Asking the Right Questions in Crowd Data Sourcing

Tova Milo

TEL AVIV UNIVERSITY
Outline

• Introduction to crowd (data) sourcing
• Databases and crowds
• Declarative is good
• How to best use resources
• Conclusion

Disclaimer: - Very high level
- More questions than answers
- Some nudity 😊

Ack: Some slides are borrowed (with permission) from the VLDB’11 tutorial [DFKK11].
Crowd Sourcing 101

Billions of devices
Crowd Sourcing 101

Ubiquitous connectivity
Examples

Citizen science

Pixels indicate Citizenworker's identified craters

GALAXY ZOO

Classify galaxies

Answer the question below using the buttons provided:

Is the galaxy simply smooth and rounded, with no sign of a disk?

foldit

Solve Puzzles for Science
Examples

Citizen journalism and sensing
Examples

Games are fun!
So what is it all about?

- Bederson & Quinn (Human Computation) CHI’11
  - Motivation (Pay, altruism, enjoyment, ...)
  - Quality control (we’ll talk more about that)
  - Aggregation (We’ll also talk more about that)
  - Human skills (Visual recognition, language, ...)
  - ...


Outline

- Introduction to crowd data sourcing
- Databases and crowds
- Declarative is good
- How to best use resources
- Conclusion
Databases and Crowds

• How can crowds help databases?
  – Fix broken data: entity resolution, inconsistencies
  – Add missing data
  – Subjective comparisons

• How can databases help crowd apps
  – Lazy data acquisition (only get the data that is needed)
  – Manage the data sourced from the crowd
  – Semi automatically create user interfaces
Database platforms for Crowd-based Data Sourcing

- Data models, query languages (query processing, optimization,...)
  - Qurk (MIT)
  - CrowdDB (Berkley, ETH)
  - sCOOP (Stanford, UCSC)
  - FusionCOMP (TsuKuba)
  - MoDaS (Tel Aviv University)
  - ...

- Data quality

- Asking (the crowd) the right questions
Qurk (MIT)

- **Goal**: crowd-source comparisons, missing data
- **Basis**: SQL3 + UDF
  - UDF encapsulates crowd input
  - Special template language for crowd UDFs
  - Specify UI, quality control, possibly opt. hints

- **References**:
  [Marcus et al, CIDR’11, SIGMOD’11]
Qurk example

Is ____ Female?

men in a “people” database

ple((256),

TASK isFemale(tuple) TYPE:Filter
Question: “is %s Female”,
Tuple[“photo”]
YesText: “Yes”
NoText: “No”

\( \exists (p) \);
The magic is in the templates

• Templates generate UIs for different kinds of crowdsourcing tasks
  – Filters: Yes/No questions
  – Joins: comparisons between two tuples (equality)
  – Order by: comparisons between two tuples (>=)
  – Generative: crowdsource attribute value

• Templates also specify quality control; e.g.
  COMBINER: MajorityVote
But, can you trust the Crowd?

Spencer Tunick
Many questions

• How to determine correctness?

• How to clean the data?

• What questions to ask?

• Who to ask? (How many? When to stop?)

• How to best use resources?
Outline

• Introduction to crowd data sourcing
• Databases and crowds
• Declarative is good (but we need more...)
• How to best use resources
• Conclusion
Example: Conflicts resolution

• Average value? Majority vote? Probabilistically?
• But some people know nothing about a given topic
• So maybe a “biased (probabilistic) vote”?
• But how to bias?
• A “chicken or the egg” problem:

To know what is true we need to know who to believe. But to know this we need to know who is usually right (and in particular, what is true..)
Example: So what can we do?

- Start with some estimation on the trust in users.
- Gain confidence in facts based on the opinion of users that supported them.
  - Give bigger weight to user that we trust.
- Then update the trust level in users, based on how many of the facts which they submitted, we believe.
- Iterate until convergence.
  Trusted users give us confidence in facts, and users that supported these facts gain our trust...
- And there is also a probabilistic version...

[Galland et al, WSDM 2010]
But what do we want?

- Not yet another data cleaning algorithm

- We want to have easy control on the employed policy (for data cleaning, query selection, user game scores,...)

- We really don’t want to (re)write Java code (for each tiny change!)

- We want (seamless) optimization, update propagation,...

Database approach:

Define a **declarative language** for specifying policies

[Deutch, Greenshpan, Kostenko, M. ICDE’11, WWW’12]
[Deutch, Koch, M. PODS’10]
Proposed language

- Add to SQL (relational algebra) a **REPAIR-KEY** construct

  **REPAIR-KEY** “repairs” key violations in the database, choosing one possible option, probabilistically, according to the support.

- And a **WHILE** construct

- **Semantics:** Markov chain of DB instances. Probability of a fact to hold in a given instance.

