evaluating modern interactive data exploration (IDE) systems, which allow data scientists of varying skill levels to manipulate, analyze, and explore large data sets, as well as to build models and apply machine learning at interactive speeds. While query performance is still important for data exploration, we believe that a much better metric would reflect the number and complexity of insights users gain in a given amount of time. This paper discusses challenges of creating such a metric and presents ideas towards a new benchmark that simulates typical user behavior and allows IDE systems to be compared in a reproducible way.

1 Introduction

There exists an ever *growing set of data-centric systems that allow data scientists of varying skill levels to manipulate, analyze and explore large data sets*. For example, tools like Tableau or Cognos allow users to quickly analyze high-dimensional data using an interactive and visual interface. Research prototypes like imMens [27], DICE [16, 17] or IDEA [9] aim to improve upon systems like Tableau by using specialized data structures, prefetching, and/or approximation techniques to guarantee interactive latencies over big data sets. Other research projects like SeeDB [20] or Data Polygamy [5] help users during the exploration process by providing recommendations for interesting visualizations or correlations, whereas systems like DataWrangler [19], Trifacta [45] or Paxata [33] assist users in data wrangling and cleaning.

Although many systems and techniques have been proposed, there is currently no systematic way to evaluate and compare them. For instance, DICE [17] and IDEA [7, 9, 12] both aim to allow users to interactively explore large data sets, but do so through different techniques. Both systems leverage speculative execution, but DICE uses a set of pre-defined exploration paths (e.g., drill-down, roll-up, pivot, etc.) to decide what to speculatively execute, whereas IDEA's model is based on a simpler set of operations which only depend on the current set of visualizations shown on screen. Similarly, both systems try to re-use results between interactions. While DICE finds overlap among different queries using standard relational algebra rewrites, IDEA rewrites queries on the probability level. This allows IDEA, in contrast to DICE, to also reuse incomplete results. Furthermore, IDEA's

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

goal is to support complex analytical workflows including predictive model building. Therefore, IDEA provides an unbounded 2D workspace and a set of pen/touch gestures to support complex analytical workflows. As opposed to DICE, which uses a more traditional interface and focuses mainly on the interactivity of traditional data-cube operations. Even though both systems are designed for interactive analytics, it is hard to draw a comparison due their different approaches and goals. Moreover, systematically comparing a system which provides approximate results, like IDEA, with a system which always calculates the full result, like Hyper [22], is a complicated question on its own. This is due to the fact that it requires determining when an approximate answer meets the same standards as the final answer to the user.

What is needed is a new benchmark for interactive data exploration systems. This is challenging because such a benchmark has to be user focused and the system's performance, which is the main metric of existing analytical benchmarks such as TPC-H [44] and TPC-DS [43], is no longer the most important metric. Moreover, in interactive data exploration systems users incrementally build queries over the course of a session, which is another factor that is not represented in the workloads of the previously stated analytical benchmarks. Arguably the only important metric for data exploration systems is how efficient a user can gain insights from a new data set such as *Insights per Minute*. Clearly, the time to execute a single query has an impact on this metric; the longer queries run, the longer users take to find an interesting insight. What makes it more complicated is that many other aspects have an impact too. First, as IDEA and DICE have demonstrated, reuse of intermediate results and idle time between interactions can be exploited therefore making workflow performance dependent on many other factors. Secondly, the time of fully running a single query matters less. For example, a system which provides 99% accurate answers for 10 queries is – in most cases – superior over a system which provides a 100% correct answer for a single query in the same amount of time. Third, a system that slowly executes some of its queries, but additionally recommends some interesting visualizations to the user or detects common data errors and other pitfalls is arguably the better tool. Finally, the user interface itself often can have a profound impact on how quickly users can make discoveries.

However, measuring the time to insights requires the design, implementation and evaluation of open-ended user studies, which can be time-consuming, expensive and hard to compare (see Section 3.1). Instead we argue for a benchmark that simulates common user behaviour during visual data exploration sessions. While this approach does not allow comparison of all facets of data exploration systems (e.g., the quality of a visual recommendation), we hope it will enable comparing techniques and systems across a multitude of common interactive exploration tasks. Furthermore, such benchmark results can then be augmented with individualized user studies to evaluate features outside of the scope of this benchmark.

Finally, as discussed before, recent interactive data exploration tools not only focus on the aspect of visual data browsing using simple analytical functions but also offer a broad range of other extensions ranging from interactive model building and machine learning over producing visual recommendations to interactive data cleaning. Covering all these aspects is a major challenge when designing a new benchmark.

In this paper, we make the first step towards defining such a new benchmark. We discuss the challenges of simulating users for benchmarking the different components of an interactive data exploration system. We also suggest potential metrics to measure the performance of these systems. The remainder of this paper is outlined as follows: Section 2 describes a potential data exploration session and its various facets; Section 3 talks about insights per minute as a metric and why it is problematic to measure in a benchmark; Section 4 discusses challenges and initial solutions for benchmarking the core functionality of an interactive data exploration systems. Sections 6, 5, and 7 extend upon these ideas to also include other aspects of IDE including visual recommendations, model building, risk evaluation, and data cleaning issues. Section 8 surveys ways to evaluate the overall user experience, and finally, Section 6 concludes.

2 A Motivational Example

To motivate various aspects an IDE benchmark should take into consideration, we outline a fictional data exploration scenario inspired by projects that have recently emerged in this field, such as Vizdom/IDEA [8, 9], SeeDB [20], QUDE [2], and imMens [27].

2.1 Use Case

Eve is a medical researcher at a major Boston area hospital. She obtained a new data set containing information about intensive care unit (ICU) patients including demographics, physical information, disease codes as well blood test results, as available in the MIMIC II data set [29]. She wants to get an overview of the data and gain insights. Eve starts off by visualizing different attributes containing physical and demographic information by creating visualizations depicting the distribution of all ages, weights and heights (Figure 1 A). She notices that patient ages are frequently set to -1, indicating that those values were not correctly recorded. She applies a data cleaning step that replaces all such occurrences with *null* (Figure 1 B) such that they are treated by the system as missing values. Weight and height have the same issue. However, because weight and height are strongly correlated, she decides not to substitute the missing values with *null*, but to perform a different data-cleaning operation that estimates missing values based on correlated attributes (Figure 1 C).

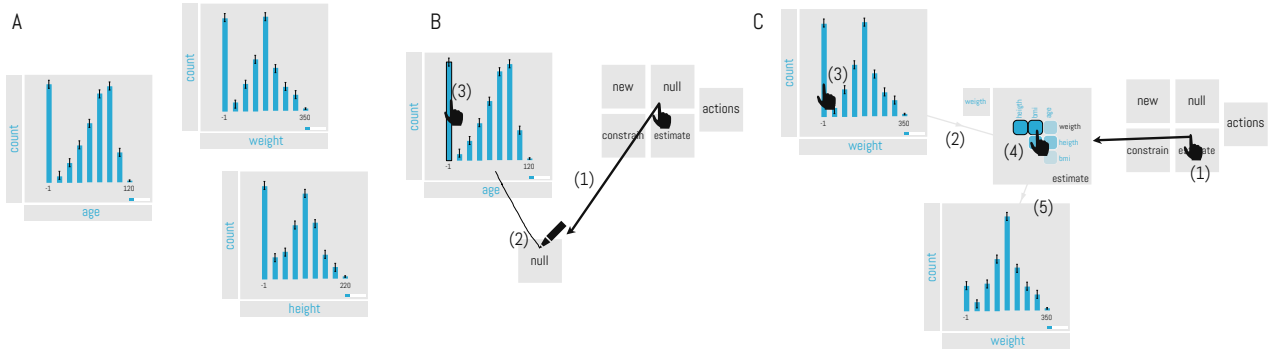


Figure 1: A: visualizations of patients' ages, weights and heights. B: Eve drags out a data-cleaning operator (1), links it to the plot of ages (2) and finally selects the range of age values that she wants to replace with *null*. C: Eve drags out a different data-cleaning operator (1). She links it to the weight visualization (2) and again selects the range of weight values she wants to clean (3). The system shows her the attributes that are most correlated with weight and Eve decides to use height and BMI (4) as the base attributes to estimate missing weight values. She creates a new visualization that shows the resulting weight distribution after this cleaning step.

Next, she is interested in finding out more about age distributions of different diseases. She creates a chain of visualizations that allows her to inspect age distributions of patients with a metabolic disease, and of patients with a heart failure (Figure 2). She then realizes that the two distributions are dissimilar and decides to test the difference for statistical significance. Because Eve substituted "-1' values with *null*, the system automatically knows that it should not consider the *null*-values as part of the permutation test. Optionally, based on these interactions the system could automatically recommend other attributes (i.e., other diseases) that might be of interest to Eve.

Eve now decides to take it a step further and test to see if she can train a classifier to predict whether a patient has a metabolic disease with a set of hand-picked attributes such as age and BMI (Figure 3). The system displays accuracy and other statistics about this model and additionally proposes modifications, such as adding further indicative attributes or changing the underlying prediction algorithm, that will increase the classifier's performance.

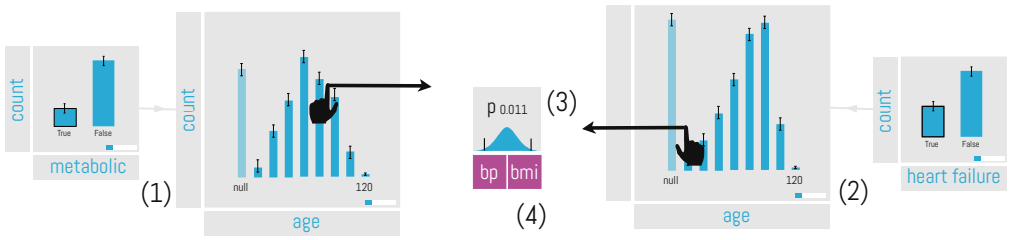


Figure 2: Chains of visualizations that show age distributions of patients with a metabolic disease (1) and heart failure (2). By dragging the two visualizations closer together the system performs a permutation test and displays significance levels (3). The system also displays additional attributes (4) that Eve might be interested in, as they might exhibit similar significance levels when comparing such sub-populations.

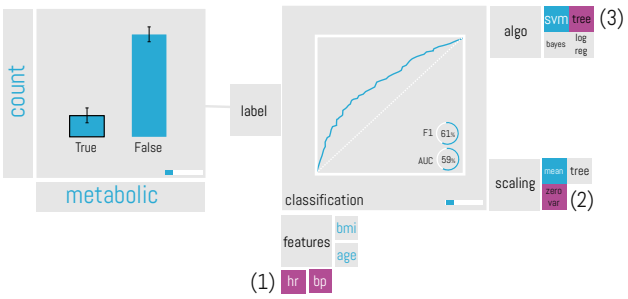


Figure 3: Classifier that predicts if a patient has a metabolic disease or not based on a patient's BMI and age. (1, 2 and 3) shows modifications that the system recommends to increase the model's performance.

2.2 Discussion

This use case exemplifies the potential of modern interactive data exploration tools. It also raises the question of whether or not Eve would have gained a higher number or more valuable insights, if she had used a different system. For instance, the results that were presented to her are approximate visualizations as indicated through the error-bars in Figure 1. Would she have gained more insights if they had been exact but took longer to compute? Would the system's time spent on finding correlations (see Figure 2) be better spent on computing more accurate approximations? Would a system that automatically implies that "-1" should be treated as a *null* value, lead to more discoveries over time?

Conducting user studies that measure the number and complexity of gained insight over time in order to compare different systems or variants thereof may shed more light on these and many other questions. However, such insight-based studies are not well suited for evaluating IDE systems, as we describe in the following section.

3 Benchmarking Interactive Data Exploration Systems

In the following, we first discuss why *Insights per Minute* or time it takes to gain the first insight are problematic as a metric for a benchmark for an IDE benchmark and provide an overview of alternative metrics that capture a similar notion.

3.1 The Problem of Using Insights in a Performance Metric

Ultimately, the goal of interactive data exploration is to extract insights from data. Thus, a system that allows to extract more insights than another system within a given time frame is preferable. However, creating a measure that captures this notion in a comparable and reproducible way is hard. What is an insight? Do different users

have different notions of insights? How do we measure the complexity and value of an insight? Are they domain-dependent?

Recent work in the information visualization community has tried to address these questions. Various studies and guidelines [13, 26, 39] proposed to conduct user studies that include open-ended tasks with domain experts. However, evaluating data exploration systems with user studies is problematic for the following reasons: (1) These studies are expensive to conduct and require a lot of manual coding. (2) The results are hard to compare and reproduce. These studies usually have a small number of participants and use within-subject designs making the measured metrics (e.g., insights per minute) unusable for comparisons across different studies. Additionally, different levels of user expertise and that different studies sample from different populations, e.g., students vs. domain experts, further complicate comparisons of study results. (3) It is difficult to design controlled user studies for entire systems. Different user interfaces (UIs) might incentivize different types of insights, e.g., bar chart vs. pie chart, supported complexity of workflows. (4) The value of an insight is unclear. What is more valuable: a simple count comparison or knowing we could build a classifier to predict a label with high accuracy?

3.2 Benchmark Overview

We advocate for a reproducible IDE benchmark that does not factor in the variability of insights, domains, users and user experiences. The core idea is to provide a benchmark that simulates typical user behavior for common data exploration tasks [21, 32], such as filtering, projecting, as well as model building or reacting to recommendations. However, the richer the operations are, the harder they become to benchmark. We therefore suggest a core-set oriented benchmark design where each core-set aims to analyze a different functional aspect of a interactive data exploration system; e.g., one core set only tests simple analytical operations, whereas another one tests more complex model building tasks. Which core-sets are used to evaluate a system, therefore, depends on the supported functionality.

We envision the following four core-sets: *Core-Set I* focuses on Interactive Visual Analysis [21,32] and consists of operations like building filter chains, aggregations, drill-downs, pivoting, etc., as currently supported by systems like Tableau. However, it will exclude interactive model building, which is part of *Core-Set II*. *Core-Set I* and *II* are tightly coupled as it seems unreasonable to assume that one would build a model without having the possibility to efficiently inspect the data set. *Core-Set III* is concerned with benchmarking the recommendation part of a system, whereas *Core-Set IV* outlines metrics to compare the interactive cleaning and risk evaluation parts of a data exploration tool. While a system that supports a higher core-set typically includes the functionality of the core-sets below, we envision that each core-set can be tested individually as discussed before. Furthermore, the higher the core-set number the harder it is to define a benchmark since the sheer complexity of supported operations is increasing and their comparability becomes more difficult. We therefore focus most of our discussion on the lower core-sets and discuss initial ideas for the higher ones.

