Exploiting Twitter as a Social Channel for Human Computation

Ernesto Diaz-Aviles
diaz@L3S.de

Ricardo Kawase
kawase@L3S.de

Wolfgang Nejdl
nejdl@L3S.de

L3S Research Center / University of Hannover. Hannover, Germany

ABSTRACT

To fully leverage the innate problem solving capabilities of humans necessitates paradigm shifts towards decentralization of human computation systems, making the existence of central authorities superfluous and even impossible. In this position paper, we propose a novel decentralized architecture that exploits the Twitter social network as a communication channel for harnessing human computation. Our framework provides individuals and organizations the necessary infrastructure for human computation, facilitating human task submission, assignment and aggregation. We presented a proof of concept and explore the feasibility of our approach in the light of several use cases.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*; K.4 [**Computer and Society**]: [General]

General Terms

Human Factors; Design

Keywords

Human Computation; Social Computer; Twitter

1. INTRODUCTION

Today's most successful *crowdsourcing* services such as Amazon's Mechanical Turk[1] and CrowdFlower[2], share a common characteristic: they are all based on centralized architectures. In these services, both, users' profile information and task distribution engine are centralized.

However, the *Social Computer* vision that we share is more likely to be based on decentralized architectures [6], like the ones provided by social networks and mobile devices, where humans would interact via free information exchange or trading to solve large-scale problems, that cannot be easily addressed by conventional computer systems and algorithms.

Social networking sites such as Twitter[3] have experienced an explosion in global Internet traffic over the past years. As of June 2011, it is estimated that the Twitter users surpassed 300 million, and they generate more than 200 million of 140-character Twitter messages – *tweets* – every day [8, 9]. Interestingly enough, nearly two-thirds of active Twitter users access the microblogging service using a mobile phone [2].

The massive amount of mobile users plus the simplicity of the interactions in Twitter, together with its scalability and real-time message exchange, make this social networking system an appealing environment to assign and collect feedback for human computation tasks, which fits the nature of a tweet: short and simple.

In this work we propose *MechSwarm*, a decentralized framework for human computation built upon Twitter's infrastructure. Our primary contributions can be summarized as follows:

- We present the building blocks necessary for a decentralized crowdsourcing architecture.
- We introduce simple yet powerful idioms for human-intelligent-task assignment over the Twitter social network, which can be considered as a protocol for human computation over a transport layer.
- We present a conceptual design of our framework and identify a number of use cases for human computation, that can take advantage of the proposed approach.

The rest of this paper is structured as follows: Section 2 introduces the terminology and core concepts of the framework, as well as, its components and workflow. In Section 3, we present a conceptual design of our approach that shows its feasibility. Section 4 introduces several Use Case scenarios and presents how human intelligence tasks can be described using the *MechSwarm Task Language*. We discuss current and future issues in Section 5. Section 6 presents related work. In Section 7, we conclude the paper. Finally, Appendix A, includes basic terminology used in Twitter as a reference.

2. MECHSWARM FRAMEWORK

First, we introduce the key concepts of our proposed framework *MechSwarm*. We borrow some terminology from Amazon's Mechanical Turk [1] and extend it in order to explain our approach.

[1]**Mechanical Turk:** `mturk.com`
[2]**CrowdFlower:** `crowdflower.com`
[3]**Twitter:** `twitter.com`

CrowdSearch 2012 workshop at WWW 2012, Lyon, France.

Human Intelligence Task (HIT):

A fine-grained task such as, "Is this a picture of the Golden Gate Bridge?" or "In the video segment about sports or technology?", which can be easily performed by humans and it can be rendered as part of a Web-based user interface.

Contributor:

A human being who is willing and able to perform a HIT. Each contributor has also a Twitter account that is used to receive a HIT request and to reply with the solution. This is equivalent to the concept of *worker* according to Amazon's Mechanical Turk terms, but we rather use the term *contributor* instead, since is a more general concept, for example, volunteers which do not expect a monetary payment for performing a given HIT are also considered contributors.