- Expresses nicely common policies for cleaning, selection of questions, scoring answers

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Cuisine</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anton’s</td>
<td>French</td>
<td>0.8</td>
</tr>
<tr>
<td>Anton’s</td>
<td>Continental</td>
<td>0.2</td>
</tr>
<tr>
<td>McDonald</td>
<td>FastFood</td>
<td>1.0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```
TriviaMaster (ICDE 2011 demo)
Some complexity results

Formal problem: Given a Markov Chain of database instances and an SQL query on the database (“what is Anton’s cuisine?”), compute the probabilities of the different answers.

- **Theorem:** Exact computation is \#P-hard

- **Theorem:** If Markov Chain is **ergodic**, computable in \( \text{EXPTIME} \)
  - Compute the stochastic matrix of transitions
  - Compute its fixpoint
  - For ergodic Markov Chain it corresponds to correct probabilities
  - Sum up probabilities of states where the query event holds

- **Theorem:** In general, \( 2\text{-EXPTIME} \)
  - Apply the above to each connected component of the Markov Chain
  - Factor by probability of being in each component
Approximations:

- **Absolute approximation**: approximates correct probability ±ε
- **Relative approximation**: approximates correct probability up to a factor in-between (1- ε), (1+ ε).

[Relative is harder to achieve]

<table>
<thead>
<tr>
<th>Language</th>
<th>Exact computation</th>
<th>Relative approx</th>
<th>Absolute approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Linear) datalog</td>
<td>#P-hard</td>
<td>NP-hard</td>
<td>In PTIME</td>
</tr>
<tr>
<td></td>
<td>In PSPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inflationary fixpoint</td>
<td>#P-hard</td>
<td>NP-hard</td>
<td>In PTIME</td>
</tr>
<tr>
<td></td>
<td>In PSPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-inflationary fixpoint</td>
<td>#P-hard</td>
<td>NP-hard</td>
<td>NP-hard; PTIME in input size and mixing time</td>
</tr>
<tr>
<td></td>
<td>In (2)EXP-TIME</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Still lots of open questions

• How (and when) can we evaluate things fast enough?

• How to store the vast amount of data?
  • Distributed Databases? Map-reduce?

• The data keeps changing. How to handle updates?

• ...

• ...
Outline

• Introduction to crowd data sourcing
• Databases and crowds
• Declarative is good
• How to best use resources
• Conclusion
**Goal**: Compute an aggregate function $f$ for each query, e.g.

- Some metric of the distribution (e.g. entropy)
- Most frequent answer
- Aggregated value (e.g. average)
Increasing knowledge

• Limited overall resources
• Limited user availability
• Bounded resources per question

Which cells to resolve?

[Boim, Greenshpan, M., Novgorodov, Polyzotis, Tan. ICDE’12,...]
Quantifying uncertainty

• Assume \( t \) answers suffice for computing \( f \) for \( q \)

• \( \text{Comp}(q) \): all possible completions of \( q \)'s column

• \( \text{Dist}(r - r') \): distance between two results of \( f \)

• \( \text{Uncertainty}(q) \): \( \max\{ \text{Dist}(f(X) - f(Y)) \mid X,Y \in \text{Comp}(q) \} \)
  i.e. the largest distance between possibly completions
Quantifying uncertainty (cont.)

• Uncertainty measures for a Users-Answers matrix M
  – Max-uncertainty(M)
  – Sum-uncertainty(M)

• Problem statement (X-uncertainty Reduction)
  Given a matrix M, a choice \( x \in \{ \text{max}, \text{sum} \} \), and a set of constraints, identify a set C of empty cells that satisfy the constraints and where

\[
\text{Max}_{M' \in M_C} \ X\text{-uncertainty}(M')
\]

is minimized.

Where \( M_C \) contains all possible matrices that we can derive from M by resolving solely the cells in C.
Example

• Target function
  – Entropy, average, most frequent,…

• Constraints
  – A: bound k on the over number of cells
  – B: also a bound k’ on questions per users
  – C: here k’ is a bound on users per question
Some complexity results

- max-Uncertainty Reduction
  
in PTIME for all constraints classes
  – Greedy algo for constraints class A (and C)
  – Using Max-flow for constraints class B

- sum-Uncertainty Reduction
  
in PTIME for constraint classes A and C
  – Dynamic programming

  NP-COMPLETE for constraints class B
  – Reduction for perfect 3 set cover
AskIt (ICDE’12 demo)

- Gather information (scientific as well as fun) on ICDE’12 authors, participants, papers, presentations,...
Lots of open questions

- Use prior knowledge about users/answers
  - Predict answers
  - Predict who can/will answer what
  [Collaborative Filtering-style analysis is useful here]

- Worse-case analysis vs. expected error

- Treat other goal functions

- Optimization

- Incremental computation
  ...

Outline

• Introduction to crowd data sourcing
• Databases and crowds
• Declarative is good
• How to best use resources
• Conclusion
• All classical issues:
  Data models, query languages, query processing, optimization, HCI

• BUT
  • (Very) interactive computation
  • (Very) large scale data
  • (Very) little control on quality/reliability
  • Closed vs. open world assumption
תודה!
Thanks!
Merci!