Perspectives on Infrastructure for Crowdsourcing

Omar Alonso
Microsoft

9 February 2011

Disclaimer

The views and opinions expressed in this talk are mine and do not necessarily reflect the official policy or position of Microsoft.

Disclaimer – II

- Personal experience
  - MTurk, CrowdFlower, Internal MS tools
- IR focus
  - Relevance evaluation, assessment, ranking, query classification, etc
  - TREC, INEX, Twitter, Facebook
- Continuity
- Industry perspective

Introduction

- Crowdsourcing is hot
- Lots of interest in the research community
  - Articles showing good results
  - Workshops and tutorials (ECIR'10, SIGIR'10, NACL'10, WSDM'11, WWW'11, etc.)
  - CrowdConf
- Large companies leveraging crowdsourcing
- Start-ups
- VCs are putting money on it

Areas of interest

- Social/behavioral science
- Human factors
- Algorithms
- Databases
- Distributed systems
- Statistics

Why Mechanical Turk

- Brand (Amazon)
- Speed of experimentation
- Price
- Diversity
- Payments
- Lots of problems and missing features
  - Still, people keep using it

Pedal to the metal

- You read the papers
- You tell your boss that crowdsourcing is the way to go
- You know need to produce hundreds of Ks of labels per month
- Easy, right?

Why not Mechanical Turk

- Spam
- Worker and task quality
- No analytics
- Need to build tools around it

Alternatives?

- First mover advantage
- The service hasn’t evolved that much
- $$$
- People are trying …
  - CrowdFlower, CloudCrowd, etc.

Infrastructure thoughts

"Unfortunately, 100% of our teenage evaluators responded with 'whatever'."

The human

- As a worker
  - I hate when instructions are not clear
  - I'm not a spammer – I just don't get what you want
  - Boring task
  - A good pay is ideal but not the only condition for engagement

The human – 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

The experimenter

- As an experimenter
  - Balancing act: an experiment that would produce the right results **and** is appealing to workers
  - Attrition
  - I want your **honest** answer for the task
  - I want qualified workers and I want the system to do some of that for me

The experimenter – 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.

The system

- Similarities with MapReduce approaches
- Integration of human computation to a language
- I would like to program the crowd
- Built-in statistics and other quality control

The system – features

- Performance and high availability
- Spam detection built in
- 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

Conclusions and questions

- Social networking and crowdsourcing
- Crowds, clouds and algorithms
- 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?