Perspectives on Infrastructure for Crowdsourcing

Omar Alonso
Microsoft Corp.
1065 La Avenida, Mountain View, CA
omar.alonso@microsoft.com

ABSTRACT

Human computation has gained a lot of interest lately as a paradigm for solving problems that computers can't do yet. Crowdsourcing marketplaces are showing that is possible to get tasks completed in time with a modest budget in a wide range of applications. However, the infrastructure that is currently available for supporting crowdsourcing tasks is very limited. There is non-trivial extra work that the experimenter has to do to get successful results. Consequently, the developer tends to spend more time dealing with the complexity and operational aspects of the task than on its design.

Given the mix of different subareas of computer science that are potentially involved, defining and designing such platform presents a number of interesting problems. In this paper, I outline the rise of human computation, summarize the limitations of a popular platform, and present a perspective on the topic that can be viewed as a starting point to debate requirements for this new research area.

Keywords

Human computation, crowdsourcing, infrastructure, experiment design

1. INTRODUCTION

Human computation is a new research area that studies the process of channeling the vast Internet population to perform tasks or provide data towards solving problems that no known efficient computer algorithms can yet solve. There are two main variations of human processing: games with a purpose and crowdsourcing marketplaces.

Games with a purpose (e.g., the ESP Game) specifically target online gamers who, in the process of playing an enjoyable game, generate useful data (e.g., image tags). There are quite a bit of game prototypes for a wide range of tasks that follow the same structure: you play a nice game while providing ("labeling") good data. A crowdsourcing marketplace is a human computation application that coordinates workers to perform tasks in exchange for rewards (usually money).

In the last couple of years, there has been a growing research interest on leveraging human computation for a broad range of tasks such as relevance evaluation, machine translation, natural language processing, etc. Besides workshops in academic venues like data mining (KDD) and information retrieval (SIGIR), this area is getting industrial traction as well with the first crowdsourcing conference (CrowdConf).

Current research on this emerging area focuses mainly on two fronts: design of games with a purpose and improving the quality of work in certain domains. Little work has been done regarding infrastructure for supporting this type of research.

Borrowing terminology from cloud computing, we can think of human computation as an *elastic human workforce*. Similar to a cloud computing platform that supports utility CPU computing, it should be possible to support utility HPU (human processing unit) computing.

In this paper, I am interested in tasks that a developer would like to run *continuously* (in a production environment) over extended periods of time in an enterprise environment rather than proof of concepts. Hence, one should expect more functionality from such infrastructure.

2. USING MTURK IN PRODUCTION

Amazon Mechanical Turk (MTurk) is probably the biggest commercial crowdsourcing platform available today, where people perform thousands of tasks on a given day. One reason on the adoption of human computation is due to the popularity of MTurk that makes it easier and affordable to conduct experimentation. Traditionally, the process of recruiting users for conducting studies in computer science is expensive and extremely time consuming, making the experiment setup very costly.

Competition in this area is growing rapidly. There are a number of startups trying to provide similar functionality (CloudCrowd) or implement wrappers around MTurk (CrowdFlower) and similar services. Metaweb is another example of a human computation platform with a pool of known workers [10] in contrast to anonymous MTurk workers.

One can perform an analysis of previous experiments with crowdsourcing and identify a number of commonalities among published research. There are four main beneficial properties of using crowdsourcing: speed, cost, quality, and diversity:

1. Experiments go very *fast*, usually getting results in less than 24hrs.
2. Running experiments is usually *cheap*. You can pay a few cents per task and end up spending $25 for the whole experiment.
3. The output is usually of good *quality*. This doesn't mean that there is no need to deal with spammers but

WSDM 2011 Workshop on Crowdsourcing for Search and Data Mining (CSDM 2011), Hong Kong, China, Feb. 9, 2011

with some quality control mechanisms in place, one can yield good results.

4. *Diversity* of workers is good.

Besides the current adoption, MTurk has several drawbacks and has received numerous criticisms about spam handling and the lack of mechanisms to review both workers and requesters. Ipeirotis has pointed out the many things that need to get fixed in MTurk [9] and there is no intention to describe them again here.