Requesters:

The individuals or organizations that need a set of HITs to be done. Each requester has a Twitter account that is used to send HITs requests and receive HITs' responses.

HIT Assignment:

When a requester needs a particular HIT to be done, he uses MechSwarm to assign it to a candidate contributor that will perform the task.

More formally, we define a Human Computation system (HCOMP-system) as a triple (T, H, A), where

- T is called *problem* and corresponds to a set of Human Intelligence Tasks (HITs),
- H is a set of human candidates to perform a task t (i.e.,*contributors*), and
- $\mathcal{A} : T \rightarrow H$ is a function that assigns each task t to a human $\mathcal{A}(t) \in H$.

The solution to the problem T is denoted by $Solution(T)$. Note that this definition does not impose any restriction on where the task submission, assignment, and completion takes place.

MechSwarm provides (i) the selection of candidate contributors H, (ii) a task assignment over this set (i.e., $\mathcal{A}$) and (iii) an aggregation mechanism to compute the final solution of the problem, i.e., $Solution(T)$.

In the rest of this section we detail the different components of the framework and the system workflow.

2.1 Components and Workflow

The *MechSwarm Task Language*, the *Human Computation Optimizer* and the *HIT-Solver* are the fundamental components of the framework. They are specified as follows:

MechSwarm Task Language. The language used to specify the HITs and basic protocol for message exchange.

Human Computation Optimizer (HCO). The component that manages the HIT requests, *contributor* selection and task assignment.

HIT-Solver. The component responsible to aggregate the completed HITs and to compute a final solution.

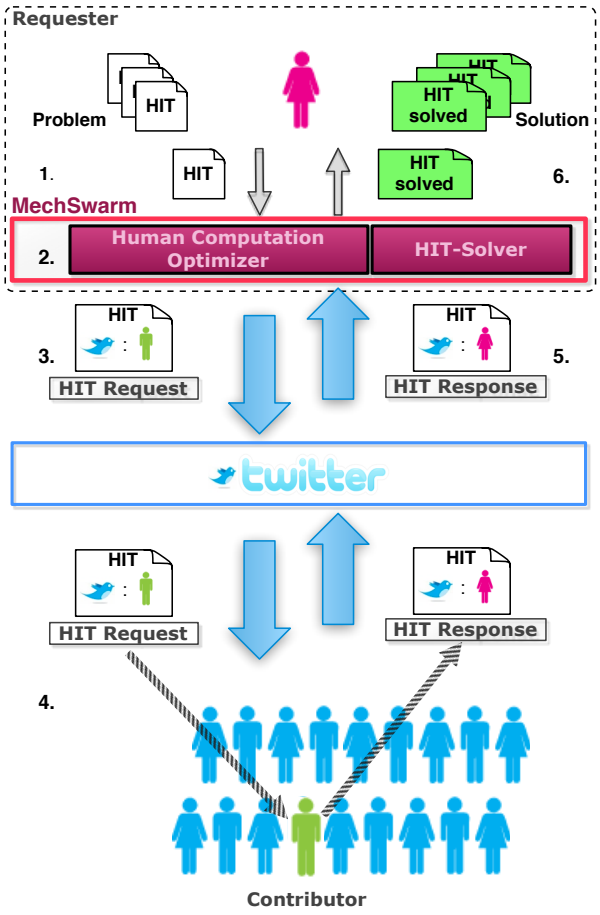

Figure 1: MechSwarm Components and Workflow. (1) A requester defines a problem (i.e., the set of HITs T). Each HIT is defined using the MechSwarm task language. (2) The requester submits the HITs to the *Human Computation Optimizer (HCO)*. The HCO identifies for each HIT a *contributor* from the requester's Twitter Social Graph (i.e., from his *followers*), and assigns him the HIT. (3) The HCO uses the Twitter social network to route the HIT request to the contributor. (4) The contributor receives the HIT, completes the assignment and issues a HIT response to the requester. (5) The *HIT-Solver* component collects the HIT responses to the specified problem on behalf of the requester, and (6) computes the problem's final solution, i.e., Solution(T).