4 Core-Set I - Interactive Visual Analysis

The core of any IDE system is the capability to browse data through visual interfaces. Techniques like linking and brushing as well as OLAP-like aggregations, traditional statistics, and attribute derivation can help understand complex relationships in the data. The level of support for these operations is determined by various factors including the user interface, the chosen set of default visualizations, and how fluid, i.e., interactive, the system is. As a matter of fact, interactivity is one of the most important aspects as recent studies show. For instance, in [26] the authors argue that latencies of more than $500ms$ can already have a profound impact on discovery rates. In the remainder of this section, we focus on benchmarking the interactivity during Interactive Visual Analysis and discuss more qualitative aspects (e.g., how to evaluate the user interface) in Section 8.

4.1 Objective

The main objective of Interactive Visual Analysis from an interactivity point of view is to ensure that at no point the user is blocked and can always make an informed decision on what to explore next [14, 15]. This requires the system to be fast and responsive, independent of the complexity of a query. Consider a system *A* that provides a visualization to a query within $500ms$ and another system *B*, which takes $10s$ to process the same query. Clearly, *A* is superior over *B*. At the same time, a system which is only $50ms$ faster than another system does not add much value as it makes less of a difference to users. Similarly, a system which provides an approximate answer in $500ms$ and a complete answer for the same query in $10s$, is likely to be superior to a system which forces the users to wait and only returns the complete result after $8s$. On the contrary, short response times (e.g., $500ms$) are barely noticeable to users and therefore only marginally affect the user experience.

To that end, the questions a benchmark should consider are: How does a system, which never provides the full answer but a good approximation compare to a system that eventually provides the complete answer in a reasonable time-frame? What is a reasonable time-frame? How can progressively refining approximate answers be evaluated? How much time is the system given in between interactions? In the following, we outline potential design choices for a benchmark to answer these questions.

4.1.1 Metric

Since we cannot efficiently measure insights per minute directly, we propose a proxy metric called the *Interactivity Performance* metric. One approach would be to measure the time the system takes to provide a good quality estimate for the results of all interactions. Thus, a system, which on average has shorter response times to interactions, would be considered superior. However, as argued before, response times below a certain latency requirement (e.g., $500ms$) are barely noticeable to users and therefore only marginally improve the user experience. Therefore, we suggest a performance metric P that reflects how often and by how much a system violates the latency requirement:

$$P = \sum_{i \in I} max(0; T_q(i) - t_u) \tag{1}$$

Where I is an ordered set of all executed interactions, $T_q(i)$ is the time the system takes to execute interaction T to q-degree of quality and t_u the latency requirement. The smaller P, the better the performance.

Response times in computer systems have been studied extensively for problem solving tasks [38]. It was found that user productivity increases as response time decreases. But systems with a high variability in response time negatively impact user efficiency. However, it remains unclear whether a system which almost always meets the latency requirement with only few major exceptions performs better than one which consistently violates interaction latencies by a little. Potential variations might include different thresholds and weighting schemes (e.g., logarithmic, exponential, etc.) to correctly penalize different system behaviors such as high variability.

There are several ways to automatically determine the quality of an (approximate) visualization. [23] defines a *relationship-based* quality metric as a good approximation in which the relationships between data groups no longer change, i.e., the point in time where in a bar-chart diagram the relationship between two bars (where one is higher than the other) does not change. Another suggested quality metric is the *just noticeable difference* (JND) [42]; a good visualization cannot be visually distinguished from the final complete visualization. In some sense, the JND-based quality metric can be seen as an *error-based* quality metric with a specific error constant. That is, a good visual approximation lies within a pre-defined error-bound of the ground-truth visualization. More specifically, $T_q(i)$ is the time it takes the system to produce a visualization for interaction i within an error-bound of ϵ. We plan to conduct a user study to evaluate which of these quality metrics works best as a placeholder for real-world users.

Finally, it is worth noting that the combination of the error-based quality metric and the Interactivity Performance P allows for a direct comparison of visualization error and the maximum time allotted to compute it. Varying the error bound-threshold ϵ and the maximum latency t_u could help to better understand the trade-offs between computation time and latency. For instance, while it takes one system to compute an approximate visualization within $500ms$ and error threshold 5%, another system may constantly perform better after $1s$.

4.2 Workload

Having the main metric defined, we still need a way to simulate the user. As described earlier, in IDE systems users incrementally build queries over the course of a session. Idle time between interactions is often used by systems to prepare for the next interaction and in contrast to many existing benchmarks, the time users take between interactions varies a lot. In many cases interactions depend on each other and allow for reuse of partial results among other things. As a result, a benchmark for visual data exploration has to somehow simulate users' behavior with their typical think-time between interactions.

One method to derive realistic workflows is to study actual user behavior and synthesize them to exemplary workflows as done in [9]. By providing a set of potential IDE sessions the simulated users can capture common user behavior. This allows for adjustments to the synthesized session (e.g., from more novice to expert users). One idea is that the think-time between interactions could be scaled up or down similar to scaling the data size to increase the level of difficulty in a benchmark. While creating the different workflows it is important to cover different aspects of the data. For instance, one workflow could consist of mainly unrelated browsing queries, where users just look at different attributes in isolation, while another workflow could comprise the creation of a deep analytics pipeline. Similarly, as for other benchmarks, simulated users should vary their interactions (e.g., as [9] shows users tend to investigate outliers as they might contain interesting information).

Finally, a benchmark for visual data exploration should also allow to vary the data and data size. To test the different properties of the system it is important that the generated data follows different types of distributions for various attributes and contains random and correlated data. One way to generate such a data set is to take an existing data set (e.g., the flight data set from [1]) and provide ways to automatically scale it to the desired size while preserving the most important data properties.

4.3 Reporting

An important aspect of a benchmark is to define the main configurations for which the result should be reported. We suggest that users of the benchmark can select a scale-factor of the benchmark just as in traditional benchmarks. In addition, the benchmark should define different configurations such as the *browsing* configuration or the *no think-time* configuration, which have different values for the latency threshold t_u, the think-time between interactions and the error-based quality metric. These configurations allow the user to configure the benchmark to the targeted use case of the IDE system, while still making all systems comparable using the standard settings.

5 Core-Set II - Interactive Model Building

Apart from supporting users in visual analysis tasks, modern IDE systems increasingly help to interactively test the predictive power of attributes and build models [8, 9, 24, 41]. Some of the common tasks in model building, such as visually inspecting attributes for feature selection, are covered by Core-Set I. Others, such as model selection, hyper-parameter tuning etc., need to be evaluated separately and require IDE systems to potentially train and test hundreds of different configurations. While some progress has been made on how to do this with interactive latency guarantees [9, 37, 40], it still remains a challenge to compare different interactive model training systems and techniques.

5.1 Metric

Arguably, the most relevant metric is Time to Model, the time it takes users to derive a model with a satisfying model quality. This, yet again, requires user studies with all its difficulties. Thus, analogous to the ideas for Core-Set I, a better metric for Core-Set II captures how long it takes a simulated user to derive a model with satisfying quality.

Interestingly, estimating the model quality can also be considered a result approximation. Thus, like for Core-Set I, we use the *Interactivity Performance* metric (see Equation 1). However, in this case we must measure the quality of a computed model rather than the quality of a visualization. More specifically, given a workload that exactly specifies which features and model class the system must use (e.g., train an SVM to predict if a patient has a metabolic disease based on its age and BMI, as done in Figure 3), with a pre-defined validation method (e.g., 10-fold cross-validation), we can compute the expected "ground-truth" model quality, e.g., the F-score, offline. This allows us to compare the required time to achieve a certain model quality of a system to the best known model quality, again represented as an F-score.

Furthermore, this metric also allows comparisons of more complex model search strategies. For example, it might be possible to allow the system to do automatic algorithm and feature selection, or even feature transformation. In this case, we envision that the workload only defines a high level task and the system reports the model quality over time. This is then compared to the best known model quality, again represented as an F-score.

It should be noted, though, that different time weighting schemes as well as using the F-score as a quality model have implicit assumptions, which may or may not reflect the users intention. For instance, the F-score is the harmonic mean of precision and recall, yet especially in the medical context often precision is more important than recall.

5.2 Workload

Since model building is strongly intertwined with the tasks discussed for Core-Set I, the workload for Core-Set II should also include operations such as browsing, inspecting attributes, etc. In the workload of this core-set model building tasks should thus be intertwined with simple OLAP-style operations to inspect the data. As most IDE systems are not yet capable of full automatic model building as proposed in MLbase [24], the workload should also support different levels of specificity. Highly specific configurations dictate which algorithm and parameters to use, whereas non-specific ones merely require the system to return predictions regardless of models class, parameters, and features used.

5.3 Reporting

Core-Set I and Core-Set II essentially use the same metric. Therefore, results can be reported as described in Section 4.3. In cases where a strict latency requirement matters less, we suggest reporting the F-score metric for each trained model within the simulated workflows.

6 Core-Set III - Recommendations

There has been an increasing interest in automatically presenting recommendations to the user when using IDE systems. These recommendations often come in different forms; e.g. suggesting visualizations [20], or queries [28], recommending data cleaning steps [45], pointing out potential correlations [5] etc. Evaluating and comparing the quality of such recommendations is difficult as they typically depend on the data domain and the history of interactions.

6.1 Metric

Prior work in this area often use their own metric of success. The visual recommendation system SeeDB [20], for instance, rewards recommendations that are orthogonal to what the user has looked at so far, thus it tries to maximize the coverage of the data space (breadth first). Other approaches put more weight on recommendations that are based on a sub set of the data that is currently being looked at (depth first). Depending on the task at hand one of these objectives will be favorable over the other. We advocate that a benchmark for recommendations be flexible enough to accommodate for multiple metrics and objectives.

The actual benchmark for recommendations as part of IDE can be designed along the lines of matrix completion problems as used by movie recommendation engines. That is, given a labeled set of "good"' recommendations it is the system's task to identify them. Being able to situationally identify such "good"' recommendations is not the only important aspect. No user wants to inspect 1,000 recommendations. Instead, like with search engines, only the top-k recommendations really matter. The best recommendations are not useful if they are displayed too late. Because of this a quality metric like Mean Average Precision (MAP), as used within the Information Retrieval community, might be the right choice to compare the quality of two recommendation systems. As before with our Interactive Performance Metric we suggest to penalize individual MAP scores for recommendations that take longer to compute than some latency threshold.

6.2 Workload

The workload for benchmarking the recommendations produced by the IDE system can be similar to the workload of Core-Set I. In fact, the recommendations should not be benchmarked in isolation as it will be important to see how the system computes them based on ongoing user interactions and how the system prioritizes the process of creating the visualizations vs. supporting the ongoing user interactions.

The challenge in creating the benchmark lies in pre-labeling the best set of recommendations during every single step of the predefined workflows. Here, the large amount of publications on how to create benchmarks for search engines that tackle a similar problem might be relevant.

6.3 Reporting

While we are potentially able to use the same Interactivity Performance metric as for the other core-sets, it is still important to provide detailed results on the quality of recommendations in form of the individual MAPs, ROC curves, etc.

7 Core-Set IV - Risk and Data Quality

The last proposed core-set of the benchmark is concerned with evaluating the capability of the system to clean and integrate data and to detect risk factors (e.g., [6, 46]). While there has been some work on creating benchmarks for certain sub-problems related to data quality (e.g., XBenchMatch [11] and RODI [34] for schema mapping or TPC-DI [36] for data integration in general), other sub-problems are still evaluated in a more ad-hoc manner. For de-duplication, people often use a few "known" data sets, such as the restaurant data set in [25, 47], but to the best of our knowledge no standardized benchmarking suite exist. Similarly, there is no good benchmark for "data wrangling" [18], which is surprising given the importance it has for data scientists.

Furthermore, there has been very little work helping the user avoid common pitfalls during the analysis process. For example, visualizations can be misleading as they sometimes hide certain details. When analyzing the age of the patients in the MIMIC-II dataset we observed that a significant number of patients were between the ages of $0 - 20$-years-old. We originally believed that this is normal as kids, especially infants, are quite often

in the emergency room. Only later we discovered, that many records which had an age of 0 since many doctors apparently use 0 if the age is unknown (like the previous mentioned -1 in Figure 1).

Similarly, high-level statistics can be misleading (e.g., resulting in the infamous Simpson Paradox), or recommendations can be extremely dangerous if not done with care. For example, in a recent paper [3] we showed that visual recommendation or correlation finders can significantly increase the risk of finding insights, which just appear by chance [2] (i.e., caused by the multi-hypothesis problem). In fact, interactive data exploration tools, even without recommendations, increase the chance of false discoveries as they enable the user to test many more hypothesis than ever before.

As part of the last core-set we plan to evaluate the system's capability to clean and integrate data as well as its ability protect users from making (common) mistakes. For example, we envision that as part of the benchmark requirements, the system lists its data integration capabilities and the types of common mistakes, the system tries to protect the user from. A more advanced benchmark design could contain certain types of data quality issues and potential pitfalls (e.g., a Simpson Paradox). That way systems could be evaluated in how many of these problems are detected as part of a workflow or how many system's operations were needed to correct the data error. This requires a very careful design of the data generator and might even entail adjustments to the attribute distributions based on the scale factor. Another idea is, that the system is tested with risk-control on and off as part of the Core-Sets I-III. This way the computational overhead of these protections can be factored in and measured.

However, the feasibility of such a benchmark for Risk and Data Quality control has yet to be determined and many questions remain unanswered.

8 Evaluating the User Experience

Evaluating the effectiveness of user interface designs, information visualizations and interaction techniques for data exploration is an ongoing area of research [4, 30]. On one hand, the composition of a set of tasks, e.g., load a dataset, plot the distribution of some attribute and filter based on another attribute, can favor one tool over another [35]. This bias is introduced through different factors in the complexity, the design, and the usability of the software, such as menu design, interaction techniques and defaults like color encoding. On the other hand, it is hard to conduct "fair" evaluations because tasks used in laboratory user studies don't normally reflect real-world tasks executed by domain experts. The tasks often need to be shortened and simplified so they can be accomplished in a given amount of time. However, simple tasks might lead to fewer discoveries.

In order to compare design choices within the same system, Munzer et al. [30] presented a model that defines various levels of design, each with its corresponding evaluation methodology. TouchViz [10], for instance, conducted a controlled study that compared the effect of two different interaction paradigms (gestural vs. *W*indows-*I*cons-*M*enus-*P*ointer) by measuring task completion times and task accuracy. In cases where these measures are not relevant (e.g., open-ended data exploration), like [31], we advocate for an open-ended protocol where researchers observe what insights domain experts gain while exploring the data in a self-directed manner over the course of a long-term study.

