Answering Queries using Humans, Databases and Algorithms

Alkis Polyzotis -- UCSC
The presentation in one slide

- Crowdsourcing is an emerging field with diverse applications.
- Crowdsourcing + Databases = a new exciting research area.
- Database principles can simplify the use of crowdsourcing.
Outline

- Crowd-sourcing and the human computer for micro-tasks.
- Existing approaches vs Our approach.
- Declarative queries over humans, databases and algorithms.
- Human-assisted graph search.
This talk is about **crowd-sourcing**

Wikipedia: "**Crowdsourcing** *is the act of outsourcing tasks ... to an undefined, large group of people* or community..."

There are several examples (past and present):

- Oxford English Dictionary (1857!)
- [Human Computers](#)
- Wikipedia
- [GWAP](#) -- Games With A Purpose
- Social Bookmarking
- [Amazon Mechanical Turk](#) -- Do microtasks for a fee.
Why are microtasks interesting?

- Complex tasks can be broken down into several microtasks.
- People are willing to solve microtasks for a fee.

A microtask is the "unit of work" for a human computer.
Properties of the human computer

- It may require monetary compensation.
- It has high latency (typically).
- It yields inconsistent/noisy answers.

Q: Is this really a computer that we want to use?

A: Human computation outperforms the most sophisticated algorithms on certain tasks.
Outline

• Crowd-sourcing and the human computer for micro-tasks.

• **Existing approaches vs Our approach.**

• Declarative queries over humans, databases and algorithms.

• Human-assisted graph search.
State-of-the-art: Programming Libraries

- Examples: TurkIT, HProbe, probably many others.

- Library enables creation and management of microtasks.

- Library also provides some useful primitives:
  - Iteration
  - Voting
Procedural Crowdsourcing is Hard!

Task: Retrieve the top-k images from a relational DB such that:
- Image displays menu item OR restaurant name
- Image is not dark
- Image is not copyrighted

Programmer has to do a lot of work:
- Implement the "plumbing" between DB and application.
- Order DB, algorhmic and human operations.
- Decide which microtasks to push to the workers.
- Decide on price of microtasks.
- Resolve inconsistencies.
Our **Vision**: Declarative Tasks.

Example: *Find all jpg pictures of a clean beach*

```
SELECT *
FROM Images I
WHERE isJPG(I) AND hbeach(I) AND hclean(I)
```

Example: *Find the "best" image of Phokion Kolaitis*

```
SELECT hbest(P.img)
FROM Person P
WHERE P.name = "Phokion Kolaitis"
```
Outline

- Crowd-sourcing and the human computer for micro-tasks.
- Existing approaches vs Our approach.
- Declarative queries over humans, databases and algorithms.
- Human-assisted graph search.
Our vision

- A task is encoded as a query.
- Application provides the UI to ask questions.
- Transparent optimizations inside the query processor
An example task/query

Find all images that are large pictures of a clean beach, or pictures of a clean and safe city

\[
\text{result}(l) := \text{img}(l) \land \text{hClean}(l) \land \text{hBeach}(l) \land \text{aLarge}(l)
\]

\[
\text{result}(l) := \text{img}(l) \land \text{hClean}(l) \land \text{haCity}(l,C) \land \text{safe}(C)
\]
Challenges in Query Optimization

\[ \text{result}(l) := \text{img}(l), \ h\text{Clean}(l), \ h\text{Beach}(l), \ a\text{Large}(l) \]

- The optimizer has to trade off among:
  - Latency to compute the answer
  - Expense to obtain answers from the crowd
  - Quality of the overall answer.

- At the same time, the following is unknown:
  - Selectivities
  - Latency for both algorithms and humans
  - Uncertainty in answers.
Choosing the right UI

The UI affects the number and type of questions asked
  ==> latency
  ==> expense
  ==> quality

The UI also affects how the final answer is derived.
Outline

- Crowd-sourcing and the human computer.
- Existing approaches vs. our approach.
- Declarative queries over humans, databases and algorithms.
- Human-assisted graph search.
The general question

*What algorithm would you develop to solve task X if you had to rely on human workers for a basic computation?*

Example: Sort a list of items, using human computation for the comparisons.

- Merge-sort, quick-sort, or something else?
- Binary or multi-way comparisons?
- How do we trade off between latency, expense and quality?
Problem: Image Categorization

**Taxonomy**

vehicle

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>nissan</td>
</tr>
</tbody>
</table>

maxima  | sentra

**Image**

![Car Image]
Solution: Ask For Directions!

1. Is it a vehicle? : YES
2. Is it a toyota? : NO
3. Is it a honda? : YES

Answers cost money
Answers have high latency
Answers have different gains
Solution: Ask For Directions!

Key problem: Which questions should we ask?

Answers cost money
Answers have high latency
Answers have different gains
Problem dimensions

- Offline vs Online
  - An offline algorithm selects the questions without knowledge of the answers to previous questions.

- Bounded vs Unbounded
  - Bounded: limited number of questions.

- Single vs Multiple target nodes.
Offline/Bounded/Single

- Offline ==> We can issue the questions in parallel.
- Bounded ==> We can bound the monetary cost.
- But, we may not be able to find the target node.

Problem statement: Select *k* questions for the crowd such that the number of candidate target nodes is minimized in the worst case.
Offline/Unbounded/Single

Offline ==> We can issue the questions in parallel. Unbounded ==> We want to find the exact target node.

Problem statement: Select the **minimal set of questions** for the crowd such that the target node can be identified **precisely**.
Overview of results

- The problems are computationally hard for DAGs.
- They become tractable for trees.
- Unbounded leads to trivial solutions in most cases.
  - I.e., we have to ask almost all questions.
- Bounded has non-trivial solutions.
Application in practice

G = input graph
while not stopping condition do
  select k questions to ask /* Offline/Bounded */
  ask questions and retrieve answers /* Parallel */
  prune nodes from G based on answers
done
Ongoing and Future Work

- System Design and Implementation
  - Data Model
  - Query Processor
  - Query Optimizer
- Crowdsourced Primitives
  - Image Categorization (CrowdSearch)
  - CrowdScreen: Select items satisfying a condition
  - CrowdRank: Select the top-k items in a set
  - CrowdSort: Sort items
- Other Applications
  - Data curation as a game
  - Natural language interface for relational databases
Thanks!

- [http://www.cs.ucsc.edu/~alkis](http://www.cs.ucsc.edu/~alkis)
  - Scoop
  - Automatic Index Tuning for Database Systems
  - Scientific Data Management

- Human Computation Workshop: Oct 19, E2#599
  - Session 1: 11:00 -- 12:30pm
  - Session 2: 1:30 -- 3:00pm
  - URL: [http://systems.soe.ucsc.edu/node/531](http://systems.soe.ucsc.edu/node/531)