The basic workflow of the system is shown in Figure 1. A requester begins by defining a problem (i.e., T) that can be split into several HITs easily tackled by humans. The HITs are expressed using the MechSwarm's task language. The requester submits the problem to the *Human Computation Optimizer*, which selects a set of *contributors* as candidates to solve the HITs (i.e., H), and assigns to each of them a task to perform (i.e., $\mathcal{A}$).

Each contributor completes the HIT assigned and sends back a response with the solution. The HIT-Solver collects the set of HITs completed and computes a final solution to the problem, i.e., $Solution(T)$.

Function	Character
Question ID start	#
Question ID end	?
Parameter delimiter	&
Parameter terminator	!

Table 1: List of reserved characters of MechSwarm Task Language to identify each part of the message. These reserved characters are configurable by the developer.

In Figure 1, we can observe that the HCP and HIT-Solver components run on the requester's infrastructure, and not on a centralized system. As a consequence, requesters' profile information remains private and does not need to be disclosed to third parties.

In the next section we present an instance on how to realize these concepts.

3. CONCEPTUAL DESIGN

In this section we demonstrate the feasibility of the proposed decentralized framework. We present and discuss how each component can be realized.

3.1 MechSwarm Task Language

One crucial point in distributing tasks among many contributors is to make sure that they are familiar with the chosen language (i.e., *protocol*) to communicate with the framework. If the communication channel and the communication languages are not coordinated, any human computation is in vain. To this end we propose a response formatting that is short and simple to use, is familiar to Twitter users and is customizable. The basic format is a tweet containing a Twitter *mention.* to the framework, followed by the identification of a HIT, followed by the the choices from a list of possible answers. For basic terminology used in Twitter, please refer to Appendix A.

In Table 1 we list the predefined character delimiters used in the MechSwarm Task Language. Note that the only character that is not customizable is the Twitter reserved symbol (@), used for *mentioning.* In the end, a request and response should be formatted as follows:

- Request Template:
  @<Target Contributor><Question>#<QuestionID>?
  <Choice1>&<Choice2>&...&<Choice n>!

- Response Template:
  @<MechSwarm Framework>#<QuestionID>?
  <Answer1>&<Answer2>&...&<Answer n> !

Please note that the list of choices in the request is optional. Furthermore, observe that complex and massive task definition require additional software tools, e.g., to select from a database the set of questions to be asked in a questionnaire, but the basic idioms presented in this section can even be input directly by the requesters.

The request and response length is restricted to 140 characters, given the message limit imposed by Twitter. Concrete examples of HITs, specified using the MechSwarm Task Language, can be found in Section 4.

3.2 Human Computation Optimizer (HCO)

The Human Computation Optimizer (HCO) is a core component in charge of managing HIT requests, *contributor* selection and task assignment. HCO exploits social proximity to assign HITs to contributors belonging to the requester's *social graph.*

We are exploring more sophisticated methods for contributor selection and task assignments, in particular we want to (i) automatically identify the nature and semantics of the problem (e.g., HITs), and (ii) learn and keep a contributor profile in order to optimize the task assignments according the capabilities of each contributor.

3.3 HIT-Solver

We consider each HIT as part of a problem. The solution of the problem does not only imply to solve each HIT, but also to produce an aggregated result, or a meaningful combination of the output produced by individual HITs. For example, in order to translate into Spanish a text document written in German, we could split the document into paragraphs, and then create and assign a HIT to a contributor requesting the translation of each of them. The final solution corresponds to the result of each HIT plus the ordering of the translated paragraphs, with respect to the original document.