9 Conclusions and Future Work

In this paper, we presented challenges and initial ideas towards a benchmark for IDE systems. We divided the benchmark into four Core-Sets: (I) Interactive Visual Analytics, (II) Interactive Model Building, (III) Recommendations, and (IV) Risk and Data Quality. For each Core-Set we proposed a potential workload and a metric. As part of [9] we already started to develop an initial benchmark for Core-Set I and currently work on releasing that part of the benchmark publicly. In addition, we are actively looking for collaborators to design a concrete

benchmark for the other Core-Sets. We further hope to achieve a standardized benchmarking suite like the TPC-benchmarks, which have been driving innovation for years and define a whole industry.

References

[1] Airline dataset. `http://www.stat.purdue.edu/~sguha/rhipe/doc/html/airline.html`. Accessed: 2016-10-17.

[2] C. Binnig, L. De Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Towards sustainable insights, or why polygamy is bad for you. In *To appear in CIDR*, 2017.

[3] C. Binnig, L. De Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Towards sustainable insights or why polygamy is bad for you. In *CIDR 2017*, 2017.

[4] S. Carpendale. Evaluating information visualizations. In *Information Visualization*, pages 19–45. Springer, 2008.

[5] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *SIGMOD 2016*, pages 1–15. ACM, 2016.

[6] Y. Chung, M. L. Mortensen, C. Binnig, and T. Kraska. Estimating the impact of unknown unknowns on aggregate query results. In *SIGMOD 2016*, pages 861–876, 2016.

[7] A. Crotty, A. Galakatos, K. Dursun, T. Kraska, C. Binnig, U. Çetintemel, and S. Zdonik. An architecture for compiling udf-centric workflows. *PVLDB*, 8(12):1466–1477, 2015.

[8] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8:2024–2027, 2015.

[9] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. The case for interactive data exploration accelerators (IDEAs). In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 11. ACM, 2016.

[10] S. M. Drucker, D. Fisher, R. Sadana, J. Herron, et al. Touchviz: a case study comparing two interfaces for data analytics on tablets. In *SIGCHI 2013*, pages 2301–2310.

[11] F. Duchateau, Z. Bellahsene, and E. Hunt. Xbenchmatch: a benchmark for XML schema matching tools. In *VLDB 2007*, pages 1318–1321.

[12] M. El-Hindi, Z. Zhao, C. Binnig, and T. Kraska. Vistrees: fast indexes for interactive data exploration. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA@SIGMOD 2016*, page 5, 2016.

[13] H. Guo, S. R. Gomez, C. Ziemkiewicz, and D. H. Laidlaw. A case study using visualization interaction logs and insight metrics to understand how analysts arrive at insights. *IEEE Trans. Vis. Comput. Graph.*, 22:51–60, 2016.

[14] P. Hanrahan. Analytic database technologies for a new kind of user: the data enthusiast. In *SIGMOD 2012*, pages 577–578.

[15] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *Queue*, 10:30, 2012.

[16] P. Jayachandran, K. Tunga, N. Kamat, and A. Nandi. Combining user interaction, speculative query execution and sampling in the dice system. *VLDB 2014*, 7:1697–1700.

[17] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *ICDE 2014*, pages 472–483.

[18] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, 10(4):271–288, 2011.

[19] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *SIGCHI 2011*, pages 3363–3372.

[20] M.-T. Ke, S. Fujimoto, and T. Imai. Seedb: a simple and morphology-preserving optical clearing agent for neuronal circuit reconstruction. *Nature Neuroscience*, 16:1154–1161, 2013.

[21] D. A. Keim. Information visualization and visual data mining. *IEEE Trans. Vis. Comput. Graph.*, 8(1):1–8, 2002.

[22] A. Kemper and T. Neumann. Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. In *ICDE 2011*, pages 195–206.

[23] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *VLDB 2015*, 8:521–532.

[24] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan. Mlbase: A distributed machine-learning system. In *CIDR*, volume 1, pages 2–1, 2013.

[25] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, T. Kraska, T. Milo, and E. Wu. Sampleclean: Fast and reliable analytics on dirty data. *IEEE Data Eng. Bull.*, 38(3):59–75, 2015.

[26] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Trans. Vis. Comput. Graph.*, 20:2122–2131, 2014.

[27] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, volume 32, pages 421–430. Wiley Online Library, 2013.

[28] P. Marcel and E. Negre. A survey of query recommendation techniques for data warehouse exploration. In *Actes des 7èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne, EDA 2011*, pages 119–134.

[29] Mimic-iii, a freely accessible critical care database. `https://mimic.physionet.org/`. Accessed: 2016-10-17.

[30] T. Munzner. A nested model for visualization design and validation. *IEEE Trans. Vis. Comput. Graph.*, 15:921–928, 2009.

[31] C. North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26:6–9, 2006.

[32] S. Oeltze, H. Doleisch, H. Hauser, and G. Weber. Interactive visual analysis of scientific data. In *Presentation at IEEE VisWeek 2012*, 2012.

[33] Paxata. `http://www.paxata.com`. Accessed: 2016-10-17.

[34] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, W. May, D. Ritze, M. G. Skjæveland, A. Solimando, and E. Kharlamov. RODI: A benchmark for automatic mapping generation in relational-to-ontology data integration. In *European Semantic Web Conference, ESWC 2015*, pages 21–37.

[35] C. Plaisant. The challenge of information visualization evaluation. In *ACM Working Conference on Advanced Visual Interfaces 2014*, pages 109–116.

[36] M. Poess, T. Rabl, and B. Caufield. TPC-DI: the first industry benchmark for data integration. *PVLDB*, 7(13):1367–1378, 2014.

[37] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[38] B. Shneiderman. Response time and display rate in human performance with computers. *ACM Computing Surveys (CSUR)*, 16(3):265–285, 1984.

[39] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *ACM AVI Workshop on Beyond Time and Errors: novel evaluation methods for information visualization 2006*, pages 1–7.

[40] E. R. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, and T. Kraska. Automating model search for large scale machine learning. In *ACM SoCC 2015*, pages 368–380.

[41] E. R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. E. Gonzalez, M. J. Franklin, M. I. Jordan, and T. Kraska. MLI: an API for Distributed Machine Learning. In *ICDM 2013*, pages 1187–1192, 2013.

[42] M. K. Stern and J. H. Johnson. Just noticeable difference. *Corsini Encyclopedia of Psychology*, 2010.

[43] TPC-DS. `http://www.tpc.org/tpcds/`, 2016. Accessed: 2016-10-17.

[44] TPC-H. `http://www.tpc.org/tpch/`, 2016. Accessed: 2016-10-17.

[45] Trifacta. `http://www.trifacta.com`. Accessed: 2016-10-17.

[46] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar. Crowdsourced enumeration queries. In *ICDE 2013*, pages 673–684, 2013.

[47] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

Runtime Support for Human-in-the-Loop Feature Engineering Systems

Michael R. Anderson
University of Michigan
mrander@umich.edu

Dolan Antenucci
University of Michigan
dol@umich.edu

Michael Cafarella
University of Michigan
michjc@umich.edu

Abstract

A machine learning system is only as good as its features, the representative properties of a phenomenon that are used as input to a machine learning algorithm. Developing features, or feature engineering, can be a lengthy, human-in-the-loop process, requiring many time-consuming cycles of writing feature code, extracting features from raw data, and model training. However, many opportunities exist to improve this workflow, such as assisting feature code development and speeding up feature extraction.

In this paper, we discuss two projects that take different approaches to accelerate feature engineering, allowing the engineer to spend more time doing what humans do best: applying domain insight to engineer high-impact features. ZOMBIE *is a general-purpose system that reduces the amount of time needed to extract features.* RACCOONDB *is a system that helps users quickly extract features for nowcasting, or using social media to estimate real-world phenomenon. We also discuss several opportunities for future research that would improve the experience and output of feature engineers.*

1 Introduction

Many of today's most interesting software systems depend on machine learning algorithms trained on large datasets; systems like search engines, recommendation systems, and image recognition systems have become drivers of both technological and economic growth. While the specific details of machine learning systems may differ, there is often one common thread: a real-world phenomenon is represented by a set of *features*. Features aim to capture properties relevant to the chosen learning task. For example, the number of times the phrase "lost my job" appears on Twitter in a given week may be a good feature for predicting unemployment levels; a count of the political figures mentioned on a Web page may be a helpful feature to classify it into a category like politics instead of sports; and a patient's weight and blood pressure may be indicators of a risk for heart disease.

The quality of the features is paramount: a machine learning system will ultimately be only as good as its features [2, 20]. That is, if the features fail to sufficiently capture the variation among specific examples of the target phenomenon, the accuracy of the system will suffer. Historically, features have been written by humans, taking advantage of experts to impart domain knowledge to trained models. More recently, deep learning methods have been successful at automatically discovering good features, especially in problem domains that are challenging for humans, such as image and signal processing [13, 18, 35].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

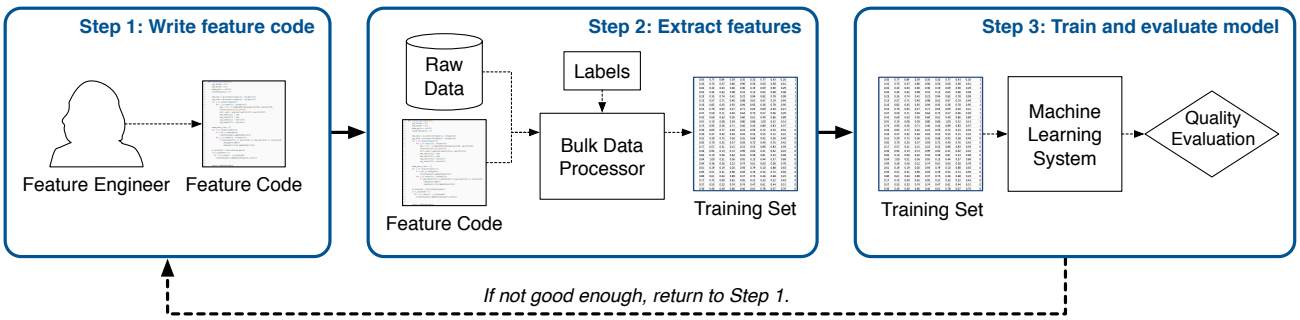


Figure 1: Feature engineering loop.

While both approaches are important to the field of data science [25], in this paper we will focus on systems designed to support the lengthy and often tedious cycle of **human-in-the-loop feature engineering**, which has been a growing focus of research in the database and systems communities [2–5, 8, 30, 34, 40]. Feature engineering is an iterative, data-centric process that we have described in detail in previous work [2, 3], and it has a general workflow followed by many different projects (e.g., [5, 21, 22, 49]).

1.1 Human-in-the-Loop Feature Engineering

In this paper, we will discuss two feature engineering projects that we have personally worked on. ZOMBIE (Section 2), a general-purpose feature engineering system, speeds up feature extraction from raw data for general supervised learning tasks, such as document classification or linear regression-based predictors. RACCOONDB (Section 3) targets a specific problem domain, speeding up the creation of features for *nowcasting models*, which estimate current values of real-world phenomena based on social media messages. Though each targets a different type of machine learning task, both involve the high-level human-in-the-loop process illustrated in Figure 1. This loop takes place while the machine learning system is being developed, before the system's final deployment. It can be described as follows:

1. **Write feature code:** The feature engineer writes code to extract features that represent salient properties of the items in a corpus of raw data. For example, when classifying Web pages, a feature may simply count the number of times a category name appears in the document or perform more complicated functions like named entity recognition using natural language processing techniques. Or, when predicting unemployment levels, a feature might be derived from topics mentioned on social media. The feature engineer may create a set of feature extractors, creating many feature values per raw data item.

2. **Extract features:** The feature code is applied to the raw data to extract the features for each raw data item, often using a bulk data processing system like MapReduce [19] or Spark [47]. When the corpus is large (e.g., a Web crawl or social media database) or when the feature code is computationally complex (e.g., involving natural language processing for document classification or comparison of time-varying signals of topic frequencies in social media), this step can take hours or longer to complete. In this step, each item is also labeled; the label may come from an existing database, be added by a human expert, or created by algorithmic means. The output of this step is a set of labeled feature vectors used to train the machine learning system.

3. **Train and evaluate the model:** Using the labeled set of examples constructed in Step 2, a machine learning system is trained with an appropriate algorithm; a naïve Bayes classifier could be chosen for our document classification example, while our unemployment predictor may use linear regression. Once trained, the model's quality—and thus the effectiveness of the features—is evaluated using an appropriate metric, usually measured over a set of test data. For our document classification example, we may measure

the percentage of test examples that were correctly classified, while for our unemployment predictor, we may use the root mean square error in the test predictions. If the quality is less than what the application requires, the feature engineer begins the process again from Step 1.

A feature engineer can go around this loop many times when developing features for a high-quality machine learning system. The impact of a particular feature to a model is very difficult to predict without actually training the model. First, because datasets are typically large and noisy, it is difficult for the engineer to know how to accurately specify a feature without repeatedly testing it. Further, within feature sets, it is difficult to know whether a new feature will bring any new information to the model [2]. Compounding the tedium, each iteration of the loop can itself take a long time to complete because each step can be lengthy: the engineer must write complicated code, apply it to a large raw dataset, and train a machine learning model. Thus, when designing a system that assists the user in human-in-the-loop feature engineering, we can speed up the engineer's workflow by either reducing the number of loop iterations or by speeding up each iteration.

1.2 Goals of a Feature Engineering System

A successful feature engineering system can take several approaches to speed up the feature engineering loop described in Section 1.1. These include:

Assistance writing feature code — Helping the feature engineer design features and write feature code can lead to fewer, shorter iterations around the loop. For example, RACCOONDB uses query expansion principles to automatically improve the applicability of feature code. Feature-focused data visualization tools can help engineers better understand data distribution and feature coverage. Development tools for data cleaning and extraction [28,49], as well as domain specific languages targeted at developing features from massive datasets [39], can reduce the time needed to write feature code. Deep learning tools can also be used to automatically create feature extractors, though often at a cost of human interpretability of the resulting features [13, 18, 35].

