Crowdsourced Data Management: Overview and Challenges

Guoliang Li  
Tsinghua University

Yudian Zheng  
Hong Kong University

Ju Fan  
Renmin University

Jiannan Wang  
SFU

Reynold Cheng  
Hong Kong University
Outline

- **Crowdsourcing Overview (30min)**
  - Motivation (5min)
  - Workflow (15min)
  - Platforms (5min)
  - Difference from Other Tutorials (5min)

- **Fundamental Techniques (100min)**
  - Quality Control (60min)
  - Cost Control (20min)
  - Latency Control (20min)

- **Crowdsourced Database Management (40min)**
  - Crowdsourced Databases (20min)
  - Crowdsourced Optimizations (10min)
  - Crowdsourced Operators (10min)

- **Challenges (10min)**
Crowdsourcing: Motivation

- A new computation model
  - Coordinating the crowd (Internet workers) to do micro-tasks in order to solve computer-hard problems.

- Examples
  - Categorize the products and create product taxonomies from the user’s standpoint.
  - An example question
    - Select the product category of Samsung S7
      - Phone
      - TV
      - Movie
Crowdsourcing: Applications

- Wikipedia
  - Collaborative knowledge

- reCAPTCHA
  - Digitalizing newspapers

- Foldit
  - Fold the structures of selected proteins

- App Testing
  - Test apps
Crowdsourcing: Popular Tasks

- **Sentiment Analysis**
  - Understand conversation: positive/negative

- **Search Relevance**
  - Return relevant results on the first search

- **Content Moderation**
  - Keep the best, lose the worst

- **Data Collection**
  - Verify and enrich your business data

- **Data Categorization**
  - Organize your data

- **Transcription**
  - Turn images and audio into useful data
Crowdsourcing Space

Granularity

Macro

Micro

Money

Entertainment

Hidden

Volunteer

Incentive

ESP Game (Luis von Ahn)

Object: Images Labeling

Human task: online game, two players guessing one common item

Example:

ESP Game

reCAPTCHA

Volunteer

Google

Wikipedia

Yahoo Answers

IMAGENET

foldit

Example:

PLAYER 1

GUESSING: CAR
GUESSING: HAT

PLAYER 2

GUESSING: BOY
GUESSING: CAR

Lobbyist

Android/iOS developer

Hourly Rate: $28/hr
Location: Ukraine
Job Success: 100%
Crowdsourcing Category

○ Game vs Payment
  – Simple tasks
    • Both payment and game can achieve high quality
  – Complex tasks
    • Game has better quality

Quality is rather important!
Crowdsourcing: Workflow

- **Requester**
  - Submit Tasks

- **Platforms**
  - Task Management

- **Workers**
  - Worker on Tasks
Crowdsourcing Requester: Workflow

○ Design Tasks
  • Task Type
  • Design Strategies
    – UI, API, Coding

○ Upload Data

○ Set Tasks
  • Price
  • Time
  • Quality

○ Publish Task
  • Pay
  • Monitor
Crowdsourcing Requester: Task Type

- **Task Type**

  Please choose the brand of the phone:
  - Apple
  - Samsung
  - Blackberry
  - Other

  What are comment features?
  - [ ] Same band
  - [ ] Same color
  - [ ] Similar price
  - [ ] Same size

  Please fill the attributes of the product:
  - Brand
  - Price
  - Size
  - Camera

  Please submit a picture of a phone with the same size as the left one.

Submit
Crowdsourcing Requester: Task Design

- **UI**

  Choose the best category for the image
  - Kitchen
  - Bath
  - Living
  - Bed

- **API**

  The Amazon Mechanical Turk API consists of web service operations for every task the service can perform. This section describes each operation in detail.
  - AcceptQualificationRequest
  - ApproveAssignment
  - AssociateQualificationWithWorker
  - CreateAdditionalAssignmentsForHIT
  - CreateHIT

- **Coding**

  (Your own Server)

  `innerHTML`

  ```python
  # Create the HIT
  response = client.create_hit(
    MaxAssignments = 10,
    LifetimeInSeconds = 600,
    AssignmentDurationInSeconds = 600,
    Reward = '0.20',
    Title = 'Answer a simple question',
    Keywords = 'question, answer, research',
    Description = 'Answer a simple question',
    Question = questionSample,
    QualificationRequirements = localRequirements
  )

  # The response included several fields that will be helpful later
  hit_type_id = response['HIT']['HITTypeId']
  hit_id = response['HIT']['HITId']
  print "Your HIT has been created. You can see it at this link:"
  print "https://workersandbox.mturk.com/mturk/preview?groupId={}".format(hit_type_id)
  print "Your HIT ID is: {}".format(hit_id)
  ```
Crowdsourcing Requester: Task Setting

- **HIT** – A group of micro-tasks (e.g., 5)
- **Price, Assignment, Time**

<table>
<thead>
<tr>
<th>Setting up your HIT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reward per assignment</strong></td>
<td>$0.05</td>
</tr>
<tr>
<td>This is how much a Worker will be paid for completing an assignment. Consider how long it will take a Worker to</td>
<td></td>
</tr>
<tr>
<td><strong>Number of assignments per HIT</strong></td>
<td>3</td>
</tr>
<tr>
<td>How many unique Workers do you want to work on each HIT?</td>
<td></td>
</tr>
<tr>
<td><strong>Time allotted per assignment</strong></td>
<td>1 Hours</td>
</tr>
<tr>
<td>Maximum time a Worker has to work on a single task. Be generous so that Workers are not rushed.</td>
<td></td>
</tr>
<tr>
<td><strong>HIT expires in</strong></td>
<td>7 Days</td>
</tr>
<tr>
<td>Maximum time your HIT will be available to Workers on Mechanical Turk.</td>
<td></td>
</tr>
<tr>
<td><strong>Auto-approve and pay Workers in</strong></td>
<td>3 Days</td>
</tr>
<tr>
<td>This is the amount of time you have to reject a Worker's assignment after they submit the assignment.</td>
<td></td>
</tr>
</tbody>
</table>
Crowdsourcing Requester: Task Setting

- Quality Control
  - Qualification test - Quiz
    Create some test questions to enable a quiz that workers must pass to work on your task.
  
  - Hidden test - Training
    Add some questions with ground truths in your task so workers who get them wrong will be eliminated.

- Worker selection
  Ensure high-quality results by eliminating workers who repeatedly fail test questions in your task.
Crowdsourcing Requester: Publish

- **Prepay**
  
  cost for workers + cost for platform + cost for test

<table>
<thead>
<tr>
<th>Expected Cost:</th>
<th>$0.00</th>
<th>$0.00</th>
<th>$10.00</th>
<th>$0.00</th>
<th>$10.00</th>
<th>$16.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contributor judgments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost buffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction fee (20%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due Now</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$10.00</td>
<td>$0.00</td>
<td>$10.00</td>
<td>$16.01</td>
</tr>
<tr>
<td>Available Funds</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Funds</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reward per Assignment:</th>
<th>$0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>x 3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimated Total Reward:</th>
<th>$0.15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Fees to Mechanical Turk:</td>
<td>+ $0.03</td>
</tr>
</tbody>
</table>

| Estimated Cost: | $0.18 |

- **Monitor**

<table>
<thead>
<tr>
<th>Real-time Statistics</th>
<th>0%</th>
<th>3</th>
<th>¥ 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finished Units</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Workers per unit</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Cost</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

SIGMOD’17 Tutorial  14
Crowdsourcing: Workers

- Task Selection
- Task Completion
- Workers are not free Cost
  - Make Money
- Workers are not oracle Quality
  - Make errors
  - Malicious workers
- Workers are dynamic Latency
  - Hard to predict
Crowdsourcing: Platforms

- Amazon Mechanical Turk (AMT)

- Requesters
- HIT (k tasks)
- Workers

more than 500,000 workers from 190 countries
Crowdsourcing: Platforms

- CrowdFlower

- **Requesters**
  - Create a new job
    - Select a template or start from scratch
  - What would you like to do?
    - Sentiment Analysis
    - Search Relevance
    - Data Validation
    - Image Annotation