Apart from the well-known limitations, the service works and has an important user base. It also has a key feature that is very important in the case of incentives: *payments*. Transferring money around in the world is a tricky business and requires process and legislation in place. A worker wants to get paid on time and the requester wants a pay a minimum with the option to include other monetary incentives.

MTurk, from a HPU view, works reasonably well like a matching site for requesters and workers. Work gets done and workers get paid with some exceptions (see [13] for a report from the field). Other features like analytics are non-existing so all data analysis has to be performed off-line using other tools.

In summary, the requester/developer ends up building a number of tools around the platform to make sure the experiments are successful. That is fine for a first generation of such platforms but more needs to be done to make crowdsourcing more successful and available to wide adoption.

3. AN INFRASTUCTURE PERSPECTIVE

Let's explore the idea of an alternative platform for crowdsourcing. At first, it would make sense to look at this platform from the following landscape: the human view (the workers), the experimenter view (the task designer), and the engine view (the common software features). For each view, I present a scenario and describe a list of desirable features.

3.1 The human view

The end user view (the worker, the human, the end user) is pretty straightforward to understand: easy to use interfaces that allow any human with *minimum* instructions to perform a well-defined task. This can be accomplished by using a browser or cell phone in combination with a task that looks *doable* with the right incentive. Like in every crowdsourcing task, we are interested in grabbing attention for a few minutes so that a number of tasks are completed successfully.

A naïve lecture would suggest that, after all, one only needs a form-based user interface to collect data. A closer look reveals that the designer must follow well know usability techniques for presenting tasks. The elastic human workforce is the planet so cultural and multilingual characteristic should be part of the design.

According to behavioral economics, it is important for people to see the value in the work they perform. An important incentive is obviously money but it should be possible to use other equivalents like points or reputation as currency that workers can use to see that their work is meaningful.

A major complaint from workers is the uncertainty with payments due to in part to fraudulent requesters [13]. At the same time, a major complain from requesters is that workers perform sloppy work or try to game the system to maximize profit.

Features:

- Routing/recommendation of similar tasks based on past behavior and/or content.
- Requester rating based on payment performance, rejected work, and overall task difficulty. A worker should be able to rate the quality of work and also the quality of the requester.
- Ability to comment on a task
- Work categorization. Similarly to a job search site, all work that is available should be classified.

3.2 The experimenter view

The experimenter has two main goals to fulfill: design the right task that produces the data that he/she is looking for and make it appealing to workers. The task has to grab *attention*. Without interesting content people won't complete the task and human computation would not be usable.

A key factor for the success of any task is *attrition*. The requester depends on HPU to complete tasks. So it must provide good trust mechanisms so there is always human workforce available.

The experimenter has to know how to ask the right questions so that minimize the number of instructions. Workers are not experts so it is not possible to assume the same understanding in terms of terminology and expertise.

Having quality control statistics and feedback from workers are key factors for improving the task. It is widely known that inter-coder reliability is a critical component of content analysis. When it is not established properly, the data and interpretations of the data should not be considered valid.

How to get qualified workers for a particular task and how to detect workers that are not doing a good job is an important part of quality control. It is possible to pre-qualify workers with a test or include honey pots in data sets to improve overall quality.

It is important to differentiate workers that are not *suitable* for a particular task from the *spammers* in the system.

Features:

- Ability to manage workers in different levels of expertise including spammers and potential cases.
- Abstract the task as much as possible from the quality control statistics. The developer should provide thresholds for good output.
- Ability to mix different pools of workers based on different profile and expertise levels.
- Honey-pot management and incremental qualification tests based on expertise and past performance.

3.3 The engine view

Cloud computing seems to be a natural fit for this type of system. One thing that is needed is to extend the idea of utility computing and combine CPU + HPU.