Faster feature extraction — Extracting features from raw data more quickly can reduce the amount of downtime a feature engineer faces. In many machine learning applications, the amount of data available is far greater than is needed to train a model well enough to evaluate feature effectiveness. For example, when using a Web crawl corpus, pages from e-commerce sites will likely provide little training information when the task is to classify documents as sports, politics, or technology. Pruning or otherwise avoiding the processing of irrelevant raw data items, as is done by both ZOMBIE and RACCOONDB, can greatly reduce the time needed to complete an iteration of the feature engineering loop.

Quicker model training — Because some types of machine learning models can take a very long time to train, speeding this up can greatly reduce the time needed to evaluate the quality of feature code. Approaches here may include scalable, parallelized algorithms [30, 36] or training a fast, approximate model that is good enough to evaluate the feature code.

Further, researchers have begun investigating several additional areas that improve the feature engineer's experience: finding and and collecting appropriate raw data from diverse sources [14]; labeling raw data through crowdsourcing or algorithmic means [21]; and automatically configuring hyperparameters in complex machine learning models [43]. Systems that continue work in these areas would be a benefit to the feature engineer.

1.3 Continuing Relevance of Database Research

Feature engineering may seem to be primarily a machine learning concern, but in fact, database research—especially systems-focused research—is highly relevant. The database community has long focused on optimizing data-centric workflows, and traditional approaches can be starting points for improving feature engineering.

When performing *query optimization*, operation execution is often dynamically ordered to reduce the number of tuples examined in a given query. In feature engineering, similar techniques may be employed to reduce the number of raw data items processed or to minimize the size of the training set used to train a model. Likewise, traditional *indexing* techniques allow for quick access to specific parts of large datasets. In feature engineering systems, methods like these can be used to quickly access parts of a large raw data corpus that have high utility to a particular learning task, allowing for faster feature extraction or training of the machine learning model.

Evaluating the effectiveness of feature code can be recast as a query over the raw data that produces a quality measure Q as a result. A high-precision value of Q may not be needed if a fast, lower-precision value can be computed, suggesting that techniques similar to those used in *approximate query answering* systems (e.g., BlinkDB [1]) may be useful. Extracting features from large datasets often involves *parallel execution* using systems like MapReduce [19] and Spark [47]. Extending these systems to include optimization techniques from parallel databases, as well as making them more supportive of machine learning workloads, as is being done with MLlib [36] and MLbase [30], will certainly accelerate feature engineering workflows.

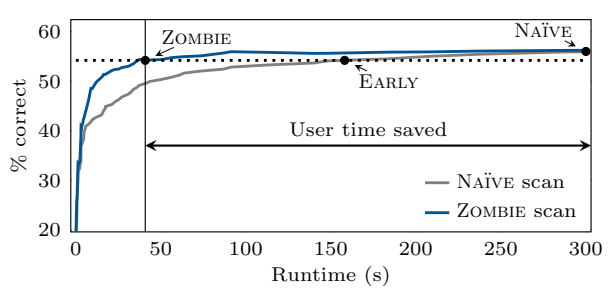
Feature extraction in many cases is an extension of *data cleaning and transformation*: raw data is often in a semi-structured format—e.g., log files, sensor output, and, to an extent, Web pages—and extracting features involves cleaning and transforming that data into a usable format. Data transformation and extraction tools such as Wrangler [28] and DeepDive [49] are currently very helpful in extracting useful data from these sources, but more specialized tools integrated into a feature engineering workflow and geared towards machine learning tasks would be a boon to feature engineers. For cases where data extraction is not straightforward, tools that assist in querying raw data or writing extraction programs using *query synthesis* (e.g, [16]) or *query expansion* (e.g, [29]) principles can speed up the feature engineer's work. Finally, *data visualization* (e.g., [45]) tools can help view raw data in terms of machine learning tasks to help feature engineers design more effective features.

In the remainder of this paper, we will discuss two projects that take different approaches to accelerate the feature engineering loop, allowing the feature engineer to spend more time doing what humans do best: applying domain insight to engineer high-impact features. In Section 2, we go over the optimizations used by ZOMBIE [3] to reduce the amount of time needed to extract features. In Section 3, we show how RACCOONDB [5] allows a user to quickly derive nowcasting features from social media. We also present an informal case study in Section 4 that further demonstrates how RACCOONDB helps users quickly estimate real-world phenomena. Finally, in Section 5, we discuss several opportunities for future research that would improving feature engineering.

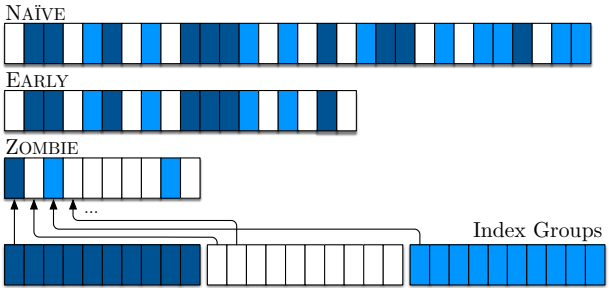
2 ZOMBIE: Accelerating Runtime through Input Selection

ZOMBIE [3] is a general-purpose feature engineering system designed to reduce the amount of time needed to evaluate the effectiveness of feature code. In particular, ZOMBIE was designed with our ***faster feature extraction*** goal from Section 1.2 in mind: it significantly shortens the time needed to complete an iteration of the loop shown in Figure 1 by reducing the time needed to apply the code to the raw data (Step 2).

Our Approach — ZOMBIE builds on ideas from query optimization, approximate query processing, and indexing to reduce overall feature code execution time through *input selection*: it dynamically learns which raw inputs are most useful to the machine learning system being trained and selects those items to be processed first. It then stops early once the machine learning system is trained adequately enough to accurately judge its quality. The resulting model is trained with only a small subset of high-impact raw data items; ZOMBIE learns which items in the corpus are likely to be redundant or irrelevant to the trained model, eliminating the need to process them. In this section, we will summarize ZOMBIE's methods and experimental results. Complete system details and results can be found in our full paper [3].



(a) Learning curves for different input selection methods. The light gray line shows a standard bulk scan of the data; ZOMBIE is shown in blue. Dots indicate stopping points.



(b) Example subsets created by different input selection methods. Each block represents a raw data item; lighter colors show more useful items.

Figure 2: Effects of input selection. ZOMBIE selects more useful raw inputs, yielding a faster learning process.

2.1 Input Selection

ZOMBIE is targeted at feature engineers building systems for supervised learning (e.g., for classification or regression). While developing features, the engineer may be willing to accept a slightly less-precise evaluation of model quality (Step 3 of Figure 1), if that quality value can be determined quickly. ZOMBIE is designed to exploit this tradeoff and generate a fast, approximate measure of the trained model's quality. To see how, consider Figure 2a, which shows the learning curves of a classification system if it were continually re-trained as new training examples are added to the training set (that is, as raw data items are processed by the feature code and labeled to produce training examples). The gray line for the NAÏVE scan shows the learning curve if the corpus is randomized, as it would be for a typical bulk scan of the data using a data processor like MapReduce or Spark. The system's accuracy increases quickly with the initial training examples, but each new example added to the training set has diminishing returns. The system could clearly stop early and get a close estimate of the model's final accuracy, represented by the point at EARLY.

ZOMBIE, however, takes this further by purposely selecting the next raw input to process, as opposed to the random draw of NAÏVE. The inputs chosen are those predicted to have a high utility towards training the model. The utility of an individual raw data item varies between learning tasks, as well as between feature code changes within the same task, making a high-quality static index identifying useful items impossible to create. Doing so would require extracting the current features from the entire corpus for analysis, which is exactly the time-consuming process we wish to avoid. Instead, in a one-time, offline pre-processing step, ZOMBIE organizes the raw data into groups of similar items called *index groups*, which are then used at runtime to identify groups of potentially useful items. ZOMBIE learns which index groups are most likely to contain useful items and selects items from those for processing. Through our experiments, we found that standard clustering methods, like the k-means algorithm, created general purpose index groups that were useful across a range of learning tasks.

The input selection process results in a trained model using just a small subset of the available raw corpus. This is illustrated in Figure 2b, which shows three different input selection methods. NAÏVE is a simple bulk scan over the entire raw data set, whose items are depicted by the multi-colored blocks, where more lightly colored blocks represent raw data items that are more useful to the learning task. The EARLY method uses the same early stopping method as ZOMBIE, but it is run using the same random layout as the full NAÏVE bulk scan. A representative subset of high- and low-quality items are processed. ZOMBIE, on the other hand, draws items from the index groups shown at the bottom of the figure, which have been clustered into groups with different utility values.[1] Once ZOMBIE learns which index groups contain the most useful items, raw data items are drawn

[1]In practice, the index groups are not so cleanly separated, but the method is still successful if there is even a small difference in average utility between the groups.

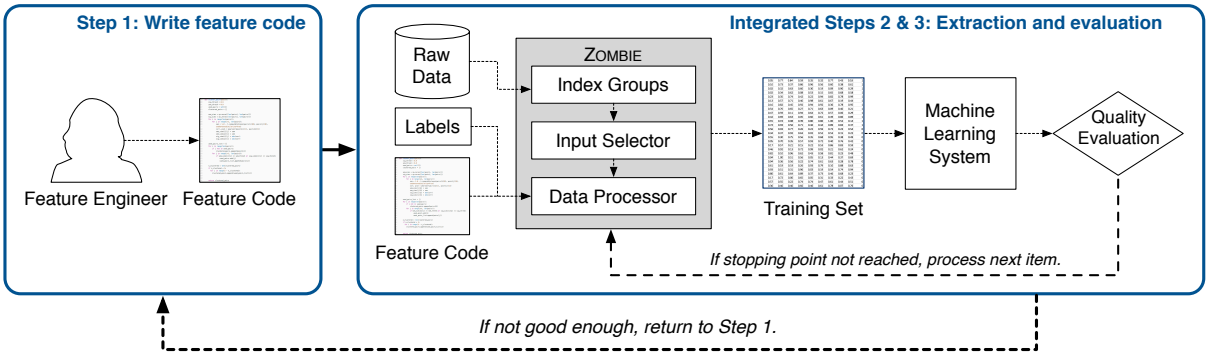


Figure 3: ZOMBIE's basic architecture, which combines Steps 2 and 3 of the feature engineering loop in Figure 1. The system's novel components are the physical index of index groups and the bandit-based input selector. These parts work with standard machine learning components to reduce the time needed to generate features from raw data.

from them, allowing the machine learning model to be trained using much less data.

Because the utility values of the index groups are unknown at the start of each iteration of the feature engineering loop, ZOMBIE faces a classic exploration versus exploitation tradeoff when trying to find the highest-utility index groups in order to maximize the processing of useful items. Thus at runtime, ZOMBIE uses a multi-armed bandit approach to determine which index groups contain the highest-utility raw data items. ZOMBIE draws items from these groups to process with the feature code to generate a new training example, retraining the machine learning model with each addition to the training set. Once the trained model has reached an appropriate stopping point (determined by observing a plateauing of the learning curve), the system stops.

2.2 System Design

Figure 3 shows the basic architecture of ZOMBIE, which combines Steps 2 and 3 of the feature engineering loop in Figure 1. When the system is initialized, the raw data is organized into a task-independent set of index groups, created using a general clustering method appropriate for the data. We create an inverted index $\mathcal{I}$, where the keys are unique identifiers for each group and each key's posting list is an unordered set of raw data items. ZOMBIE's input selector uses a standard upper confidence bound (UCB) multi-armed bandit strategy [11] to explore the index groups in $\mathcal{I}$ and exploit the ones determined to have the highest utility to the learning task. The feature functions are applied to each item selected for processing to create a feature vector for the training set, which is provided to the machine learning algorithm to generate a model. A quality function Q evaluates the model and maintains statistics on the model's improvement with each new item. The output of Q is used as the reward to update the input selector's multi-armed bandit, and the process is repeated. This output is also monitored to detect a plateauing of the learning curve, at which point the processing is stopped and the final quality measure is presented to the feature engineer.

2.3 Summary of Experimental Results

ZOMBIE was tested on several different machine learning tasks, using feature functions of varying complexity. Compared to a baseline method of a random bulk scan of the raw data that used the same plateau-based early stopping method, ZOMBIE showed an average of nearly 2x speedup for regression tasks and up to an 8x speedup for classification tasks (Table 3), meaning that each iteration of the feature evaluation loop was significantly shorter when using ZOMBIE. Compared to a full, single-processor bulk scan, as might be done in practice, ZOMBIE yielded up to a 90x speedup. For each of the tasks in Table 3, we used the same raw data corpus (1 million Wikipedia pages),

Task	Speedup
2-class classify	5.5x
6-class classify	7.8x
Regression	1.7x

Table 3: ZOMBIE's speedup results vs. a bulk scan with early stopping on several machine learning tasks.

index groups (k-means clusters), and feature set (counts of 40 specific words in each document). The range in speedup values highlights how the applicability of a set of index groups can vary between learning tasks. Our index groups represented the variability in the token-based features well, so when the features were highly relevant to the learning task, ZOMBIE could quickly find the high-utility raw data items, as it did in the document classification tasks. The feature set was less relevant for the regression task, so there were few actual high-utility items, making it difficult for ZOMBIE to provide more than a modest speedup. Still, the processing time was cut nearly in half compared to the early stopping baseline. For detailed experimental results, including evaluation of index group creation methods and design choices for the multi-armed bandit, see our full paper [3].

2.4 Related Work

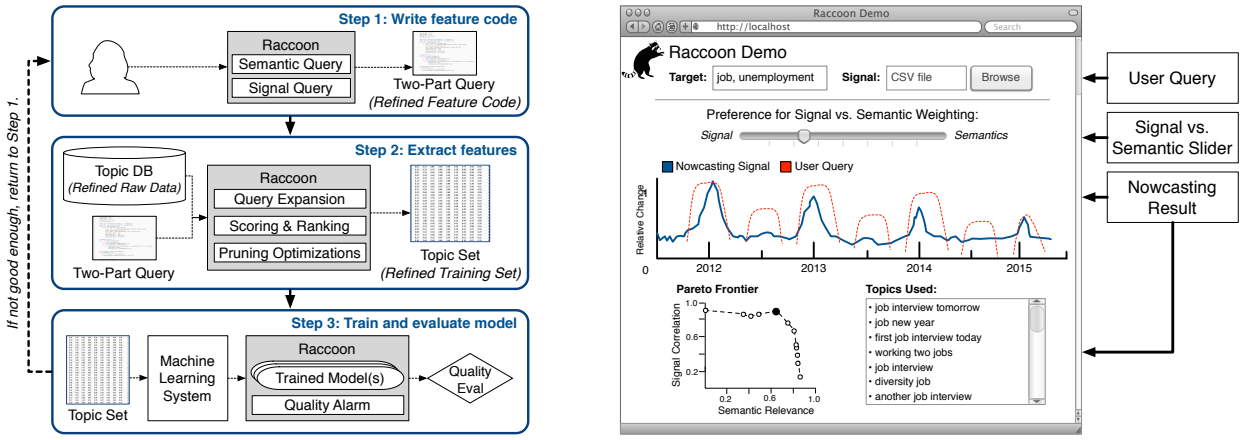
ZOMBIE [3] draws on work from a diverse set of fields. Feature engineering and feature selection have been the focus of some recent research in the database community [2, 30, 48]. ZOMBIE's goal of approximating the feature quality metric is similar to approximate query answering systems [1,12,23]. The practice of selecting the inputs for a training set based on their predicted utility is related to active learning [42], though in active learning the goal is typically to minimize the amount of labels needed to generate, based on a pre-existing feature set. We have also presented a demonstration development environment that uses ZOMBIE's input selection method [4].