- **HIT (k tasks)**
  - iPhone 2 = iPad Two?
    - equal
    - non-equal
  - iWatch Two = iPad2?
    - equal
    - non-equal
  - Submit

- **Workers**
  - Jobs: 12
  - Tasks: 48
  - Accuracy: 92%
# AMT vs CrowdFlower

<table>
<thead>
<tr>
<th></th>
<th>AMT</th>
<th>CrowdFlower</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Design: UI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Task Design: API</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Task Design: Coding</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Quality: Qualification Test</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Quality: Hidden Test</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Quality: Worker Selection</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Task Types</td>
<td>All Types</td>
<td>All Types</td>
</tr>
</tbody>
</table>
AMT Task Statistics

http://www.mturk-tracker.com
Other Crowdsourcing Platforms

- **Macrotask**
  - **Upwork**
    - [https://www.upwork.com](https://www.upwork.com)
  - **Zhubajie**
    - [http://www.zbj.com](http://www.zbj.com)

- **Microtask**
  - **ChinaCrowds** *(cover all features of AMT and CrowdFlower)*
    - [http://www.chinacrowds.com](http://www.chinacrowds.com)
Crowdsourcing: Challenges

- Crowd is not free
- Reduce monetary cost
- Crowd is not real-time
- Reduce time
- Crowd may return incorrect answers
- Improve quality
Crowdsourced Data Management

- A crowd-powered database system
  - Users require to write code to utilize crowdsourcing platforms
  - Encapsulates the complexities of interacting with the crowd
  - Make DB more powerful
- Crowd-powered interface
- Crowd-powered Operators
- Crowdsourcing Optimization

![Diagram of crowdsourcing system and operators](image)

- SQL-like Crowdsourcing Query Language
  - Query Optimizer
  - Query
  - Results

- Crowdsourcing Operators
  - CrowdSelect
  - CrowdJoin
  - CrowdSort
  - CrowdTopK
  - CrowdMax
  - CrowdMin
  - CrowdCount
  - CrowdCollect
  - CrowdFill

- Crowdsourcing Executive
  - Truth Inference
  - Task Assignment
  - Answer Reasoning
  - Task Design
  - Latency Reduction

- Crowdsourcing Requester
  - Query
  - Relations
  - Statistics

- Crowdsourcing Platform
  - Tasks
  - Answers
Tutorial Outline

- **Fundamental Optimization**
  - Quality Control
  - Cost Control
  - Latency Control

- **Crowd-powered Database**

- **Crowd-powered Operators**
  - Selection/Join/Group
  - Topk/Sort
  - Collection/Fill

- **Challenges**
Existing Works

SIGMOD
VLDB
ICDE
SUM

0 2 4 6 8 10 12 14 16
Differences with Existing Tutorials

- **VLDB’16**
  - Human factors involved in task assignment and completion.

- **VLDB’15**
  - Truth inference in quality control

- **ICDE’15**
  - Individual crowdsourcing operators, crowdsourced data mining and social applications

- **VLDB’12**
  - Crowdsourcing platforms and Design principles

- **Our Tutorial**
  - Control **quality, cost and latency**
  - Design **Crowdsourced Database**
Outline

◦ Crowdsourcing Overview (30min)
  – Motivation (5min)
  – Workflow (15min)
  – Platforms (5min)
  – Difference from Other Tutorials (5min)

◦ Fundamental Techniques (100min)
  – Quality Control (60min)
    – Cost Control (20min)
    – Latency Control (20min)

◦ Crowdsourced Database Management (40min)
  – Crowdsourced Databases (20min)
  – Crowdsourced Optimizations (10min)
  – Crowdsourced Operators (10min)

◦ Challenges (10min)
Why Quality Control?

- **Huge Amount** of Crowdsourced Data
  - [Graph showing AMT worker availability]
  - Statistics in AMT:
    - Over 500K workers
    - Over 1M tasks

- **Inevitable noise & error**
  - [Images of people and icons representing various issues]

- **Goal**: Obtain reliable information in Crowdsourced Data
Crowdsourcing Workflow

- **Requester** deploys tasks and budget on crowdsourcing platform (e.g., AMT)
- **Workers** interact with platform (2 phases)
  1. When a worker comes to the platform, the worker will be assigned to a set of tasks (task assignment);
  2. When a worker accomplishes tasks, the platform will collect answers from the worker (truth inference).

![Diagram of crowdsourcing workflow with Requester, DB, Truth Inference, Task Assignment, and Workers]
Outline of Quality Control

Part I. Truth Inference
- Problem Definition
- Condition 1: with ground truth
  - Qualification Test & Hidden Test
- Condition 2: without ground truth
  - Unified Framework
  - Differences in Existing Works
  - Experimental Results

Part II. Task Assignment
- Problem Definition
- Differences in Existing Works
Part I. Truth Inference

- An Example Task

What is the current affiliation for Michael Franklin?

A. University of California, Berkeley
B. University of Chicago

I support A. UCB!
Principle: Redundancy

- Collect Answers from Multiple Workers

What is the current affiliation for Michael Franklin?

A. University of California, Berkeley
B. University of Chicago

How to infer the truth of the task?
Outline of Quality Control

- Part I. Truth Inference
  - Problem Definition
  - Condition 1: with ground truth
    - Qualification Test & Hidden Test
  - Condition 2: without ground truth
    - Unified Framework
    - Differences in Existing Works
    - Experimental Results

- Part II. Task Assignment
  - Problem Definition
  - Differences in Existing Works
Given different tasks’ answers collected from workers, the target is to infer the truth of each task.

Truth Inference Definition
A Simple Solution

- Majority Voting
  Take the answer that is voted by the majority (or most) of workers.

- Limitation
  Treat each worker equally, neglecting the diverse quality for each worker.
The Key to Truth Inference

- The key is to know each worker’s quality

Suppose quality of 4 workers are known
How to know worker’s quality?

1. If a small set of tasks with ground truth are known in advance (e.g., refer to experts)

   We can estimate each worker’s quality based on the answering performance for the tasks with known truth

2. If no ground truth is known in advance

   The only way is to estimate each worker’s quality based on the collected answers from all workers for all tasks
Outline

○ Part I. Truth Inference
  – Problem Definition
  – Condition 1: with ground truth
    • Qualification Test & Hidden Test
  – Condition 2: without ground truth
    • Unified Framework
    • Existing Works
    • Experimental Results

○ Part II. Task Assignment
  – Problem Definition
  – Differences in Existing Works
1. A Small Set of Ground Truth is Known

- Qualification Test (*like an “exam”*)

  Assign the tasks (with known truth) to the worker when the worker comes at first time e.g., *if the worker answers 8 over 10 tasks correctly, then the quality is 0.8*

- Hidden Test (*like a “landmine”*)

  Embed the tasks (with known truth) in all the tasks assigned to the worker e.g., *each time 10 tasks are assigned to a worker, then 10 tasks compose of 9 real tasks (with unknown truth), and 1 task with known truth*
1. A Small Set of Ground Truth is Known

- Limitations of two approaches
  
  1. need to know ground truth (may refer to experts);
  2. waste of money because workers need to answer these “extra” tasks;
  3. as reported (Zheng et al. VLDB’17), these techniques may not improve much quality.

Thus the assumption of “no ground truth is known” is widely adopted by existing works.
Outline

○ Part I. Truth Inference
  – Problem Definition
  – Condition 1: with ground truth
    • Qualification Test & Hidden Test
  – Condition 2: without ground truth
    • Unified Framework
    • Existing Works
    • Experimental Results

○ Part II. Task Assignment
  – Problem Definition
  – Differences in Existing Works
2. If No Ground Truth is Known

- How to know each worker’s quality given the collected answers for all tasks?