The final step of computing the aggregated solution of a problem is performed by the HIT-Solver.

4. USE CASES

We reserve this section to expose a list of use cases (UC), encompassing several human computation tasks, that can be effectively solved using our framework. We use the Twitter account "MechSwarm" in our discussion below.

UC1: Pairwise Comparisons. The framework is prepared to listen to all tweets that mention the account (@MechSwarm) and, if required, to acknowledge the received response. Additionally, *HIT-Solver* computes the final solution based on the HIT responses received. The framework logs all responses received, in order not to processing them more than once.

Request: "@Contributor
Which one is your favorite search engine? #favSearch?Google&Yahoo!"

Response: "@MechSwarm #favSearch? Yahoo!"

UC2: Sound Verification. The framework can be used to confirm results from unsupervised methods as automatic tagging images, videos or sounds.

Request: "@Contributor
Is http:///example.com/sound.mp3 a bird? #soundHIT?yes&no!"

Response: "@MechSwarm #soundHIT? yes!"

UC3: Image Tagging Additionally, yet another application is to provide means for contributors to add correct human judged metadata to resources.

Request: "@Contributor Tag image http://example.com/picture.png #tagImageHIT?"

Response: "@MechSwarm #tagImageHIT? dog&animal&nature!"

UC4: Near Duplicate Detection For the task of video duplicate detection, the contributors could access a simple interface displaying two videos and two buttons ("yes" and "no"). Once the contributor clicks on one of the buttons this triggers his Twitter account to post the formatted message understandable by the framework.

Request: "@Contributor Are these terms/videos the same? http://example.com/V1V2/ #matchVideoHIT?yes&no!"

Response: "@MechSwarm #matchVideoHIT? yes!".

UC5: Translation The requesters can post HITs that require sentences to be translated to a certain language.

Request: "@Contributor Translate to Portuguese: Hello world #translateHIT?"

Response: "@MechSwarm #translateHIT? Ola' Mundo!".

5. DISCUSSION AND FUTURE WORK

Twitter aggregates millions of users that are interconnected through follower/followee ties. The users interactions in Twitter, using mobile devices, open the opportunity to achieve large scale human computations, similar than the ones performed in centralized crowdsourcing systems, with the additional benefits of contributor's social ties and real-time information exchange.

Regarding the monetary motivation supported by crowdsourcing systems like Amazon Mechanical Turk, we think that alternative decentralized trade spaces for human computation are possible, where rewards and incentives to individuals do not necessarily involve a monetary payment for their contributions. Clear examples exist of such spaces that support our vision, Wikipedia, for instance, can be considered as a massive human computation task of knowledge gathering, where the vast majority of contributors does not receive money for their efforts, but are motivated by intrinsic rewards that comes from work achieved itself [5]. The decentralized framework we discussed here is flexible enough to also incorporate monetary rewards if it is required, but the *contract* would be established directly by the requester and contributors, without any intermediaries.

Our work opens the door to interesting future directions. One interesting question is: how to exploit *plurality* for error-resilient HIT solving? Additionally, one issue to be examined is how the public HIT responses from one contributor influences others. It is a reasonable assumption that any suggestion or recommendation before the execution of a task may bias its outcome, thus should be empirically verified.

We plan to deploy a live implementation of our framework. We want to explore how can more complex tasks be solved using the basic idioms we proposed, is it possible to achieve the functionality provided by tools like TurKit [4] using our framework?

In particular, we are interested in use case scenario *UC1: Pairwise Comparisons*, which is at the core of learning to rank and collaborative filtering algorithms, which can be realized using a decentralized crowdsourcing workforce.

6. RELATED WORK

When talking about Human Computation, there are two main concepts that come in mind: crowdsourcing and Games With A Purpose (GWAP) [10]. Crowdsourcing is the act of gathering together the solutions performed by large groups of people over some specific task. Today the most prominent human computation application for is Amazon Mechanical Turk, a marketplace for crowdsourcing. Amazon Mechanical Turk works as a platform to coordinate (humans) to perform simple tasks that usually computers cannot, in exchange for monetary rewards.