3 RACCOONDB: Finding Features in Social Media

RACCOONDB [5] is a feature engineering system we designed for a specific problem domain: generating *nowcasting* features for economic and social science models. Nowcasting is the process of extracting trends from social media to generate a time-varying signal that accurately describes and quantifies a real-world phenomenon. For example, the weekly trend of US unemployment claims can be modeled using the frequency of tweets about unemployment-related topics [7]. Nowcasting has been used to model flu activity [24], mortgage refinancing [10], and more [17, 41, 46]. These models, however, have been typically created as one-off and highly manual endeavors, with researchers tediously searching through a social media corpus to find topics that are appropriate proxies for their target phenomena. By focusing on this specific problem, RACCOONDB is able to be more aggressive than a general feature engineering system could be with improving the nowcasting feature engineering loop.

The process used to created these models follows the feature engineering loop in Figure 1: the researcher writes code or a query—based on either textual or time series matching—to specify her desired topics (Step 1). Next, she executes the code or query over a large social media corpus to generate a training set based on these topics (Step 2). Finally, she uses the training set to build a model of the target phenomenon, and then evaluates its predictiveness using a holdout dataset (Step 3). This process is repeated many times by the researcher, in part because the social media corpus can be very diverse, making it difficult for the researcher to know precisely which topics to select. And because the corpus can be very large (over 150 million distinct topics in our experiments), each iteration can be lengthy, taking dozens of minutes to many hours, depending on the system used.

Our Approach — RACCOONDB targets two of the goals of feature engineering systems we discussed in Section 1.2: ***assistance writing feature code*** and ***faster feature extraction***. First, we used query expansion techniques to assist users in writing queries (i.e., "feature code") that find relevant topics, even when their initial queries inadequately describe their target phenomena. Second, we used a number of query optimization techniques to assist with faster feature extraction, allowing queries to run in interactive time and users to quickly modify their queries to improve their results. In the remainder of this section, we will briefly describe the design of RACCOONDB and present several experimental results. For an in-depth discussion of nowcasting and RACCOONDB's design and technical evaluation, see our full paper [5].



(a) Feature engineering with RACCOONDB

(b) RACCOONDB user interface

Figure 4: In Step 1 of (a), users provide *refined* feature code in the form of a two-part query using the interface in (b). In Step 2, RACCOONDB generates a set of topics that serve as a *refined* training set. In Step 3, RACCOONDB generates several results, which the user explores in real-time with the slider interface in (b).

3.1 System Design

RACCOONDB is a system that allows a user to query a database of *topics* extracted from social media messages to produce a nowcasting result. Each topic represents a collection of similar messages (e.g., those discussing the loss of a job) and contains both a textual label (e.g., "job loss") and a time-varying signal with the frequency these messages appear (e.g., daily). It is this set of tuples that RACCOONDB uses to answer nowcasting queries.

Assistance Writing Feature Code — Figure 4a shows how RACCOONDB fits in the feature engineering loop. First, the user writes a two-part query (Step 1) that declaratively describes her target phenomenon both semantically—as a text string—and as a time-varying signal (Figure 4b). This can be considered a refined version of the feature code in Figure 1. Ideally, this signal would represent historical ground truth data for the target phenomenon, but when this is not available, the user can use her domain knowledge to describe this in a distant supervision manner [37]. For example, if targeting box office sales, she may indicate peaks during the summer when blockbuster movies are generally released. Because users may not have perfect knowledge about the target, RACCOONDB employs query expansion principles during feature extraction (Step 2) to expand the semantic part of the query and allows partial time series to be input as the signal part. This makes RACCOONDB more robust to error in the nowcasting query, as our quality experiments in Section 3.2 show.

Further, during evaluation (Step 3), RACCOONDB assists users with evaluating their models in two ways. Since users may have more confidence in either of their semantic or signal query components, multiple models are trained, each weighting the query components differently. During evaluation, users can explore these models' results in real time with a signal vs. semantic weighting slider (Figure 4b). Additionally, for users who lack ground truth data and provide a query signal that only roughly estimates their target phenomena, RACCOONDB features a *query quality alarm*. This alarm uses a model trained on a set of labeled low- and high-quality queries, alerting users if the model classifies their query as low quality.

Faster Feature Extraction — Processing this two-part, computationally intensive query over a huge corpus of social media can take a significant amount of time if done naïvely. For instance, our experiments in Section 3.2 show that popular data systems like Apache Spark and PostgreSQL took dozens of minutes to many hours to query 150 million topics. Our goal in designing RACCOONDB was to return results in interactive time to allow users to quickly hone their nowcasting results. We accomplished this goal (with queries returned often in about

one second) through a number of optimization techniques (Step 2).

First, RACCOONDB uses *semantic-based candidate pruning* to avoid the expensive scoring of many topics. This works by first semantically expanding the user's query with a thesaurus-based method. Each word in each topic (as well as the query) is expanded into a set of synonyms, and a similarity score is computed based on the overlap of the expanded sets of words. This score is used to rank candidate topics for potential inclusion in the query answer. Computing this score for every topic is very expensive, so RACCOONDB prunes the candidates by analyzing the graph of synonym sets in the thesaurus, as well as by dynamically computing a threshold score below which candidates are excluded. In practice, this can reduce query computation time by over 90%.

Second, RACCOONDB uses *signal-based candidate pruning* to further avoid fully scoring many topics. To do this, RACCOONDB employs confidence interval pruning to limit the amount of expensive correlation computations it performs. Correlations are first calculated using a very low-resolution version of the topic and query signals (20 data points, rather than nearly 200). Using the standard 95% confidence interval for Pearson correlations, we eliminate all topics whose upper confidence bound was less than the lower confidence bound of the k-th highest ranked topic based on the low-resolution correlation. The full, expensive correlation was calculated for the remaining topics. When combined with our semantic-based pruning, this further reduces runtimes to interactive speeds (averaging 3.6 seconds with 1 core and 1.2 seconds with 30 cores in our experiments).

Finally, RACCOONDB takes advantage of several *low-level optimizations*. Semantic similarity and signal correlations are implemented as vector operations, allowing the system to use SIMD operations. Signal and topic label data is stored using compact integer representations, speeding up lookup times. The topic database is partitioned, allowing many calculations to be highly parallelized.

3.2 Experimental Results

We performed a number of experiments that tested RACCOONDB's output quality and runtime performance. In this section, we will briefly summarize the runtime results and present a novel evaluation of querying RACCOONDB with less than perfect information. The experiments center around estimating several different target phenomena (box office sales, flu activity, etc.), which we chose because of the availability of historical ground truth data and their use in past nowcasting research. The topics we used were n-grams of up to four words in length extracted from 40 billion tweets collected over four years. An in-depth discussion of these experiments and others is available in our full paper [5].

	PostgreSQL	Spark	RACDB
1 core	$>$ 6 h	–	3.6 s
30 cores	–	1173 s	1.2 s

Table 4: Average nowcasting query processing times for PostgreSQL, Apache Spark, and RACCOONDB.

One of the goals of RACCOONDB is to process nowcasting queries in interactive time—something we have found traditional data processing systems are unable to do. Table 4 compares the query processing times for RACCOONDB and two popular data systems (PostgreSQL and Apache Spark), using two different levels of parallelization. We averaged runtimes across our six target phenomena. Because PostgreSQL and Spark are very inefficient at nowcasting query processing, there runtimes are quite poor. In contrast, RACCOONDB is able to process queries orders of magnitude faster, achieving interactive runtimes using few resources.

3.2.1 Evaluating Quality

The feature engineering loop is human-driven, and while that human may be an expert in the problem domain, she may be less informed about what constitutes quality features for a nowcasting model. When this is the case, further iterations of feature engineering are often needed to narrow in on the high-quality features. In the following experiments, we show that RACCOONDB can still produce quality results under different levels of user knowledge and with varied amounts of error in the user's query. These experiments are compared to a

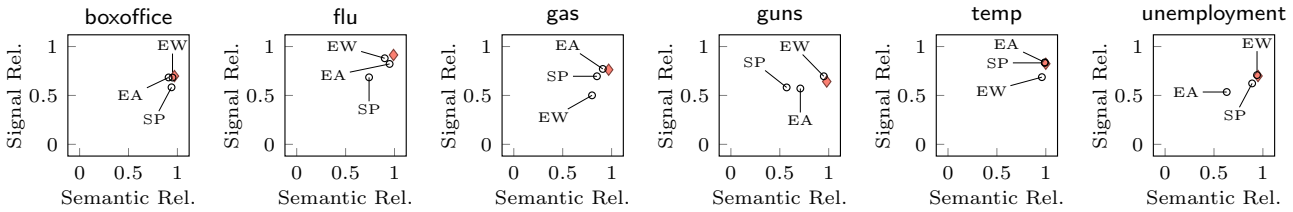
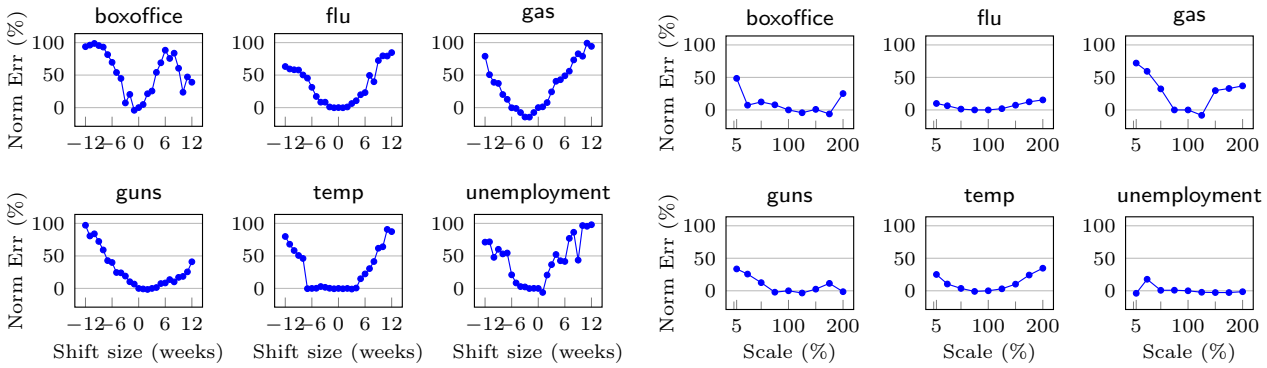


Figure 5: Our less-informed users EqWidth (EW), EqAmp (EA), and SeasonalPeaks (SP), compared to the fully-informed user (red diamond).



(a) Error from shifting user signal query peaks, relative to our distant supervision user model.

(b) RACCOONDB user interface

Figure 6: Effects of various types of error in the user signal query.

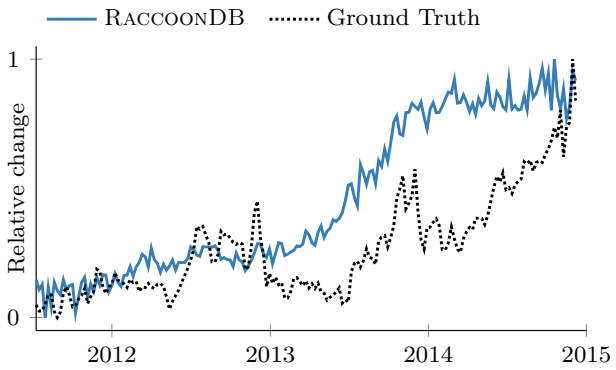
baseline *distant supervision user model*, where we assumed the user did not have access to accurate ground truth signal data and created the query signal based on a rough estimate of the target phenomenon. To simulate this, we created signals based on the actual ground truth signals by finding major peaks in the signal and synthesizing partial user signals consisting only of smooth representations of the actual peaks.

Performance of Less-Informed Users — To test how resilient RACCOONDB is to varying levels of user knowledge, we created three variants of our distant supervision user model, referred to as our *less-informed user models*. Assuming that high-quality semantic components are relatively easy for a user to construct, we focused on the more difficult part: the query signal. For the EqWidth user model, our simulated users only know when major peaks occur in the signal and not how long they last, so the synthesized peaks all have the same three-month width. For EqAmp, users have no intuition about peak height, so the synthesized peaks have the correct width, but all have equal amplitudes. Finally, for SeasonalPeaks, users only know about peaks within the last year (2014), so the synthesized peaks from 2014 are seasonally repeated over previous years (2011 - 2013).

Figure 5 shows the accuracy of our different less-informed user models using RACCOONDB. Displayed in the figure are EqWidth (EW), EqAmp (EA), and SeasonalPeaks (SP). In most cases, the less-informed user models still perform reasonably well compared to the distant supervision user model. Certain target phenomena are not as resilient to these variations though. For example, unemployment has a significant drop in semantic relevance when using the EqAmp user model. This is due to a multi-year downward trend in unemployment during this time period that reflected a general improvement in the US economy. Since EqAmp did not reflect this trend, RACCOONDB selected some unrelated topics in addition to the unemployment-related topics, obtaining a poor 0.63 semantic relevance. On average, though, the user models do not show a large decrease in accuracy: EqWidth's semantic and signal relevance decreases by just 6% and 8%; EqAmp by just 13% and 7%; and SeasonalPeaks by just 15% and 12% respectively.