Answers:

- B
- B
- B
- A
- A
- B

Current affiliation?
A. UCB
B. Chicago

Current affiliation?
A. Google
B. Recruit.ai
**Unified Framework in Existing Works**

- **Input:** Workers’ answers for all tasks

- **Algorithm Framework:**

  ```
  Initialize Quality for each worker
  while (not converged) {
    Quality for each worker ➔ Truth for each task ;
    Truth for each task ➔ Quality for each worker ;
  }
  ```

- **Output:** Quality for each worker and Truth for each task
Inherent Relationship 1

1. Quality for each worker → Truth for each task

Quality:

Truth:

A. UCB (1.0 from worker 3)
B. Chicago (1.0 + 1.0 from workers 1 & 2)

A. Google (1.0 from worker 2)
B. Recruit.ai (1.0 + 1.0 from workers 1 & 3)
2. Truth for each task

Truth:

- A. UCB
- B. Chicago

Quality:

- B: 1.0 (correct: 2/2)
- A: 0.5 (correct: 1/2)
- B: 0.5 (correct: 1/2)

Current affiliation?

- A. Google
- B. Recruit.ai
Outline

○ Part I. Truth Inference
  – Problem Definition
  – Condition 1: with ground truth
    • Qualification Test & Hidden Test
  – Condition 2: without ground truth
    • Unified Framework
    • Existing Works
    • Experimental Results

○ Part II. Task Assignment
  – Problem Definition
  – Differences in Existing Works
Existing works

- **Classic Method**
  D&S [Dawid and Skene. JRSS 1979]

- **Recent Methods**
  (1) **Database Community:**
  CATD [Li et al. VLDB14], PM [Li et al. SIGMOD14], iCrowd [Fan et al. SIGMOD15], DOCS [Zheng et al. VLDB17]

  (2) **Data Mining Community:**
  ZC [Demartini et al. WWW12], Multi [Welinder et al. NIPS 2010], CBCC [Venanzi et al. WWW14]

  (3) **Machine Learning Community:**
  GLAD [Whitehill et al. NIPS09], Minimax [Zhou et al. NIPS12], BCC [Kim et al. AISTATS12], LFC [Raykar et al. JMLR10], KOS [Karger et al. NIPS11], VI-BP [Liu et al. NIPS12], VI-MF [Liu et al. NIPS12], LFC_N [Raykar et al. JMLR10]
Differences in Existing works

Tasks

- Different Task Types
  
  *What type of tasks they focus on?*
  
  *E.g., single-label tasks…*

- Different Task Models
  
  *How they model each task?*
  
  *E.g., task difficulty…*

Workers

- Different Worker Models
  
  *How they model each worker?*
  
  *E.g., worker probability (a value)…*
Tasks: Different Tasks Types

- **Decision-Making Tasks** (yes/no task)
  
  Is Bill Gates currently the CEO of Microsoft?  
  - Yes  
  - No  

  e.g., Demartini et al. WWW12, Whitehill et al. NIPS09, Kim et al. AISTATS12, Venanzi et al. WWW14, Raykar et al. JMLR10

- **Single-Label Tasks** (multiple choices)
  
  Identify the sentiment of the tweet: ……  
  - Pos  
  - Neu  
  - Neg  

  e.g., Li et al. VLDB14, Li et al. SIGMOD14, Demartini et al. WWW12, Whitehill et al. NIPS09, Kim et al. AISTATS12

- **Numeric Tasks** (answer with numeric values)
  
  What is the height for Mount Everest?  

  □ m  

  e.g., Li et al. VLDB14, Li et al. SIGMOD14
Task Difficulty: a value

If a task receives many contradicting (or ambiguous) answers, then it is regarded as a difficult task.

e.g., Welinder et al. NIPS 2010, Ma et al. KDD16

Diverse Domains: a vector

Did Michael Jordan win more NBA championships than Kobe Bryant?

Is there a name for the song that FC Barcelona is known for?
Tasks: Different Task Models (cont’d)

- Diverse Domains (cont’d)

To obtain the each task’s model:

1. Use machine learning approaches e.g., LDA [Blei et al. JMLR03], TwitterLDA [Zhao et al. ECIR11].

2. Use entity linking (map entity to knowledge bases).

Did Michael Jordan win more NBA championships than Kobe Bryant?
Workers: Different Worker Models

- **Worker Probability**: a value $p \in [0,1]$

  The probability that the worker answers tasks correctly e.g., a worker answers 8 over 10 tasks correctly, then the worker probability is 0.8.
  e.g., Demartini et al. WWW12, Whitehill et al. NIPS09

- **Confidence Interval**: a range $[p - \varepsilon, p + \varepsilon]$

  $\varepsilon$ is related to the number of tasks answered => the more answers collected, the smaller $\varepsilon$ is.
  e.g., two workers answer 8 over 10 tasks and 40 over 50 tasks correctly, then the latter worker has a smaller $\varepsilon$.

  e.g., Li et al. VLDB14
Workers: Different Worker Models (cont’d)

- **Confusion Matrix**: a matrix

Capture a worker’s answer for different choices given a specific truth

<table>
<thead>
<tr>
<th></th>
<th>Pos</th>
<th>Neu</th>
<th>Neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos</td>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Neu</td>
<td>0.3</td>
<td>0.6</td>
<td>0.1</td>
</tr>
<tr>
<td>Neg</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Given that the truth of a task is “Neu”, the probability that the worker answers “Pos” is 0.3.

- **Bias** $\tau$ & **Variance** $\sigma$: numerical task

Answer follows Gaussian distribution: $\text{ans} \sim N(t + \tau, \sigma)$

- e.g., Kim et al. AISTATS12, Venanzi et al. WWW14
- e.g., Raykar et al. JMLR10
Workers: Different Worker Models (cont’d)

- Quality Across Diverse Domains: a vector

How to decide the scope of domains?

Idea: Use domains from Knowledge Bases

e.g., Ma et al. KDD16, Zheng et al. VLDB17

SIGMOD’17 Tutorial
<table>
<thead>
<tr>
<th>Method</th>
<th>Task Type</th>
<th>Task Model</th>
<th>Worker Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>Decision-Making Task, Single-Choice Task</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Mean / Median</td>
<td>Numeric Task</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CATD [Li et al. VLDB14]</td>
<td>Decision-Making Task, Single-Choice Task, Numeric Task</td>
<td>No</td>
<td>Worker Probability, Confidence</td>
</tr>
</tbody>
</table>
## Summary of Truth Inference Methods (cont’d)

<table>
<thead>
<tr>
<th>Method</th>
<th>Task Type</th>
<th>Task Model</th>
<th>Worker Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM [Li et al. SIGMOD14]</td>
<td>Decision-Making Task, Single-Choice Task, Numeric Task</td>
<td>No</td>
<td>Worker Probability</td>
</tr>
<tr>
<td>Multi [Welinder et al. NIPS 2010]</td>
<td>Decision-Making Task</td>
<td>Diverse Domains</td>
<td>Diverse Domains, Worker Bias, Worker Variance</td>
</tr>
<tr>
<td>LFC_N [Raykar et al. JMLR10]</td>
<td>Numeric Task</td>
<td>No</td>
<td>Worker Variance</td>
</tr>
</tbody>
</table>
Outline

○ Part I. Truth Inference
  – Problem Definition
  – Condition 1: with ground truth
    • Qualification Test & Hidden Test
  – Condition 2: without ground truth
    • Unified Framework
    • Existing Works
    • Experimental Results

○ Part II. Task Assignment
  – Problem Definition
  – Differences in Existing Works
Experimental Results (Zheng et al. VLDB17)

- **Statistics of Datasets**

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Tasks</th>
<th># Answers Per Task</th>
<th># Workers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentiment Analysis</td>
<td>1000</td>
<td>20</td>
<td>185</td>
<td>Given a tweet, the worker will identify the sentiment of the tweet</td>
</tr>
<tr>
<td>[Zheng et al. VLDB17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duck</td>
<td>108</td>
<td>39</td>
<td>39</td>
<td>Given an image, the worker will identify whether the image contains a duck or not</td>
</tr>
<tr>
<td>[Welinder et al. NIPS10]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>8315</td>
<td>3</td>
<td>85</td>
<td>Given a pair of products, the worker will identify whether or not they refer to the same product</td>
</tr>
<tr>
<td>[Wang et al. VLDB12]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experimental Results

- Observations (Sentiment Analysis)

#workers’ answers conform to long-tail phenomenon (Li et al. VLDB14)

Not all workers are of very high quality
Experimental Results (cont’d)

- Change of Quality vs. \#Answers (Sentiment Analysis)

Observations:

1. The quality increases with \#answers;

2. The quality improvement is significant with few answers, and is marginal with more answers;

3. Most methods are similar, except for Majority Voting (in pink color).
Experimental Results (cont’d)

- Performance on more datasets

**Dataset “Duck”**

**Dataset “Product”**

![Graphs showing accuracy and F1-score for different datasets and models.](image)
Which method is the best?