MapReduce with human computation looks like an interesting avenue to explore. Both share the following similarities:

- Large task divided into smaller sub-problems
- Work distributed among nodes (workers)
- Collect all answers and combine them

There are some variations as well, notably:

- Human response time varies
- Some tasks are not suitable for human computation
- Consensus among the results

The engine view should have the ability to integrate human-computation into a language. For example, if a developer wants to incorporate human computation to a program, he/she should be able to execute a given task using a particular data set to say a number of workers and expect an inter-agreement level that meets a particular threshold. Parameswaran and Polyzotis describe a declarative language involving human computation functions [19] and discuss the advantages versus a procedural approach.

There are number of statistics available for computing inter agreement like Cohen's kappa, Fleiss' kappa, and Krippendorff's alpha to name a few. In some cases, they may not be appropriate for a given experiment but it would be nice if the system provides such inter-rater statistics by default and the ability to plug-in your own reliability metric.

Features:

- Performance and high availability
- Spam detection built in the system
- Payments (including international markets)
- Inter-agreement statistics library and ability to plug-in a user-defined one
- Uncertainty management
- High-level language for designing tasks
- Analytics

3.4 Integration point

The point of the views discussed before is to show that is possible to think of a crowdsourcing platform on those three actors.

I believe that the power of human computation relies on designing tasks that need to be completed with CPU and HPU. The challenge is then to identify where HPU should be added in such a way that adds value to the task that needs to get done. An example of such approach is Soylent [2] that integrates crowdsourcing into a traditional software program like MS Word.

Remote procedure call is a well-know technique to transfer control and data over a communication network in the context of CPU. A similar mechanism, human procedure call, can be achieved if the platform provides consensus, reliability, and validity as part of the result aggregation for a given task.

Independently of the current offers on the market, the success of a new platform would rely on the ease of use, integration mechanism, and features that make the creation of work and analysis of results accessible to developers.

Features:

- Language model that allows easy integration with other enterprise systems.
- ETL tools

4. RELATED WORK

Related work in human computation and crowdsourcing touches several fields. I mention some of the current research that is relevant to the ideas presented so far.

The notion of human computation as a distributed system is presented in different ways in the literature. Davis et al. outline a comparison with CPU and shows examples of HPU vs. CPU [6]. Heymann and Garcia-Molina present a human programming environment and its model based on MTurk [8]. Quinn and Bederson introduce a taxonomy of distributed human computation in [12].

There is previous work on using crowdsourcing for information retrieval and natural language processing. Alonso and Mizzaro compared a single topic to TREC and found that workers were as good as the original assessors [1]. Tang and Sanderson used crowdsourcing to evaluate user preferences on spatial diversity [16]. Grady and Lease focused their work in human factors for crowdsourcing assessments [7]. Other kind of experiments in IR can be found in [4].

The NLP community has been using MTurk for different tasks. The research work by Snow et al. [14] shows the quality of workers in the context of four different NLP tasks such as affect recognition, word similarity, textual entailment, and event temporal ordering. Callison-Burch shows how Mechanical Turk can be used for evaluating machine translation [3].

A number of tools have been developed for solving some of the limitations of MTurk with TurKit being one of the most popular ones [11]. TurKontrol, a planner for controlling crowdsourced workflows, is presented in [5].

In terms of generic infrastructure, scientific workflows and databases have gotten a lot of attention in recent years that should be possible to adapt to human computation [15].

5. SUMMARY AND CONCLUSIONS

The panel on crowds, clouds, and algorithms covers similar topics to the ones presented in this paper [19]. The development of systems that include crowdsourcing and cloud computing systems will be a major driver of information technology innovation going forward.

Applications that leverage "big data" are likely to benefit from a mix of CPU and HCU. Techniques from a number of computer science areas can used to design such platforms.

From an infrastructure perspective, the main questions are:

- What are the basic software components for HPU?

- Should the underlying infrastructure be a game engine? People are very familiar with game consoles so there is no training needed. On the other hand, games only task may impose a bound on the kind of experiments one would interested to run.
- Should be part of a social networking infrastructure? After all, social networking can provide elastic workforce at any time if there is the right level of engagement.
- Is crowdsourcing for external consumption or can it be adopted on the enterprise as well? There is previous research that shows that crowdsourcing works on the enterprise [17]. Would be possible to combine workforce inside and outside of the enterprise?
- What is the database model for crowdsourcing/HPU?