MANUAL Query	Ground Truth Correlation
"solar panels"	0.13
"solar panel installation"	0.00
"solar installations"	-0.22
"solar savings"	-0.19
"solar"	0.06
"solar city"	0.30
"alternative energy"	-0.54
"photovoltaics"	-0.37
"solar energy"	-0.10
"purchase solar"	-0.16
"solar manufacturing"	-0.22
"home solar"	-0.36
RACCOONDB Query	
User signal + "solar panels"	0.73
User signal + "solar panel energy"	0.75
User signal + "solar panel installation"	0.71
User signal + "solar revenue"	0.79
User signal + "solar"	0.80

(a) User queries



Result topics	
pct solar	less with diy solar
solar pool	sunmodule solar panel
#solar #solars	deals on diy solar
now solar power	uv solar
solar wind	our solar panels
tenesol solar	green with diy solar
...	...

(b) RACCOONDBresult

Figure 7: (a) Case study queries for predicting *solar panel installations* using a MANUAL approach and RACCOONDB, along with correlations with a holdout ground truth signal; *user signal* is a user-chosen rough estimate for the target and is the quarterly revenue for Solar City Corporation. (b) RACCOONDB's result signal and topics for the "solar" query.

Resilience to User Query Errors — We also set out to demonstrate RACCOONDB's resilience to errors within a user's query. For instance, the user may have intuition about where her target trend peaks but may be off by several weeks. We introduced two types of errors to the distant supervision user model signals to simulate this. For the first, we shifted the peaks in weekly increments, between -12 and +12 weeks. Figure 6a shows how error increases with respect to a non-shifted signal. As would be expected, the result quality worsened as the shifts increased in either direction. However, the error was not catastrophic: even if a user is off by a few weeks with her estimated peaks, she could still achieve decent results. It is worth noting the behavior of boxoffice, where its results began to improve again when shifted more than six weeks into the future. This occurred due to the bimodal nature of movie box office sales, which peak in both the summer and winter months. With enough error added, the user's signal query began to accurately describe a different peak in the actual box office data.

The second error we explored is with regards to the width of each peak. We scaled the peak widths from 5% up to 200% of the original peak widths from our distant supervision user model signals (Figure 7b). Results were similar to our peak shifting experiment, where only the extreme errors resulted in a poor-quality result. RACCOONDB gives users some room for error when defining their query signals.

3.3 Related Work

The idea of RACCOONDB [5] as an optimized, general nowcasting query system draws upon our experience with designing previous nowcasting systems [7–9] and various other areas of database research: feature selection [26, 48], query optimization [45], and multi-criteria ranking [27]. We have also presented a demonstration system highlighting the user interaction of RACCOONDB [6].

4 A RACCOONDB Case Study

While our experiments have validated our technical claims for RACCOONDB, we wanted to see how useful our system is in a real-world use case, where a researcher wants to estimate a novel target phenomenon. How does the output quality and time required by the user compare to a traditional approach to nowcasting? To test this, we assigned one of our authors to be a test subject to perform such a task, while the other two authors were assigned to choose a target phenomenon, with the ground truth data for the target hidden from the test subject. Note that this study should not be confused with a formal user study; this is instead a simple, informal evaluation.

The chosen target was "US solar panel installations in the last five years," which would be evaluated against a ground truth signal of weekly photovoltaic installation counts from the Open PV Project [44]. The test subject was instructed to estimate this target using two different approaches. In the first (MANUAL), he was instructed to follow a traditional approach to nowcasting, manually searching for topics by keyword in our topic database and then use the signals of the topics found to build a model to estimate the target. In the second, he was instructed to use our RACCOONDB demonstration system [9] to estimate the target.

The test subject started the case study by spending several minutes searching online for a rough estimate of the target's trend, and he found the quarterly revenue for Solar City Corporation, which he assumed would share a similar trend with the target. Table 7a summarizes the test subject's queries for both estimation approaches. When using MANUAL to estimate the target, the test subject queried for several minutes, trying to find topic signals that matched the Solar City revenue signal, but he had very little success in his searches, averaging just 0.23 correlation (using absolute value) with the ground truth signal.[2] When using RACCOONDB over the course of a few minutes, he submitted five queries with various keywords and included the Solar City revenue signal with each query. When presented with the results, he adjusted the signal vs. semantic weighting slider to find the result with "relevant-looking" topics and a signal that "matched well" with the revenue signal. Figure 7 shows the user-chosen RACCOONDB result for his "solar" query. Across his five queries and chosen results, the results averaged 0.75 correlation with the ground truth signal.[3]

While the total time spent querying with the two approaches was not drastically different (under 10 minutes for MANUAL and under 5 minutes for RACCOONDB), the test subject was able to find a high-quality result with his first RACCOONDB query; subsequent queries were simply explorations into improving that result. In contrast, the test subject submitted many reasonable queries with MANUAL and still received very low-quality results. When processed by RACCOONDB, many of these same queries (e.g., "solar panels") returned high-quality results due to RACCOONDB's use of query expansion principles and its two-part query. This case study, while informal, provides additional confidence in RACCOONDB's real-world usage.

5 Other Opportunities

Human-in-the-loop feature engineering will likely remain an important part of building successful machine learning systems. The feature engineering loop shown in Figure 1 contains many areas that are prime opportunities for research by the database and systems communities.

Step 1 of the feature engineering loop—writing feature code—is where the feature engineer applies their domain expertise and creativity to create effective features. RACCOONDB assists the user in this area by semantically expanding the user's textual query, allowing more coverage over a diverse dataset than a less-informed user may be able to provide. Tools like FeatureInsight [15] have begun to help feature engineers visualize the effects of features on classification problems, but more advanced feature visualization systems (perhaps a visualization query system like SeeDB [45]) could help feature engineers discover subtle properties of raw data

[2] We also tried performing PCA on the chosen topics' signals and using the first factor as the target's estimate—similar to what RACCOONDB does to aggregate topics into a final estimate—but the results did not improve.

[3] Certain scientific fields will find this correlation to be quite low, but social sciences generally find anything above 0.5 to be strong.

that would make effective features. Other potential directions include incorporating automated or assisted data cleaning and transformation directly into the feature engineering workflow; integrating a data cleaning system (e.g., [28, 31, 38]) with real-time feedback from a machine learning pipeline; and applying deep learning methods [13, 18, 35] to automatically create feature extractors.

Related to feature code development is the problem of labeling training examples in datasets that are too large to feasibly label by hand. Research in distant supervision techniques [37] and label generation from noisy human-provided data [21] has shown promise. Further reducing this bottleneck would reduce the end-to-end time needed to create a successful machine learning system.

A source of significant downtime for a feature engineer is waiting for feature code to be applied to a large dataset. ZOMBIE attacks this problem by automatically finding a good subset of the data to process, while RACCOONDB takes advantage of application-specific properties of the data to aggressively prune its large corpus for each query. Other optimization approaches might include caching and shared computation [45] that take into account the evolution of feature code: feature sets may change incrementally, allowing some results from previous iterations of the feature engineering loop to be reused. On the other hand, raw data items that have the same features on one iteration of feature code may have vastly different features on another. Efficiently partitioning the raw data and parallelizing the computation under these dynamic conditions also pose interesting challenges.

Training a machine learning system can also be a lengthy part of the feature engineering workflow. Several projects have targeted training large models in a distributed fashion, such as MLlib [36] and MLbase [30, 43], as well as model selection management systems [32]. Further, determining the ideal hyperparameters for a machine learning model has seen some attention (e.g., from the MLlib team) but remains an open problem. Feature selection [26] (as opposed to feature engineering) is important to optimizing the performance of machine learning systems and has been looked at by several research groups [33, 48]. The goals of these projects are generally orthogonal to those of feature engineering; they aim to quickly train and configure the final machine learning model, rather than help the feature engineer evaluate the effectiveness of a particular feature set. Applying these optimization methods to evaluating feature effectiveness may require a different set of system requirements and novel techniques, such as developing approximate machine learning models.

6 Conclusion

Feature engineering is an import area of research that is inherently data-centric. In this paper, we presented two examples of systems aimed at improving the feature engineer's workflow by minimizing runtime and assisting with the creation of effective features from large raw datasets. The database community is in an ideal position to contribute to new feature engineering projects with its decades of experience in managing, optimizing, visualizing, and otherwise working with data.

Acknowledgements

This work, along with that of ZOMBIE and RACCOONDB, is supported by NSF grants 0903629, 1054913, 1064606, and 1131500, as well as by gifts from Yahoo! and Google.

References

[1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica, *BlinkDB: Queries with bounded errors and bounded response times on very large data*, EuroSys, 2013.

[2] Michael Anderson, Dolan Antenucci, Victor Bittorf, Matthew Burgess, Michael J Cafarella, Arun Kumar, Feng Niu, Yongjoo Park, Christopher Ré, and Ce Zhang, *Brainwash: A data system for feature engineering.*, CIDR, 2013.

[3] Michael R Anderson and Michael Cafarella, *Input selection for fast feature engineering*, ICDE, 2016.

[4] Michael R. Anderson, Michael Cafarella, Yixing Jiang, Guan Wang, and Bochun Zhang, *An integrated development environment for faster feature engineering*, PVLDB **7** (2014), no. 13, 1657–1660.

[5] Dolan Antenucci, Michael R. Anderson, and Michael Cafarella, *A declarative query processing system for nowcasting*, PVLDB **10** (2017), no. 3, 145–156.

[6] Dolan Antenucci, Michael R Anderson, Penghua Zhao, and Michael Cafarella, *A query system for social media signals*, ICDE, 2016.

[7] Dolan Antenucci, Michael Cafarella, Margaret C Levenstein, Christopher Ré, and Matthew D Shapiro, *Using social media to measure labor market flows*, Working Paper 20010, National Bureau of Economic Research, March 2014.

[8] Dolan Antenucci, Michael J Cafarella, Margaret Levenstein, Christopher Ré, and Matthew Shapiro, *Ringtail: Feature selection for easier nowcasting.*, WebDB, 2013.

[9] Dolan Antenucci, Erdong Li, Shaobo Liu, Bochun Zhang, Michael J Cafarella, and Christopher Ré, *Ringtail: A generalized nowcasting system*, PVLDB **6** (2013), no. 12, 1358–1361.

[10] Nikos Askitas, Klaus F Zimmermann, et al., *Detecting mortgage delinquencies*, Tech. report, IZA, 2011.

[11] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer, *Finite-time analysis of the multiarmed bandit problem*, Machine Learning **47** (2002), no. 2-3, 235–256.

[12] Brian Babcock, Surajit Chaudhuri, and Gautam Das, *Dynamic sample selection for approximate query processing*, SIGMOD, 2003.

[13] Yoshua Bengio, Aaron Courville, and Pascal Vincent, *Representation learning: A review and new perspectives*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013), no. 8, 1798–1828.

[14] Anant Bhardwaj, Souvik Bhattacherjee, Amit Chavan, Amol Deshpande, Aaron J Elmore, Samuel Madden, and Aditya G Parameswaran, *Datahub: Collaborative data science & dataset version management at scale*, CIDR, 2015.

[15] Michael Brooks, Saleema Amershi, Bongshin Lee, Steven M Drucker, Ashish Kapoor, and Patrice Simard, *FeatureInsight: Visual support for error-driven feature ideation in text classification*, VAST, 2015.

[16] Alvin Cheung, Armando Solar-Lezama, and Samuel Madden, *Optimizing database-backed applications with query synthesis*, ACM SIGPLAN Notices **48** (2013), no. 6, 3–14.

[17] H. Choi and H. Varian, *Predicting the present with Google Trends*, Tech. report, Google, Inc., 2011.

[18] Adam Coates, Andrew Y. Ng, and Honglak Lee, *An analysis of single-layer networks in unsupervised feature learning*, AISTATS, 2011.

[19] Jeffrey Dean and Sanjay Ghemawat, *MapReduce: Simplified data processing on large clusters*, OSDI, 2004.

[20] Pedro Domingos, *A few useful things to know about machine learning*, Communications of the ACM **55** (2012), no. 10, 78.

[21] Henry R Ehrenberg, Jaeho Shin, Alexander J Ratner, Jason A Fries, and Christopher Ré, *Data programming with DDLite: Putting humans in a different part of the loop*, HILDA, 2016.

[22] David Ferrucci, *An Overview of the DeepQA Project*, AI Magazine (2012).

[23] Minos N Garofalakis and Phillip B Gibbons, *Approximate query processing: Taming the terabytes.*, VLDB, 2001.

[24] J. Ginsberg, M. H. Mohebbi, R. Patel, L. Brammer, M. S. Smolinksi, and L. Brilliant, *Detecting influenza epidemics using search engine query data*, Nature (2009).

[25] Anthony Goldbloom, *Kaggle: The home of data science*, Extract SF, 2015.

[26] Isabelle Guyon and André Elisseeff, *An introduction to variable and feature selection*, Journal of Machine Learning Research **3** (2003), 1157–1182.

[27] Ihab F Ilyas, Walid G Aref, and Ahmed K Elmagarmid, *Supporting top-k join queries in relational databases*, VLDB Journal **13** (2004), no. 3, 207–221.

[28] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer, *Wrangler: Interactive visual specification of data transformation scripts*, SIGCHI, 2011.

[29] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu, *SnipSuggest: Context-aware autocompletion for SQL*, PLVDB **4** (2010), no. 1, 22–33.

[30] Tim Kraska, Ameet Talwalkar, John C. Duchi, Rean Griffith, Michael J. Franklin, and Michael I. Jordan, *MLbase: A distributed machine-learning system*, CIDR, 2013.

[31] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg, *ActiveClean: Interactive data cleaning for statistical modeling*, PLVDB **9** (2016), no. 12, 948–959.

[32] Arun Kumar, Robert McCann, Jeffrey Naughton, and Jignesh M Patel, *Model selection management systems: The next frontier of advanced analytics*, ACM SIGMOD Record **44** (2016), no. 4, 17–22.

[33] Arun Kumar, Jeffrey Naughton, Jignesh M Patel, and Xiaojin Zhu, *To join or not to join? thinking twice about joins before feature selection*, SIGMOD, 2016.

[34] Arun Kumar, Feng Niu, and Christopher Ré, *Hazy: Making it easier to build and maintain big-data analytics*, Communications of the ACM **56** (2013), no. 3, 40–49.

[35] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen, Jeffrey Dean, and Andrew Y. Ng, *Building high-level features using large scale unsupervised learning*, ICML, 2012.

[36] Xiangrui Meng, Joseph Bradley, B Yuvaz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D Tsai, Manish Amde, Sean Owen, et al., *MLlib: Machine learning in apache spark*, Journal of Machine Learning Research **17** (2016), no. 34, 1–7.

[37] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky, *Distant supervision for relation extraction without labeled data*, ACL-IJCNLP, 2009.