○ Decision-Making & Single-Label Tasks
  – “Majority Voting” if sufficient data is given (each task collects more than 20 answers);
  – “D&S [Dawid and Skene JRSS 1979]” if limited data is given (a robust method);

○ Numeric Tasks
  – “Mean” since it is robust in practice;
  – “PM [Li et al. SIGMOD14]” as advanced techniques.
Take-Away for Truth Inference

○ The key to truth is to compute each worker’s quality

○ if some truth is known:
  qualification test and hidden test;

○ if no truth is known:
  Truth

(1) relationships between “quality for each worker” and “truth for each task”

(2) different task types & models and worker models
Crowdsourcing Workflow

- Requester deploys tasks and budget on crowdsourcing platform (e.g., Amazon Mechanical Turk)
- Workers interact with platform (2 phases)
  1. when a worker comes to the platform, the worker will be assigned to a set of tasks (task assignment);
  2. when a worker accomplishes tasks, the platform will collect answers from the worker (truth inference).
Part II. Task Assignment

- Existing platforms support online task assignment
  - Amazon Mechanical Turk
    - "External HIT"

- Intuition: requesters want to wisely use the budgets
  - I am requester, and I want to use my budgets very well!
  - We are workers!

How to allocate suitable tasks to workers?
Task Assignment Problem

Given a pool of $n$ tasks, which set of the $k$ tasks should be batched in a HIT and assigned to the worker?

Example:
Suppose we have $n=4$ tasks, and each time $k=2$ tasks are assigned as a HIT.
This problem is complex!

- Simple enumeration: “n choose k” combinations
  
  \((n = 100, \ k = 5) \rightarrow 100M\) assignments

- Need efficient (online) assignment
  
  Fast response to worker’s request

- Develop efficient heuristics
  
  Assignment time linear in \#tasks: \(O(n)\)
Outline

- Part I. Truth Inference
  - Problem Definition
  - Condition 1: with ground truth
    - Qualification Test & Hidden Test
  - Condition 2: without ground truth
    - Unified Framework
    - Existing Works
    - Experimental Results

- Part II. Task Assignment
  - Problem Definition
  - Existing Works
Main Idea

Most suitable tasks

3 factors for characterizing a suitable task:
Answer uncertainty
Worker quality
Requesters’ objectives
Factor 1: Answer Uncertainty

- Consider a decision-making task (yes/no)

<table>
<thead>
<tr>
<th></th>
<th>0 yes</th>
<th>1 yes</th>
<th>2 no</th>
<th>3 yes</th>
<th>0 no</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

- Select a task whose answers are the most uncertain or inconsistent

  e.g., Liu et al. VLDB12, Roim et al. ICDE12
Factor 1: Answer Uncertainty

- **Entropy** (Zheng et al. SIGMOD15)
  Given $c$ choices for a task and the distribution of answers for a task $\bar{p} = (p_1, p_2, \ldots, p_c)$
  The task’s entropy is:
  \[
  H(\bar{p}) = -\sum_{i=1}^{c} p_i \log p_i
  \]
  e.g., a task receives 1 “yes” and 2 “no”, then the distribution is $(1/3, 2/3)$, and entropy is 0.637.

- **Expected change of entropy** (Roim et al. ICDE12)
  $(1/3, 2/3)$ should be more uncertain than $(10/30, 20/30)$:
  \[
  E[H(\bar{p}')] - H(\bar{p})
  \]
Factor 2: Worker Quality

- Assign tasks to the worker with the suitable expertise
e.g., Ho et al. AAAI12, Zheng et al. VLDB17

- Uncertainty: consider the matching domains in tasks and the worker

SIGMOD’17 Tutorial
Factor 3: Objectives of Requesters

- Requesters may have different objectives (aka “evaluation metric”) for different applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Sentiment Analysis</th>
<th>Entity Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td><img src="image" alt="Sentiment Analysis Task" /></td>
<td><img src="image" alt="Entity Resolution Task" /></td>
</tr>
<tr>
<td><strong>Evaluation Metric</strong></td>
<td>Accuracy</td>
<td>F-score (&quot;equal&quot; label)</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Accuracy" /></td>
<td><img src="image" alt="F-score" /></td>
</tr>
</tbody>
</table>
Factor 3: Objectives of Requesters

- Solution in QASCA (Zheng et al. SIGMOD15)
  1. Leverage the answers collected from workers to create a “distribution matrix”;
  2. Leverage the “distribution matrix” to estimate the quality improvement for a specific set of selected tasks.

- Idea: Select the best set of tasks with highest quality improvement in the specified evaluation metric.
Factor 3: Objectives of Requesters

- Other Objectives

1. **Threshold on entropy** *(e.g., Li et al. WSDM17)*
   e.g., in the final state, each task should have a constraint that its entropy ≥ 0.6.

2. **Threshold on worker quality** *(e.g., Fan et al. SIGMOD15)*
   e.g., in the final state, each task should have overall aggregated worker quality ≥ 2.0.

3. **Maximize total utility** *(e.g., Ho et al. AAAI12)*
   e.g., after the answer is given, the requester receives some utility related to worker quality, and the goal is to assign tasks that maximize the total utility.
## Task Assignment

<table>
<thead>
<tr>
<th>Method</th>
<th>Factor 1: Answer Uncertainty</th>
<th>Factor 2: Worker Quality</th>
<th>Factor 3: Requesters’ Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTA [Ho et al. AAAI12]</td>
<td>Majority</td>
<td>Worker probability</td>
<td>Maximize total utility</td>
</tr>
<tr>
<td>CDAS [Liu et al. VLDB12]</td>
<td>Majority</td>
<td>Worker probability</td>
<td>A threshold on confidence + early termination of confident tasks</td>
</tr>
<tr>
<td>iCrowd [Fan et al. SIGMOD15]</td>
<td>Majority</td>
<td>Diverse domains</td>
<td>Maximize overall worker quality</td>
</tr>
<tr>
<td>AskIt! [Roim et al. ICDE12]</td>
<td>Entropy-based</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DOCS [Zheng et al. VLDB17]</td>
<td>Expected change of entropy</td>
<td>Diverse domains</td>
<td>No</td>
</tr>
<tr>
<td>CrowdPOI [Hu et al. ICDE16]</td>
<td>Expected change of accuracy</td>
<td>Worker probability</td>
<td>No</td>
</tr>
<tr>
<td>Opt-KG [Li et al. WSDM17]</td>
<td>Majority</td>
<td>No</td>
<td>≥ threshold on entropy</td>
</tr>
</tbody>
</table>
Take-Away for Task Assignment

- Require **online** and **efficient** heuristics

- Key idea: assign the **most suitable** task to worker, based on:
  
  (1) uncertainty of collected answers;
  (2) worker quality; and
  (3) requester’ objectives.
Public Datasets & Codes

- **Public crowdsourcing datasets**
  ([http://i.cs.hku.hk/~ydzheng2/crowd_survey/datasets.html](http://i.cs.hku.hk/~ydzheng2/crowd_survey/datasets.html)).