Games With A Purpose, or human computation games, exploit the idea of having human players to compete in solving problems. Many domains have profit from the GWAP approach, mostly annotation of images and music [12, 3] and also collecting common sense facts [11, 7].

Another great example that exploits Human Computation is the reCAPTCHA project[4], which provides a *captcha* service that is primarily used to identify whether is a human accessing some online content, and at the same time, collects the feedback to correct words in digitalized books that optical character recognition (OCR) programs fail to recognize with certainty.

First, like Amazon Mechanical Turk we propose a framework that defines workflows and terminologies for modeling human computational tasks. Second, from GWAPs we share the motivational power embedded in games and social networks to leverage the distribution and completion of tasks. Lastly, like reCAPTCHA, instead of forcing users to search for tasks, we bring the tasks to the users, using the Twitter's nature of pushing notifications.

Like a crowdsourcing marketplace, our goal is to support Human Computation, but we propose a decentralized approach based on Twitter's architecture and social graph, which is quite different from the aforementioned works.

7. CONCLUSION

In this paper, we introduced a decentralized human computation framework, *MechSwarm*, that exploits Twitter's social network to harness the problem solving power of human intelligence.

The framework does not require a centralized system to manage task submission, assignment, and completion, but has the potential to empower individuals and organization to distribute tasks across large number of human contributors over Twitter's social graph. We presented a conceptual design and explore the feasibility of our approach in the light of several use cases.

We envision that decentralized architectures for human computation will emerge as viable alternatives to well established crowdsourcing services. Our approach is a small step towards realizing this vision.

[4]**reCAPTCHA:** `google.com/recaptcha`

8. REFERENCES

[1] J. Barr and L. F. Cabrera. Ai gets a brain. *Queue*, 4:24–29, May 2006.

[2] Edison Research. Twitter usage in america: 2010. `http://www.edisonresearch.com/`, 2010.

[3] E. Law, L. von Ahn, R. Dannenberg, and M. Crawford. Tagatune: a game for music and sound annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[4] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 57–66, New York, NY, USA, 2010. ACM.

[5] M. Poppendieck. Unjust desserts? *Better Software*, pages 33–47, July/August July/August 2004.

[6] D. Robertson and F. Giunchiglia. The social computer: Combining machine and human computation (DISI-10-036). Technical report, University of Trento, 2010.

[7] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems*, pages 1223–1237, 2002.

[8] C. Taylor. Social networking 'utopia' isn't coming. CNN Tech. `http://goo.gl/emF5j`, June 2011.

[9] @twittereng. 200 million tweets per day. Twitter Blog. `http://goo.gl/eybp0`, June 2011.

[10] L. von Ahn. Games with a purpose. *IEEE Computer*, 39(6):92–94, 2006.

[11] L. von Ahn, M. Kedia, and M. Blum. Verbosity: a game for collecting common-sense facts. In *CHI'06*, pages 75–78, 2006.

[12] L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *CHI'06*, pages 55–64. ACM Press, 2006.

APPENDIX

Appendix A: Basic Twitter Terminology

- **Tweet**: A message posted via Twitter containing 140 characters or fewer.
- **@**: The @ sign is used to call out usernames in Tweets.
- **Mention** A mention is any Twitter update that contains *@username* anywhere in the body of the Tweet.
- **Follower**: A follower is another Twitter user who follows a specific account.
- **Followee**: Reflects other Twitter users that a specific account chose to follow.
- **Lists**: Curated groups of other Twitter users. Used to tie specific individuals into a group on your Twitter account.
- **Reply**: A Tweet posted in reply to another user's message, usually posted by clicking the "reply" button next to their Tweet. Always begins with @username.
- **Retweet**: A Tweet by another user, forwarded by someone else.