Human computation is here to stay. Researchers on different areas of information technology are finding HPU an alternative way to collect data to solve problems efficiently.

There are a number of problems with current commercial platforms: they are very rudimentary, there are no tools for data analysis, no data integration with existing systems, and lack of feedback loop between workers and requesters. This creates a number of adoption problems as developers have to spend upfront considerable effort to make experiments viable. For ad-hoc experimentation this may be fine but then an organization needs to run human computation tasks on a continuum, a better service is needed.

The main research questions for this emerging area are:

- What are the tasks suitable for human computation?
- What is the best way to perform human computation?
- What is the best way to combine CPU with HPU for solving problems?
- What are the desirable integration points for a computation that involves CPU and HPU?

I presented a perspective on infrastructure for human computation and outlined challenges and opportunities having an enterprise setting on mind. The observations are made after running lots of experiments in different domains and building missing features that would make life easier.

The list is not exhaustive nor pretends to be a requirements document. The goal is mainly to open a debate on what kind of features and services a crowdsourcing platform should provide in the future. I believe that the more we crowdsource tasks over time, the more we learn about potential features and useful scenarios.

Finally, this area is attracting a lot of interest from industry and academia so I would expect a considerable amount of work in the coming years dedicated to build the next generation of crowdsourcing/human computation infrastructure.

6. REFERENCES

[1] O. Alonso and S. Mizzaro. "Can we get rid of TREC Assessors? Using Mechanical Turk for Relevance Assessment". *ACM SIGIR Workshop on the Future of IR Evaluation* (2009)

[2] M. Bernstein et al. "Soylent: A Word Processor with a Crowd Inside", *UIST* (2010)

[3] C. Callison-Burch. "Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk", *EMNLP* (2009)

[4] V. Carvalho, M. Lease and E. Yilmaz (Eds). *Proc. of the SIGIR Workshop on Crowdsourcing for Search Evaluation* (2010)

[5] P. Dai, Mausam, and D. Weld. "Decision-Theoretic of Crowd-Sourced Workflows", *AAAI* (2010)

[6] J. Davis et al. "The HPU", *IEEE ACVHL* (2010)

[7] C. Grady and M. Lease. "Crowdsourcing Document Relevance Assessment with Mechanical Turk". *NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010)

[8] P. Heymann and H. Garcia-Molina. "Human Processing", Technical Report, Stanford Info Lab (2010)

[9] P. Ipeirotis http://behind-the-enemy-lines.blogspot.com/2010/10/plea-to-amazon-fix-mechanical-turk.html

[10] S. Kochhar, S. Mazzocchi, P. Paritosh. "The Anatomy of a Large-Scale Human Computation Engine", *KDD HCOMP* (2010)

[11] G. Little et al. "TurKit: Tools for Iterative Tasks on Mechanical Turk", *KDD HCOMP* (2009)

[12] A. Quinn and B. Bederson. "A Taxonomy of Distributed Human Computation", Technical Report HCIL-2009-23, UMD, Human-Computer Interaction Lab (2009)

[13] M. Silberman et al. "Seller's problems in human computation markets", *KDD HCOMP* (2010)

[14] R. Snow et al. "Cheap and Fast But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks". *EMNLP* (2008)

[15] M. Stonebraker et al. "Requirements for Science Data Bases and SciDB", *CIDR Perspectives* (2009)

[16] J. Tang and M. Sanderson. "Evaluation and User Preference Study on Spatial Diversity", *ECIR* (2010)

[17] M. Vukovic, J. Laredo, and S. Rajagopal "Challenges and Experiences in Deploying Enterprise Crowdsourcing Service", *ICWE* (2010)

[18] Panel on "Crowds, Clouds, and Algorithms: Exploring the Human Side of 'Big Data' Applications, *SIGMOD* (2010)

[19] A. Parameswaran and N. Polyzotis. "Answering Queries using Humans, Algorithms and Databases", *CIDR* (2011)