- **Implementations of truth inference algorithms**

- **Implementations of task assignment algorithms**
  ([https://github.com/TsinghuaDatabaseGroup/CrowdOTA](https://github.com/TsinghuaDatabaseGroup/CrowdOTA)).
Reference – Truth Inference

Reference – Truth Inference (cont’d)

Reference – Truth Inference (cont’d)

Reference – Task Assignment

[10] Jing Gao, Qi Li, Bo Zhao, Wei Fan, and Jiawei Han, Enabling the Discovery of Reliable Information from Passively and Actively Crowdsourced Data, KDD’16 tutorial.
Outline

- Crowdsourcing Overview (30min)
  - Motivation (5min)
  - Workflow (15min)
  - Platforms (5min)
  - Difference from Other Tutorials (5min)

- Fundamental Techniques (100min)
  - Quality Control (60min)
  - Cost Control (20min)
  - Latency Control (20min)

- Crowdsourced Database Management (40min)
  - Crowdsourced Databases (20min)
  - Crowdsourced Optimizations (10min)
  - Crowdsourced Operators (10min)

- Challenges (10min)
Cost Control

- **Goal**
  - How to reduce monetary cost?

- **Cost** = $n \times c$
  - $n$: number of tasks
  - $c$: cost of each task

- **Challenges**
  - How to reduce $n$?
  - How to reduce $c$?
Classification of Existing Techniques

- **How to reduce** $n$?
  - Task Pruning
  - Answer Deduction
  - Task Selection
  - Sampling

- **How to reduce** $c$?
  - Task Design

The Database Community

The HCI Community
Task Pruning

- **Key Idea**
  - Prune the tasks that machines can do well

- **Easy Task vs. Hard Task**

<table>
<thead>
<tr>
<th>Are they the same?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPHONE 6 = iphone 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Are they the same?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM = Big Blue</td>
</tr>
</tbody>
</table>

- **How to quantify "difficulty"**
  - Similarity value
  - Match probability

- Jiannan Wang, Tim Kraska, Michael J. Franklin, Jianhua Feng: CrowdER: Crowdsourcing Entity Resolution. VLDB 2012
- Steven Euijong Whang, Peter Lofgren, Hector Garcia-Molina: Question Selection for Crowd Entity Resolution. VLDB 2013
Task Pruning (cont’d)

○ **Workflow (non-iterative)**
  1. Rank tasks based on "difficulty"
  2. Prune the tasks whose difficulty ≤ threshold

○ **Pros**
  – Support a large variety of applications

○ **Cons**
  – Only work for easy tasks (i.e., the ones that machines can do well)
Classification of Existing Techniques

- **How to reduce** $n$?
  - Task Pruning
  - Answer Deduction
  - Task Selection
  - Sampling

- **How to reduce** $c$?
  - Task Design

The Database Community

The HCI Community
Answer Deduction

- **Key Idea**
  - Prune the tasks whose answers can be **deduced** from existing crowdsourced tasks

- **Example: Transitivity**

  ![Diagram showing transitivity deduction example]
Answer Deduction (cont’d)

○ Workflow (iterative)

1. Pick up some tasks from a task pool
2. Collect answers of the tasks from the Crowd
3. Remove the tasks whose answers can be deduced
Answer Deduction (cont’d)

- **Pros**
  - Work for both easy and **hard** tasks

- **Cons**
  - Human errors can be amplified

Wrong
Classification of Existing Techniques

- **How to reduce** $n$?
  - Task Pruning
  - Answer Deduction
  - Task Selection
  - Sampling

- **How to reduce** $c$?
  - Task Design

The Database Community

The HCI Community
Task Selection

- **Key Idea**
  - Select the most **beneficial** tasks to crowdsourc

- **Example 1: Active Learning**
  - Most beneficial for training a model

---

- Mozafari et al. Scaling Up Crowd-Sourcing to Very Large Datasets: A Case for Active Learning. PVLDB 2014
- Gokhale et al. Corleone: hands-off crowdsourcing for entity matching. SIGMOD 2014
Task Selection

- **Key Idea**
  - Select the most beneficial tasks to crowdsource

- **Example 2: Top-k**
  - Most beneficial for getting the top-k results

Which picture visualizes the best SFU Campus?

Rank by computers

The most beneficial task:

VS.

Xiaohang Zhang, Guoliang Li, Jianhua Feng: Crowdsourced Top-k Algorithms: An Experimental Evaluation. PVLDB 2016
Task Selection (cont’d)

- **Workflow (iterative)**
  1. Select a set of most beneficial tasks
  2. Collect their answers from the Crowd
  3. Update models and results

- **Pros**
  - Allow for a flexible quality/cost trade-off

- **Cons**
  - Hurt latency (since only a small number of tasks can be crowdsourced at each iteration)
Classification of Existing Techniques

- How to reduce $n$?
  - Task Pruning
  - Answer Deduction
  - Task Selection
  - Sampling

- How to reduce $c$?
  - Task Design

The Database Community

The HCI Community
Sampling

- Key Idea
  - Ask the crowd to work on **sample** data

- Example: SampleClean

Who published more?

Rakesh Agrawal
- Microsoft
- Publication: 353
- Fields: Databases, D
- Collaborated with 365

Jeffrey D. Ullman
- Stanford University
- Publication: 460
- Fields: Databases, A
- Collaborated with 317

Michael Franklin
- University of California
- Publication: 561
- Fields: Databases, P
- Collaborated with 345
Sampling (Cont’d)

- **Workflow (iterative)**
  1. Generate tasks based on a sample
  2. Collect the task answers from the Crowd
  3. Infer the results of the full data

- **Pros**
  - Provable bounds for quality (e.g., the paper count is 211±5 with 95% probability)

- **Cons**
  - Limited to certain applications (e.g., it does not work for max)
Classification of Existing Techniques

- **How to reduce $n$?**
  - Task Pruning
  - Answer Deduction
  - Task Selection
  - Sampling

- **How to reduce $c$?**
  - Task Design

The Database Community

The HCI Community
Task Design (Cont’d)

- **Key Idea**
  - Optimize User Interface

- **Example 1: Count**

  How many are female?
  
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6

Adam Marcus, David R. Karger, Samuel Madden, Rob Miller, Sewoong Oh: Counting with the Crowd. PVLDB 6(2): 167-180 (2013)
Task Design (Cont’d)

- **Key Idea**
  - Optimize User Interface

- **Example 2: Image Labeling**

![Image Labeling Example](image.png)
Summary of Cost Control

- Two directions
  - How to reduce n?  DB
  - How to reduce c?  HCl

- DB and HCl should work together

- Non-iterative and iterative workflows are both widely used
Outline

◦ Crowdsourcing Overview (30min)
  – Motivation (5min)
  – Workflow (15min)
  – Platforms (5min)
  – Difference from Other Tutorials (5min)

◦ Fundamental Techniques (100min)
  – Quality Control (60min)
  – Cost Control (20min)
  – Latency Control (20min)

◦ Crowdsourced Database Management (40min)
  – Crowdsourced Databases (20min)
  – Crowdsourced Optimizations (10min)
  – Crowdsourced Operators (10min)

◦ Challenges (10min)
Latency Control

○ **Goal**
  – How to reduce latency?