[38] John Morcos, Ziawasch Abedjan, Ihab Francis Ilyas, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker, *DataXFormer: An interactive data transformation tool*, SIGMOD, 2015.

[39] Veselin Raychev, Pavol Bielik, Martin Vechev, and Andreas Krause, *Learning programs from noisy data*, ACM SIGPLAN Notices, vol. 51, ACM, 2016, pp. 761–774.

[40] Christopher Ré, Amir Abbas Sadeghian, Zifei Shan, Jaeho Shin, Feiran Wang, Sen Wu, and Ce Zhang, *Feature engineering for knowledge base construction*, IEEE Data Engineering Bulletin **37** (2014), no. 3.

[41] Steven L Scott and Hal Varian, *Bayesian variable selection for nowcasting economic time series*, Economics of Digitization, University of Chicago Press, 2014.

[42] Burr Settles, *Active learning literature survey*, Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[43] Evan R Sparks, Ameet Talwalkar, Daniel Haas, Michael J Franklin, Michael I Jordan, and Tim Kraska, *Automating model search for large scale machine learning*, SoCC, 2015.

[44] *The Open PV Project*, `https://openpv.nrel.gov/`.

[45] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis, *SeeDB: Efficient data-driven visualization recommendations to support visual analytics*, VLDB, 2015.

[46] Lynn Wu and Erik Brynjolfsson, *The future of prediction: How google searches foreshadow housing prices and sales*, Economics of Digitization, U. of Chicago Press, 2014.

[47] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica, *Spark: Cluster computing with working sets*, HotCloud, 2010.

[48] Ce Zhang, Arun Kumar, and Christopher Ré, *Materialization optimizations for feature selection workloads*, SIGMOD, 2014.

[49] Ce Zhang, Jaeho Shin, Christopher Ré, Michael Cafarella, and Feng Niu, *Extracting databases from dark data with DeepDive*, SIGMOD, 2016.

The Values Challenge for Big Data

H. V. Jagadish
Univ. of Michigan

Abstract

As Big Data and analytics defined on top of Big Data have increasingly greater impacts on society, we humans are becoming incorporated in a Big Data loop: our activities, transactions, posts, and images, are all being recorded as Big Data; and in turn the analysis of Big Data is being used to make decisions that affect us. This paper explores characteristics of this grand loop of Big Data and begins the definition of a research agenda to address associated challenges.

1 Introduction

When we think about human in the loop for Big Data, we usually consider a human user of a database system. Most often, our concern is humans querying data in a database: How do they communicate their query intent? How do they make sense of returned results? Sometimes our concerns may go beyond these to humans who have to set up the system: How can one define a good schema? How can one integrate and clean data? How can one wrangle and load data? How can one tune the system for good performance? How usable is the database system [1]?

While all of these are important questions, and definitely worthy of study, the humans we focused on above do not include humans with perhaps the least voice: humans who generate data and are the targets of data analysis. These are human members of society at large. They are the humans in the grand loop of Big Data. Much of Big Data is obtained from their activities: consciously produced, as on social media; generated transactionally, as with credit card transactions; or recorded by third parties, as by a camera on the street. Big Data processing is all about taking these diverse sources of data, and analyzing them to produce value for the processor of Big Data. But this value is realized through advertising to the human, selecting for a job, or providing credit. All of these are activities that impact the human who was the source of the data in the first place. That is the grand loop of Big Data.

Big Data is commonly considered to pose challenges along four axes: Volume, Velocity, Variety, and Veracity. (Some sources consider only the first three of these four). While these are dimensions of technical challenge, there is also a socio-technical challenge, in terms of Values. By this, we do not mean Value (e.g. to the implementor of a Big Data system); rather we mean our Values as a society, and what impact Big Data has on these values. Ultimately, Big Data cannot be successful if this Values challenge is not adequately addressed. As such, this axis is perhaps even more important than the others. Furthermore, humans in the Big Data loop are indeed impacted by Volume, Variety, etc, but humans are centrally in the grand loop of the Big Data Values axis.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

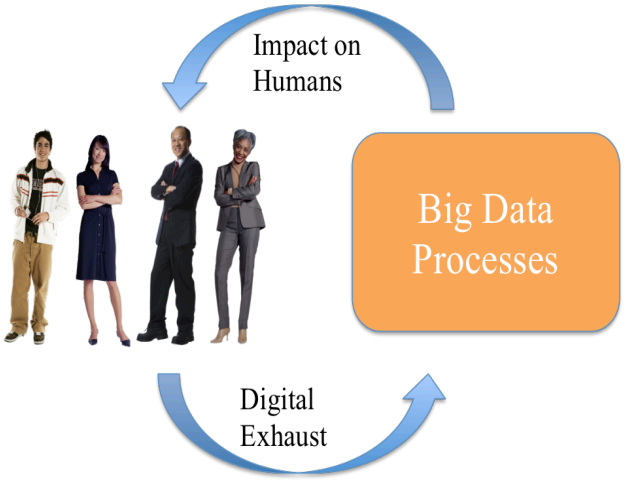


Figure 1: The Grand Loop of Big Data

In this paper, we describe some of the challenges in this grand loop. Naturally, the first challenges that come to mind are those of privacy and anonymity. We consider these in Sections 2 and 3 respectively. Issues of transparency are considered in Section 4. Somewhat surprisingly to many computer scientists, Big Data also has a significant validity challenge. This challenge is considered in Section 5. Many Data Scientists assume that the data are neutral and speak for themselves. In Sections 6 and 7, we discuss how the biases of data scientists can be reflected in the supposedly neutral results . Finally, there is a social impact of our work on Big Data, impact that we cannot ignore, as discussed in Section 8. We finally conclude with a short code of ethics in Section 9.

2 Privacy

Privacy is of course the first thing that anyone thinks of in the context of Big Data. And with good reason. There is no question that the whole point of Big Data is to suck in a great deal of information about us, to put together data from multiple sources, to analyze all this data, and thereby to generate insights that someone can use in their interactions with us: to sell something better or determine credit terms, for example [2].

Unfortunately, there are too many even today, and especially in the tech world, who do not appreciate the importance of privacy. Scott McNealy famously said [3], "You have zero privacy anyway. Get over it." Indeed, there are many who wish fondly for the old days in a small town where everyone knew everyone, and everyone also knew everything about everyone. There was a strong social fabric, and low privacy was a cost paid to achieve that strong social fabric. In big cities, we get privacy, but a much weaker social fabric. However, giving up privacy online does not cause a strong social fabric to be created. We end up with no privacy, and still the same, presumably weak, social fabric.

Why should someone care about privacy if they are not doing anything wrong? Are there things you do that you do not want to discuss with someone close to you, like a spouse or a parent? Imagine how it would feel if you did not have the option of not discussing these things. Even worse, imagine that you do not have the option of not having to explain yourself to everyone in the world. How conformist a life would you be compelled to lead? Privacy is the cornerstone of our freedom to be ourselves [4, 5].

Two other common misconceptions about privacy are worth correcting. Privacy and security are often discussed in the same breath. Those in the know tell us all the time that these are two different things: the former is

a policy question regarding user data, and the latter is an implementation question regarding a computer system. As such, most discussions of security are at the system level, and do not consider data privacy explicitly; and vice versa. However, security breaches can lead to privacy violations. The two concepts, though distinct, are related. A problem that deserves greater attention is how to minimize privacy loss when the inevitable security breach occurs. This is a security-aware direction for data privacy research, focused on issues like data representation, very different from the current body of work on privacy-preserving analysis, which assumes only legitimate access.

A second misconception is that privacy is all or nothing. Once a thought has escaped my brain and stated to a second party, then some may think that ends my privacy rights regarding that thought. But in practice, we do share secrets with confidantes, and we expect them not to share the secrets further. The secret can remain private even after it has been shared within a select group of people. In fact, the right way to think about privacy is as control over information sharing [6]. This motivates the need to control information sharing or at least detect unauthorized sharing. For some simple cases, we have work in digital watermarking. But there is need for much more work to provide the needed assurances.

3 Anonymity

One important use case for sensitive data is to find patterns in the aggregate. A classic case here is medical research: individual patient health records are sensitive, but important progress in life-saving treatments can result from analyzing the aggregates. The traditional solution, even encoded in regulations today, is to use de-identified data. Typically, this is understood to mean removal of several specific sensitive attributes, such as name, address, and social security number. However, it is usually possible to re-identify the patient from the de-identified data with these fields removed. [7]. Similar exploits are well-known in the case of AOL [8] and Netflix [9]. In the medical scenario, astonishingly, genetic sequence data can be retained in the de-identified information. Clearly, a patient can uniquely be identified if we have their entire genome sequence, or even significant parts of it. But for many medical questions, access to the genome is essential. So a stricter rule, prohibiting access to the genome, would prevent addressing many important questions.

Computer scientists have studied privacy-preserving data mining for quite a while now [10]. Techniques such as k-anonymity and l-diversity have even found their way into some curricula. However, these simple techniques do not always work to protect private information. Differential privacy [11] has recently received considerable attention as a technique that can provably obfuscate the presence of any individual record in an aggregated data set. The basic idea is to add a carefully calibrated amount of noise into any release of (aggregated) data. The basic idea works very well, if the aggregate is queried exactly once. The theory gets much more involved, and the practical value diminishes (in terms of more noise to be added to achieve obfuscation to some level) if multiple queries are allowed.

Even with its current limitations, differential privacy works for a common privacy setting with private individual records and public analysis of aggregates. But there are many scenarios where we do need individual information, such as for marketing, for employment and credit decisions, for shipping goods, for providing mobile phone service. Appropriate models for privacy in these arenas are still open. For example, there is now beginning to be good work in obfuscating movement traces with location tracking [12]. But any obfuscation also relies on some model of expected use to ensure minimal loss of value to the analysis due to the noise added.

4 Transparency

How can a decision-maker trust the results of Big Data analysis? There is a long pipeline, with many steps, in many of which some technical expert has made decisions that could impact the final result. Furthermore, some steps may involve manipulations that are just too hard to explain. For example, most deep learning algorithms

are in this class. There is considerable work on algorithmic explanation [13]. There is certainly a great deal of work on data provenance [14]. But what we need is not just an explanation of what happened at any one step. Rather, we need to understand the entire Big Data pipeline. Workflow provenance [15] ideas are a good starting point, but require considerable extensions to be able to work with the Big Data pipeline.

It may be even more important for an affected individual, than the decision-maker, to trust the results of Big Data analysis. Difficulties can arise not only because of errors in the algorithms and data processing, but also because of errors in the source data. An instance of this problem that we have a great deal of experience with, in the US and several other countries, is the notion of a credit score. An analysis conducted by the US government itself, indicates that approximately 20% of us have at least one unwarranted negative 'credit affecting' report on file [16]. In addition, of course, almost all of us have some negative information on file that we wish were not there. To address these difficulties, there is a process, cumbersome and bureaucratic though it may be, to try to correct errors in credit reports. More important, the Fair Credit Reporting Act in the U.S. requires 'lenders', very broadly defined, to provide explanations when they deny credit, employment, or other benefits related to good credit. So lenders, and similar companies, do have systems in place to explain denials, but these systems leave a great deal to be desired, and are a fertile area for further research.

One additional consideration that can complicate matters is the proprietary nature of many models. If a company is too transparent, an adversary may find it easier to reverse engineer their model. This gives rise to the open problem of how to reveal information to customer that they care about without revealing anything (or at least as little as possible) about the model used.

5 Validity

Far too often today, computer scientists, with the tools to hammer away at Big Data, arrive at conclusions that are invalid, for a variety of possible reasons. While few statisticians, at least consciously, will make these mistakes, we see them all the time in our intense desire to get meaning out of Big Data.

A basic problem with much of Big Data analysis is the Streetlight effect [17]. In traditional analysis, data are collected for the explicit purpose of interest. There is a notion of completeness, or sample representativeness. With Big Data, most of the data we work with are repurposed – they may have been created for some purpose, but are used to throw light on some other purpose. For example, tweets have become an important source of data for many analyses (cf. [18]). Yet, we know that people with Twitter accounts are not representative of the population, and even among them, the actual tweets may not be representative sample of what they think. (E.g. I may have a clear preference for a presidential candidate, but may choose not to tweet about it).

Another problem with Big Data is that we often find patterns without understanding why. However, we know that correlation does not imply causality. What this means is that we can have a small change in the environment or the system being characterized, and this can be enough to cause any algorithmic deduction to fail. Perhaps the best known example of this problem is Google Flu [19].

Another potential problem with Big Data is that there often are a very large number of attributes or features that can be derived. Given enough possibilities, there are likely to be a few features that just happen to be correlated with the result we are trying to predict. Statisticians know this as the multiple hypothesis testing problem. Given a finite set of hypotheses, there are statistical techniques to correct for their multiplicity, but these are not often used by Big Data practitioners. Furthermore, large data sets help to ameliorate this problem by reducing the probability of chance correlations. But we often do not have a finite set of hypotheses, or at least not a count of them. The reason is that there are so many different ways in which models can be set up, data can be prepared, and features can be selected. It is commonplace to perform exploratory data analysis first to develop hypotheses, which are then tested on the very same data. Furthermore, these hypotheses, or prediction models, are sometimes iteratively tuned, compounding the difficulty. The consequence is that when we think we have learned something from Big Data, we actually may or may not have learned something that's true.

6 Fairness

People who work with data will often insist that they are just doing what the data tells them: that they are completely neutral and merely exposing the truths in the data they analyzed. Unfortunately, this is never the case. Data scientists make a myriad choices when constructing a model, and these choices can reflect unstated assumptions and unconscious biases even when the data scientist is trying her best to be neutral. Furthermore, there are additional choices to be made in the data collected, the population sampled, the attributes examined, and even the way the results are presented. [20]

There have recently been numerous accounts in the press about data-driven discrimination. [22, 23]. As there is growing recognition of these issues, a small sub-community has developed around issues of algorithmic fairness. There is now an annual workshop series on Fairness and Algorithmic Transparency in Machine Learning [21].

The first question, of course, is how we define fairness. All of us, at one time or another, have complained about how life is unfair. If algorithms are to be fair, we have to capture this subjective notion as a mathematical constraint. One simple notion is to require that algorithms not explicitly consider sensitive attributes, such as race or sex. In fact, there usually are laws preventing such consideration.

However, this notion is far too minimal. Even before we had Big Data, we had the concept of surrogate attributes used for redlining. When lenders were prevented from discriminating on the grounds of race, they began to discriminate based on zip code, which could serve as a proxy for race in a segregated society. With Big Data, similar correlations can become much easier to find, and also much more subtle (for example, involving a combination of correlates, rather than just one). Even when the algorithm-designer has no desire to discriminate, the algorithm may learn correlations, and produce discriminatory results [22, 23].