○ **Latency =** $n \times t$
  – $n$: number of tasks
  – $t$: latency of each task

○ **Latency =** The completion time of the last task
Classification of Latency Control

1. Single Task
   - Reduce the latency of a single task

2. Single Batch
   - Reduce the latency of a batch of tasks

3. Multiple Batches
   - Reduce the latency of multiple batches of tasks
Single-Task Latency Control

• Latency consists of
  – Phase 1: Recruitment Time
  – Phase 2: Qualification and Training Time
  – Phase 3: Work Time

• Improve Phase 1
  – See the next slide

• Improve Phase 2
  – Remove this phase by applying other quality control techniques (e.g., worker elimination)

• Improve Phase 3
  – Better User Interfaces
Reduce Recruitment Time

- **Retainer Pool**
  - Pre-recruit a pool of crowd workers

  **Workers sign up in advance**
  
  - Get paid: 0.5 cent per minute
  - Wait at most: 5 minutes

  **Alert when task is ready**
  
  - Get paid:
  - alert()
  - Start now!
  - OK
  - 5 minutes
Classification of Latency Control

1. Single Task
   – Reduce the latency of a single task

2. Single Batch
   – Reduce the latency of a batch of tasks

3. Multiple Batches
   – Reduce the latency of multiple batches of tasks
Single-Batch Latency Control

- **Idea 1: Pricing Model**
  - Model the relationship between task price and completion time

- **Predict worker behaviors** [1,2]
  - Recruitment Time
  - Work Time

- **Set task price**
  - Fixed Pricing [2]
  - Dynamic Pricing [3]

---

[1]. Wang et al. Estimating the completion time of crowdsourced tasks using survival analysis models. CSDM 2011
Single-Batch Latency Control

- Idea 2: Straggler Mitigation
  - Replicate a task to multiple workers and return the result of the fastest worker
Classification of Latency Control

1. Single Task
   - Reduce the latency of a single task

2. Single Batch
   - Reduce the latency of a batch of tasks

3. Multiple Batches
   - Reduce the latency of multiple batches of tasks
Multiple-Batches Latency Control

Why multiple batches?
- To save cost
  - Answer Deduction (e.g., leverage transitivity)
  - Task Selection (e.g., active learning)
Multiple-Batches Latency Control

- **Two extreme cases**
  - Single task per batch: high latency
  - All tasks in one batch: high cost

- **Idea 1**
  - Choose the maximum batch size that does not hurt cost \(^1,2\)

- **Idea 2**
  - Model as a latency budget allocation problem \(^3\)

---

1. Jiannan Wang, Guoliang Li, Tim Kraska, Michael J. Franklin, Jianhua Feng: Leveraging transitive relations for crowdsourced joins. SIGMOD 2013
Summary of Latency Control

- **Latency**
  - The completion time of the last task

- **Classification of Latency Control**
  - Single-Task
    - Retainer Pool
    - Better UIs
  - Single-Batch
    - Pricing Model
    - Straggler Mitigation
  - Multiple-Batches
    - Batch size
Two Take-Away Messages

- **There is no free lunch**
  - Cost control
    - Trades off quality (or/and latency) for cost
  - Latency control
    - Trades off quality (or/and cost) for latency

- **Learn from other communities**
  - Task Design (from HCI)
  - Straggler Mitigation (from Distributed System)
Reference – Cost Control

Reference – Latency Control

Outline

- Crowdsourcing Overview (30min)
  - Motivation (5min)
  - Workflow (15min)
  - Platforms (5min)
  - Difference from Other Tutorials (5min)
- Fundamental Techniques (100min)
  - Quality Control (60min)
  - Cost Control (20min)
  - Latency Control (20min)
- Crowdsourced Database Management (40min)
  - Crowdsourced Databases (20min)
  - Crowdsourced Optimizations (10min)
  - Crowdsourced Operators (10min)
- Challenges (10min)
Why Crowdsourcing DB Systems

- Limitations of Traditional DB Systems

Table: car

<table>
<thead>
<tr>
<th>make</th>
<th>model</th>
<th>body_style</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volve</td>
<td>S80</td>
<td>Sedan</td>
<td>$10K</td>
</tr>
<tr>
<td>Volve</td>
<td>XC60</td>
<td>SUV</td>
<td>$20K</td>
</tr>
<tr>
<td>BMW</td>
<td>X5</td>
<td>SUV</td>
<td>$25K</td>
</tr>
<tr>
<td>?</td>
<td>Prius</td>
<td>Sedan</td>
<td>$15K</td>
</tr>
</tbody>
</table>

```
SELECT * FROM car WHERE make = "Toyota"
```

# of rows 0

Problem: Close world assumption
Why Crowdsourcing DB Systems

Limitations of Traditional DB Systems

Table: car_image

<table>
<thead>
<tr>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
</table>

```
SELECT * FROM car C, car_image M
WHERE M.make = C.make AND M.model = C.model AND M.color = "red"
```

# of rows: 0

Problem: Machine-hard tasks
Crowdsourcing DB Systems

- Integrating crowd functionality to DB
  - Close world $\rightarrow$ Open world
  - Processing DB-hard queries
Existing Crowd DB Systems

- **CrowdDB**
  - UC Berkeley & ETH Zurich

- **Qurk**
  - MIT

- **Deco**
  - Stanford

- **CDAS**
  - NUS

- **CDB**
  - Tsinghua
System Architecture

Crowdsourcing Query Optimization

Query Parser

Initial Plan

Query Optimizer

Optimized Plan

Query Executor

σ

R

S

Relational Data Model

Crowdsourced Data

Crowdsourcing Query

Requester

Crowdsourcing Query Language

Result

Query Processing

UI Templates

Parameters

HIT Manager

HIT 1

XXX

HIT 2

XXX

Crowdsourcing Operator Design

Crowdsourcing Platforms

Crowd Interaction

σ

R

S

Crowdsourced

Data
Running Example

car_review R1

<table>
<thead>
<tr>
<th>review</th>
<th>make</th>
<th>model</th>
<th>sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>Volvo</td>
<td>S80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...The 2014 Volvo S80 is the flagship model for the brand...</td>
</tr>
<tr>
<td>r2</td>
<td>Volvo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...S80 is a Volvo model having problems in oil pump..</td>
</tr>
<tr>
<td>r3</td>
<td>BMW</td>
<td>X5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...The BMW X5 is surprisingly agile for a big SUV..</td>
</tr>
</tbody>
</table>

car R2

<table>
<thead>
<tr>
<th>id</th>
<th>make</th>
<th>model</th>
<th>style</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>Volvo</td>
<td>S80</td>
<td>Sedan</td>
</tr>
<tr>
<td>a2</td>
<td>Toyota</td>
<td>Avalon</td>
<td>Sedan</td>
</tr>
<tr>
<td>a3</td>
<td>Volvo</td>
<td>XC60</td>
<td>SUV</td>
</tr>
<tr>
<td>a4</td>
<td>Toyota</td>
<td>Corolla</td>
<td>Sedan</td>
</tr>
<tr>
<td>a5</td>
<td>BMW</td>
<td>X5</td>
<td>SUV</td>
</tr>
<tr>
<td>a6</td>
<td>Toyota</td>
<td>Camry</td>
<td>Sedan</td>
</tr>
</tbody>
</table>

car_image R3

m1
m2
m3
m4
m5

Example Query:
Find black cars with high-quality images and positive reviews
Crowdsourcing DB Systems

- **System Overview**
  - CrowdDB
  - Qurk
  - Deco
  - CDAS
  - CDB

- **Operator Design**
  - Design Principles
CrowdB Query Language

CrowdSQL: Crowdsource missing data

Missing Columns

<table>
<thead>
<tr>
<th>review</th>
<th>make</th>
<th>model</th>
<th>sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx</td>
<td>Volvo</td>
<td>S80</td>
<td>?</td>
</tr>
</tbody>
</table>

Missing Tuples

<table>
<thead>
<tr>
<th>make</th>
<th>model</th>
<th>style</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

CREATE TABLE car_review
(
  review STRING,
  make CROWD STRING,
  model CROWD STRING,
  sentiment CROWD STRING
);

CREATE CROWD TABLE car
(
  make STRING,
  model STRING,
  color STRING,
  style STRING,
  PRIMARY KEY (make, model)
);
CrowdDB Query Language

- CrowdSQL: Crowdsourcing DB-hard tasks

**Crowd-powered Filtering**

```
SELECT review
FROM car_review
WHERE sentiment ~= "pos";
```

**Is the review positive?**

**Crowd-Powered Ordering**

```
SELECT image i
FROM car_image
WHERE subject = "Volvo S60"
ORDER BY CROWDORDER("clarity");
```

**Which one is better?**

The Volvo S80 is the flagship model of this brand…
CrowdDB Query Processing

- Crowd operators for data missing

```
SELECT *
FROM car C, car_review R
WHERE C.make = R.make AND C.model = R.model AND C.make = "Volvo"
```
CrowdDB Query Processing

- Crowd operators for DB-hard tasks

```sql
SELECT *
FROM company C1, company C2
WHERE C1.name ~= C2.name
```

```sql
SELECT *
FROM image M
ORDER BY CROWDORDER ("clarity")
```

Are the following entities the same?

IBM == Big Blue

- Yes
- No

Which picture visualizes better "Golden Gate Bridge"

- [ ] Yes
- [ ] No

Submit

CrowdCompare
CrowdDB Query Optimization

- **Strategy: Rule-based optimizer**
  - Pushing down selects
  - Determining join orders

\[ \sigma_{\text{make} = \text{"Volvo"}} \]

\[
\begin{align*}
\sigma & \quad \text{Make} = \text{"Volvo"} \\
\text{car} C & \quad \text{car} \_\text{review} R
\end{align*}
\]

Fill out the missing **Car review**

| review |  \\
|--------|  \\
| make |  \\
| model |  \\
| sentiment |  \\

Fill out the **Car** data

| make |  \\
|-------|  \\
| model |  \\
| color |  \\
| style | Volvo |
Crowdsourcing DB Systems

- **System Overview**
  - CrowdDB
  - Qurk
  - Deco
  - CDAS
  - CDB

- **Operator Design**
  - Design Principles
Qurk Query Language

- SQL with User-Defined Functions (UDFs)

```sql
SELECT i.image
FROM car_image i
WHERE isBlack(i)
```

**TASK** `isBlack(field)` **TYPE** Filter:

Prompt: "&lt;table&gt;&lt;tr&gt; 
&lt;td&gt;&lt;img src='%s'&gt;&lt;/td&gt; 
&lt;td&gt;Is the car in black color?&lt;/td&gt; 
&lt;/tr&gt;&lt;/table&gt;", tuple[field]

YesText: "Yes"
NoText: "No"
Combiner: MajorityVote
Qurk Query Processing

- Designing crowd-powered operators
  - Crowd Join: Designing better interfaces

![Image of cars comparing](https://via.placeholder.com/150)

**Simple Join**

- Is the same car in the two images?
  - Yes
  - No

**Naïve Batching**

- Is the same car in the two images?
  - Yes
  - No

**Smart Batching**

- Find pairs of images of the same car?
  - I did not find any pairs.
Qurk Query Processing

- Designing crowd-powered operators
  - Crowd Sort: Designing better interfaces

Rate the visualization of image

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>worst</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which one visualizes better?

- A is better
- B is better

Rating-Based Interface

Comparing-Based Interface
Qurk Query Optimization

- Join: Feature filtering optimization

```sql
SELECT *
FROM car_image M1 JOIN car_image M2
ON sameCar(M1.img, M2.img) AND
POSSIBLY make(M1.img) = make(M2.img) AND
POSSIBLY style(M1.img) = style(M2.img)
```

Filtering pairs with different makes & colors

- Is filtering feature always helpful?
  - Filtering cost vs. join cost
    - What if all cars has the same style
  - Causing false negatives, e.g., color
  - Disagreement among the crowd
Crowdsourcing DB Systems

- **System Overview**
  - CrowdDB
  - Qurk
  - Deco
  - CDAS
  - CDB

- **Operator Design**
  - Design Principles

Crowdsourcing Systems

Crowdsourcing Operators
Deco Query Language

- Conceptual Relation

  Car (make, model, [door-num], [style])

  - Anchor Attributes
  - Dependent Attribute-groups

- Raw Schema

  CarA (make, model)  // Anchor table
  CarD1 (make, model, door-num)  // Dependent table
  CarD2 (make, model, style)  // Dependent table

- Fetch Rules: How to collect data

  ∅ ⇒ make, model  // Ask for a new car
  make, model ⇒ door-num  // Ask for d-n of a given car
  make, model ⇒ style  // Ask for style of a given car
Deco Query Language

- Resolution rules

\[
\text{image} \Rightarrow \text{style}: \text{majority-of-3} \quad \text{// majority vote} \\
\varnothing \Rightarrow \text{make,model}: \text{dupElim} \quad \text{//eliminate duplicates}
\]

- Query

- Collecting style and color of at least 8 SUV cars

- SQL Query:

\[
\begin{align*}
\text{SELECT make, model, door-num, style} \\
\text{FROM Car} \\
\text{WHERE style = "SUV" MINTUPLES 8}
\end{align*}
\]

- Standard SQL Syntax and Semantics
- New keyword: MINTUPLES
Deco Query Processing

- Crowd Operator: Fetch

  - Fetch \([\emptyset \Rightarrow \text{ma, mo}]\)
  - Fetch \([\text{ma, mo} \Rightarrow \text{st}]\)
  - Fetch \([\text{ma, mo} \Rightarrow \text{dn}]\)

- Collect New Car

- Collect style of a given car

- Machine Operators
  - Scan: insert a collected tuple into raw table
  - Resolve: e.g., \text{majority-of-3}, \text{dupElim}
  - DLOJoin: traditional join
Deco Query Optimization

- **Example**
  - Current Status of the database

<table>
<thead>
<tr>
<th>CarA</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>model</td>
<td></td>
</tr>
<tr>
<td>Volve</td>
<td>S80</td>
<td></td>
</tr>
<tr>
<td>Toyota</td>
<td>Corolla</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>X5</td>
<td></td>
</tr>
<tr>
<td>Volvo</td>
<td>XC60</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CarD2</th>
<th>make</th>
<th>model</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>model</td>
<td>Style</td>
<td></td>
</tr>
<tr>
<td>Volve</td>
<td>XC60</td>
<td>SUV</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>X5</td>
<td>SUV</td>
<td></td>
</tr>
<tr>
<td>Volvo</td>
<td>S80</td>
<td>Sedan</td>
<td></td>
</tr>
</tbody>
</table>

- Selectivity of [style='SUV'] = 0.1
- Selectivity of dupElim = 1.0
- Each fetch incurs $0.05

- **How will a query be evaluated?**
Deco Query Processing

MinTuples
[8]

DLOJoin
[ma, mo]

Filter
[st=“SUV”]

DLOJoin
[ma, mo]

Resolve
[dupElim]

Scan
[CarA]

Resolve
[maj3]

Scan
[CarD1]

Resolve
[maj3]

Scan
[CarD2]

Fetch
[∅⇒ma, mo]

Fetch
[ma, mo⇒st]

Fetch
[ma, mo⇒dn]
Cost Estimation

- Let us consider a simple case
  - Resolve [dupElim]
    - Target: 8 SUV cars
    - DB: 2 SUV cars, 1 Sedan car, and 1 unknown car
    - Estimated: 2.1 SUV
  - Fetch
    - Target: \((8 - 2.1)\) SUV cars
    - Sel \([\text{style}=\text{‘SUV’}]\) = 0.1
    - Fetch 59 cars
  - Cost: \(59 \times \$0.05 = \$2.95\)
Deco Query Optimization

Better Plan: Reverse Query Plan

Reverse Plan incurs less cost in this query
Crowdsourcing DB Systems

- **System Overview**
  - CrowdDB
  - Qurk
  - Deco
  - CDAS
  - CDB

- **Operator Design**
  - Design Principles
CDAS Query Language

- SQL with Crowdsourcing on demand
  - Crowdsourcing when columns are unknown

```sql
SELECT c.*, i.image, r.review
FROM car_image i, car_review r
WHERE r.sentiment = "pos" AND i.color = "black"
AND r.make = i.make AND r.model = i.model
```

- Is the review matching with the image?
- The Vovlo S80 is the flagship model of this brand...
- Is the review positive?
- Is the car in black?
CDAS Query Processing

- Designing Crowd Operators
  - CrowdFill: filling missing values
  - CrowdSelect: filtering items
  - CrowdJoin: matching items from multiple sources