A commonly adopted notion of fairness, at the individual level, is that any two individuals with similar final outcomes get treated similarly. For example, if we predict something about creditworthiness or employee performance, that these predictions are not systematically biased for particular values of a sensitive attribute, such as race.

Individual fairness is hard to measure, and hence hard to guarantee, even though there is good work on techniques to achieve it (cf. [24]). A much simpler notion of fairness is simply to look at outcomes. In the selected set, e.g. of people considered creditworthy or employable, is the distribution of values for a sensitive attribute, such as race, the same as in the population as a whole? This is the notion we have in mind when we use the term under-represented. The goal of affirmative action is increase the representation of under-represented groups. Algorithms can be designed to accomplish this, if desired.

The prevalent wisdom today is that fairness is a constraint that imposes a cost on algorithm performance. Mathematically, this may appear obvious. While this notion of constraint may be true for group fairness, further thought should make clear that individual fairness should not be a cost: we do want to make the best possible prediction for each individual, without regard to race, sex, etc. If we find that our algorithm output is consistently biased for any value of sensitive attribute, then we should be able to improve the algorithm by stratifying the population based on this sensitive attribute, and separately correcting for the bias in each stratum.

While individual notions of fairness are best dealt with through algorithm design, group notions of fairness are more difficult to handle purely within a scoring and classification algorithm. It would appear that we can profit from using set-oriented ideas commonly used in database management.

7 Diversity

Diversity is much lauded for the benefits it provides, in education settings, in work groups, and elsewhere [25]. If diversity is desired, then it can be worked into a dataset, or the algorithm, as a goal [26]. Just as with fairness, there are many different mathematical definitions of diversity: one has to choose the most appropriate

one for one's particular circumstance. Irrespective of the specific definition chosen, we note that diversity is a group concept: diversity measures apply to a set and not to an individual item. The challenge is that most algorithms are scoring or labeling individuals, so it is not easy to introduce a desired property of the result set when determining score or label for an individual. In fact, even the definition of result *set* is not straightforward: for university admissions it may be the yearly batch, for a company hiring it may be a moving window over some period, and so on.

A completely different aspect of diversity to consider is that human decision-makers are usually diverse, while the same decision-making algorithms may be used widely. Given many independent human decision-makers, not all will make the same decision. For example, given the same set of candidates, different human experts will likely differ on the best choice. Also, given the same circumstances, different judges will likely impose somewhat different sentences for a crime. In contrast, copies of an algorithm will all behave identically. Understanding and managing this lack of diversity is important. In some cases, such as criminal sentencing, the greater standardization may even lead to benefits. However, in many situations we may actually prefer diversity. It is an open research question how to build this diversity into algorithm design. We may be able to draw inspiration from extensive existing work on diverse result sets for queries.

8 Social Impact

We live in a world that evolves. We need our algorithms to evolve with us. Unfortunately, our algorithms are trained on Big Data from the past. (We do not have a choice, since we do not have training data from the future). We use these past data to make decisions about the future. The unstated assumption is that the future will look like the past. When this is not true, dependence on algorithms can lead to *ossification*, where the algorithms, like old curmudgeons, are resisting change. For example, consider a company that puts into place new programs to make their workplace friendlier to women than it was in the past. Such a company is obviously looking to change its employee make up, increasing the number of women employees; yet, its Big Data driven employee success prediction algorithm working on past data, may depress hiring of women by scoring their potential for success in the old company rather than the one with the new programs in place.

Another aspect of Big Data to consider is that addressing Big Data problems requires big resources. In many cases, only big companies may be able to afford these resources. In consequence, the benefits of Big Data may flow only to a few, at the expense of smaller companies and individual citizens. These issues are not well characterized, but underlie much of the worry that many feel about Big Data [27]. But technological progress need not result in such concentration of power. For example, technological innovations such as cloud computing, have made unprecedented computing resources available to all. The field is open for innovation in this dimension. An initial beginning has been made by efforts such as [28].

9 Conclusions

As we consider humans in the Big Data loop, perhaps the most important dimension of Big Data we must address is *Values*. Big Data is a powerful force with huge impacts on society. These impacts can be both good and bad. We need to work to make sure we get as much of the good with as little of the bad as possible.

Towards this end, I have proposed a code of conduct, with two simple rules:

- Do not surprise. Of course, the result of analysis can be surprising. In fact, we often look for surprises. But the process itself should not be a surprise. An individual should not be surprised with the manner of analysis carried out with data about them.
- Own the outcome. As technologists, we cannot have blinders on and disavow responsibility for the data we manage, the results we produce by analyzing this data, or what others do with our Big Data results.

We must teach this code of conduct, and the ethical framework to follow it, to every student of Data Science. To assist the community to do this, I have prepared a modular online course [29] that is available on the web with a Creative Commons license. I encourage you to use material from it in your classes as appropriate.

We must also initiate a body of research into the responsible use of data, and the technologies that assist us in such responsible use. A recent workshop [30] has started to develop a community around this topic.

This work was supported in part by the National Science Foundation, under grant IIS-1250880.

References

[1] H. V. Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, Cong Yu. Making database systems usable. *SIGMOD Conference*, Beijing, China, 2007, pp. 13–24.

[2] Bruce Schnier. *Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World*. WW Norton and Company, 2015.

[3] Stephen Manes. Private Lives? Not Ours! *PC World* 18 (6): 312, April 18, 2000.

[4] Louis Brandeis, Samuel Warren. The Right To Privacy *Harvard Law Review* IV(5), Dec.15, 1890

[5] Daniel Solove. A Taxonomy of Privacy. 154 *U. Pennsylvania Law Review* 477, 2006.

[6] Helen Nissenbaum. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford University Press, 2009.

[7] Khaled El Emam, Sam Rodgers, and Bradley Malin. Anonymizing and Sharing Patient-Level Data. *BMJ* 350:h1139, 2015.

[8] Michael Barbaro,Tom Zeller Jr. A Face Is Exposed for AOL Searcher No. 4417749. *The New York Times* Aug. 9, 2006. www.nytimes.com/2006/08/09/technology/09aol.html,

[9] A. Narayanan, V. Shmatikov. Myths and Fallacies of "Personally Identifiable Information". *CACM* 53(6): 24-26, June 2010.

[10] Jaideep Vaidya. *Privacy Preserving Data Mining*. Springer-Verlag Berlin, Heidelberg, 2009.

[11] C. Dwork, A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9(3-4):211-407, 2014 .

[12] John Krumm. A Survey of Computational Location Privacy. *Personal and Ubiquitous Computing* 13(6): 391-399, 2009.

[13] Bryce Goodman,Seth Flaxman. European Union regulations on algorithmic decision-making and a right to explanation. https://arxiv.org/pdf/1606.08813.pdf, 31Aug 2016.

[14] James Cheney, Laura Chiticariu, Wang-Chiew Tan. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases* 1(4):379-474, 2007.

[15] Susan Davidson, Sarah Cohen-Boulakia, Anat Eyal, Bertram Ludascher, Timothy McPhillips, Shawn Bowers, Manish Kumar Anand, Juliana Freire. Provenance in Scientific Workflow Systems. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 30(4): 44-50, 2007.

[16] Federal Trade Commission. In FTC Study, Five Percent of Consumers Had Errors on Their Credit Reports That Could Result in Less Favorable Terms for Loans. https://www.ftc.gov/news-events/press-releases/2013/02/ftc-study-five-percent-consumers-had-errors-their-credit-reports.

[17] David Freedman. Why Scientific Studies Are So Often Wrong: The Streetlight Effect. *Discover* Magazine, July-August 2010, http://discovermagazine.com/2010/jul-aug/29-why-scientific-studies-often-wrong-streetlight-effect

[18] D Antenucci, MJ Cafarella, MC Levenstein, C Re, M Shapiro. Using Social Media to Measure Labor Market Flows. NBER Working Paper No. 20010, issued Mar 2014. See http://econprediction.eecs.umich.edu/

[19] David Lazer, Ryan Kennedy, Gary King, Alessandro Vespignani. The Parable of Google Flu: Traps in Big Data Analysis. *Science* 343(6176): 1203-1205, 14 Mar 2014.

[20] Solon Barocas, Andrew D. Selbst Big Data's Disparate Impact 104 *California Law Review* 671, 2016.

[21] Fairness, Accountability, and Transparency in Machine Learning. See http://www.fatml.org.

[22] Julia Angwin, Jeff Larson, Surya Mattu, Lauren Kirchner. Machine Bias. *ProPublica*, May 23, 2016. See https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[23] H. V. Jagadish. Why We are Hard on Amazon and Should Be. Blog post at http://www.bigdatadialog.com/fairness/why-we-are-hard-on-amazon-and-should-be. Aug 19, 2016.

[24] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* pages 214-226, 2012.

[25] Scott Page. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton University Press, 2007.

[26] Marina Drosou, Evaggelia Pitoura. Multiple Radii DisC Diversity: Result Diversification Based on Dissimilarity and Coverage. *ACM Transactions on Database Systems* 40(1): 4, Mar 2015.

[27] Cathy O'Neil. Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy. Crown, 2016.

[28] Data Science for Social Good. See http://dssg.uchicago.edu/.

[29] H V Jagadish. *Data Science Ethics*. EdX MOOC at https://www.edx.org/course/data-science-ethics-michiganx-ds101x, 2016.

[30] Serge Abiteboul, Gerome Miklau,Julia Stoyanovich, Gerhard Weikum. Data, Responsibly. *Dagstuhl Seminar Proceedings*, vol. 16291, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2016.

HILDA 2016 Workshop: A Report

Arnab Nandi
The Ohio State University
arnab@cse.osu.edu

Alan Fekete
University of Sydney
alan.fekete@sydney.edu.au

Carsten Binnig
Brown University
carsten_binnig@brown.edu

Abstract

The first annual workshop "Human-in-the-Loop Data Analytics" (HILDA) was held on June 26, 2016 in association with ACM SIGMOD in San Francisco. The workshop sparked some excellent conversations between speakers and attendees from many disciplines spanning both industry and academia, and covered a variety of sub-themes related to human-in-the-loop data challenges.

1 Introduction

An often overlooked – but critical – part of the data management life cycle is the human-in-the-loop. The key focus of the HILDA workshop was to evaluate, understand, and improve the participation of humans in data management, with the eventual goal towards building optimized data management systems and techniques that treat humans as first-class citizens, alongside data. We aimed to bring together those in the database research community who are paying attention to the distinctive characteristics of people which impact the ways data management activities occur, and to reach out to other communities such as visualization, data mining, human-computer interaction as they grapple with data-related challenges.

HILDA 2016 accepted 16 papers [1] from 32 submissions. 8 of these were presented as longer talks, and others as short "taster" talks; all were also presented as posters. There were 53 registered participants and attendance reached over 90 at times during the workshop. There was a genuine buzz in the room with several great interactions, especially given the format where each session ended with a panel discussion among the presenters. We were very lucky to have two outstanding keynotes. Laura Haas (IBM Research) spoke about platforms for collaboration that have been built at the Accelerated Discovery Lab. These integrate information about people, data and tools, and support conversation as a metaphor for mixed-initiative interactions. Jeffrey Heer (University of Washington) spoke about declarative languages to describe (and thus allow automated generation and optimization of) data transformation and interactive presentation pipelines.

2 Research Themes

The presented papers spanned a multitude of facets of the HILDA theme, ranging from novel user interfaces, to infrastructural support for interactive analytics, to interactive data preparation.

User Interfaces: The usability of a data management system is hugely influenced by the nature of the interface

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

through which the users perform tasks such as express their queries, define the schema, tune for performance, and observe results. A significant part of the design towards supporting interactive analytics involves innovation in the visualization infrastructure. A tool for recommending visualizations was presented, as well as visual interfaces for particular tasks such as provenance, cluster adjustment, and identification of exceptional cases in data curation. There was also interest in conversational interfaces where a data platform and one or more users work together over multiple data manipulation steps through a (restricted) natural language.

Interactive Analytics: A second theme discussed during the workshop was centered around the question of how to design database systems to achieve the interactivity of analytical tasks – a crucial component towards accelerating human involvement. Papers tackled three important areas to achieve this goal: co-design of database system internals, interfaces, and visualizations; enabling ad-hoc analytics over new data sources by avoiding expensive data preparation and indexing costs; working with machine learning models and results (e.g., trends) in an interactive fashion.

One critical point raised during the workshop was that while enabling interactivity enables better and more widespread use of advanced analytical tools, it also significantly increases the risk of making spurious discoveries. Therefore, quantifying the risk is of utmost importance for the next-generation of interactive data exploration systems.

Data Cleaning, Extraction, and Labeling: Another key area where the involvement of humans was highlighted was that of bringing data into a structured form for downstream processing. In contrast to traditional data ingestion where data pipelines are considered a "one-time" effort, there is a growing understanding that this is typically an iterative and interactive process. Issues such as iterative data cleaning, supervised extraction, and guided data preparation are becoming quite common. Beyond traditional databases methods towards enabling humans to programmatically label data in machine learning pipelines was also discussed.

3 Bridging Communities

One major success of the workshop was the bringing together of people from the visualization, human-computer interaction, data mining, and data management communities. We were fortunate to see presentations and participation from students, researchers, and industry representatives from outside core data management areas. The trans-disciplinary interest in the area is bolstered by the presence of complementary workshops, such as the "Data Systems for Interactive Analysis" workshop at IEEE VIS, and the "Interactive Data Exploration and Analytics" workshop at ACM SIGKDD.

Industry Involvement: HILDA 2016 was supported with sponsorship from diverse companies interested in this new area of data analytics: IBM, Paxata, Tableau, and Trifacta. We are especially grateful to Paxata for providing scholarships for seven students to attend the workshop and the SIGMOD conference. There was also a strong industry presence in the program committee, in the paper submissions, and in discussions in the room.

Thoughts for the future: We are happy to announce that the 2nd annual HILDA 2017 workshop will be held at ACM SIGMOD 2017 in Raleigh, NC, USA on May 14, 2017. We encourage submissions to the workshop. More information is available at the HILDA website, `http://hilda.io`.

References

[1] C Binnig, A Fekete, A Nandi. *Proceedings of the First Annual Workshop on Human-in-the-Loop Data Analytics.* HILDA 2016.

Data Engineering

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name _____

IEEE Member # _____

Mailing Address _____

_____

Country _____

Email _____

Phone _____

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du
Key Laboratory of Data Engineering and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398