- Select Images
  - \( C_1: \text{make} = \ldots \)
  - \( C_2: \text{model} = \ldots \)
  - \( C_3: \text{style} = \ldots \)

  **Your Choice:**
  - Yes, it does
  - No, it doesn’t

- Join Image and Review
  - Conditions:
    - \( C_1: \text{make} \)
    - \( C_2: \text{model} \)

  **Your Choice:**
  - Yes
  - No

- Fill Car Attributes
  - **color** of car in the image:
    - 1: black
    - 2: red
    - 3: blue

  **Your Choice:**
CDAS Query Processing

- **Performance metrics**
  - Monetary cost: Unit price * # of HITs
  - Latency: # of crowdsourcing rounds

- **Optimization Objectives:**
  - Cost Minimization: finding a query plan minimizing the monetary cost
  - Cost Bounded Latency Minimization: finding a query plan with bounded cost and the minimum latency

- **Key Optimization Idea**
  - Cost-based query optimization
  - Balance the tradeoff between cost and latency
CDAS Query Optimization

- Cost-Latency Tradeoff

CSelect $\phi_4^s$
CSelect $\phi_3^s$
CSelect $\phi_2^s$
CSelect $\phi_1^s$

$R_3$

$C_4$: style = ”Sedan”
$C_3$: make = ”Volvo”
$C_2$: quality = ”high”
$C_1$: color = ”black”

How to balance cost-latency tradeoff?

Less cost, higher latency

More cost, lower latency
CDAS Query Optimization

- **How to implement Join**
  - CJoin: Compare every pairs
  - CFill: Fill missing join attributes

- **A Hybrid CFill-CJoin Optimization**

```sql
SELECT * FROM car R2, car_image R3
WHERE R2.make = R3.make AND R2.model = R3.model
```
Complex query optimization

- The latency constraint allocation problem

- Latency constraint $l$

- Latency constraint $L - l$

- Latency constraint $\bar{L} - l$

- Latency constraint $\bar{L}$

- Latency constraint $L$
Crowdsourcing DB Systems

- **System Overview**
  - CrowdDB
  - Qurk
  - Deco
  - CDAS
  - CDB

- **Operator Design**
  - Design Principles

Crowdsourcing Systems

Crowdsourcing Operators
CDB Query Language

- Collect Semantics
  - Fill Semantics
    
    ```
    FILL car_image.color
    WHERE car_image.make = "Volvo";
    ```
  
  - Collect Semantics
    
    ```
    COLLECT car.make, car.model
    WHERE car.style = "SUV";
    ```

- Query Semantics
  
  ```
  SELECT *
  FROM car_image M, car C, car_review R
  WHERE M.(make,model) CROWDJOIN C.(make,model)
  AND R.(make, model) CROWDJOIN C.(make,model)
  AND M.color CROWDEQUAL "red"
  ```
CDB Query Processing

- Graph-Based Query Model
  - Computing matching probabilities for each CROWDJOIN
  - Building a query graph that connects tuple pairs with matching probabilities larger than a threshold

```
car_review       car       car_image
```

```
+----------------+    +----------------+    +----------------+
|                |    |                |    |                |
|                |    |                |    |                |
|                |    |                |    |                |
|                |    |                |    |                |
|                |    |                |    |                |
```
CDB Query Processing

- Graph-Based Query Model
  - Crowdsourcing all edges (Yes/No tasks)
  - Coloring edges by the crowd answers
  - Result tuple: a path containing all CROWDJOINs

```
car_review  car  car_image
```

```
  car_image
      /        
  car      car_image
  / /       / /
car review car_image
```

“red”
CDB Query Optimization

- **Monetary cost control**
  - Traditional goal: finding an optimal join order
  - CDB goal: selecting **minimum number of edges**

Traditional: 2 tasks + 5 tasks + 1 task = 8 tasks

“red”
CDB Query Optimization

- Monetary cost control
  - Traditional goal: finding an optimal join order
  - CDB goal: selecting minimum number of edges

Traditional: 2 tasks + 5 tasks + 1 task = 8 tasks

CDB: 5 tasks

NP-HARD ➔ Various Heuristics
CDB Query Optimization

- Latency control
  - Partitioning the graph into connected components
  - Crowdsourcing each component in parallel
CDB Query Optimization

- Quality control
  - Probabilistic truth inference model
    \[ p_i = \frac{\prod_{(w,a) \in V_t} (q_w)^{1\{i=a\}} \cdot \left( \frac{1-q_w}{\ell-1} \right)^{1\{i \neq a\}}}{\sum_{j=1}^{\ell} \prod_{(w,a) \in V_t} (q_w)^{1\{j=a\}} \cdot \left( \frac{1-q_w}{\ell-1} \right)^{1\{j \neq a\}}} \]
  - Entropy-based task assignment model
    \[ \mathcal{I}(t) = \mathcal{H}(\vec{p}) - \sum_{i=1}^{\ell} \left[ p_i \cdot q_w + (1-p_i) \cdot \frac{1-q_w}{\ell-1} \right] \cdot \mathcal{H}(\vec{p}'). \]

- Other Task Types
  - Single-choice & Multi-choice tasks
  - Fill-in-blank tasks
  - Collection tasks
Take-Away for System Design

- Data Model
  - Relational model
  - Open world assumption

- Query Language
  - Extending SQL
  - Supporting interactions with the crowd

- Query Processing
  - Tree-based vs. Graph-based
  - Crowd-powered operators
  - Optimization: Quality, Cost, and Latency
Crowdsourcing DB Systems

- **System Overview**
  - CrowdDB
  - Qurk
  - Deco
  - CDAS
  - CDB

- **Operator Design**
  - Design Principles

Crowdsourcing Systems

Crowdsourcing Operators
Design Principles

- Leveraging crowdsourcing techniques
  - Quality Controlling
    - Truth Inference: inferring correct answers
    - Task Assignment: assigning tasks judiciously
  - Cost Controlling
    - Answer Deduction: avoiding unnecessary costs
    - Task Selection: selecting most beneficial tasks
  - Latency Controlling
    - Round Reduction: reducing # of rounds
  - Task Design
    - Interface Design: interacting with crowd wisely
Crowdsourced Selection

- **Objective**
  - Identifying items satisfying some conditions

- **Key Idea**
  - Task Assignment: cost vs. quality

Find all images containing SUV cars from an image set.

For each image:

- $(x,y)$: $x$ YES, $y$ No

**Truth Inference**
- Output PASS?
- Output FAIL?

**Task Assignment**
- Ask one more?
Crowdsourced Selection

- Key Idea
  - Latency Controlling: cost vs. latency

Find 2 images with SUV cars from 100 images

Sequential
  - C: 4 L: 4
  - Round 1
  - Round 2
  - Round 3
  - Round 4

Parallel
  - C: 100 L: 1
  - Round 1
  - ❌
  - ✅
  - ❌
  - ✅

Hybrid
  - C: 4 L: 3
  - Round 1
  - Round 2
  - Round 3
  - 👍

A. D. Sarma et al.: Crowd-powered find algorithms. ICDE 2014: 964-975
Crowdsourced Join

Objective
- Identifying record pairs referring to same entity

Key Idea
- Answer Deduction, e.g., using Transitivity

- Jiannan Wang, Guoliang Li, Tim Kraska, Michael J. Franklin, Jianhua Feng: Leveraging transitive relations for crowdsourced joins. SIGMOD 2013
- Donatella Firmani, Barna Saha, Divesh Srivastava: Online Entity Resolution Using an Oracle. PVLDB 2016
Crowdsourced Join

- **Key Idea**
  - Task Selection, e.g., selecting **beneficial** tasks

- One task deduced
- No task deduced

- Jiannan Wang, Guoliang Li, Tim Kraska, Michael J. Franklin, Jianhua Feng: Leveraging transitive relations for crowdsourced joins. SIGMOD 2013
Crowdsourced TopK/Sort

- **Objective**
  - Finding top-\(k\) items (or a ranked list) wrt. Criterion

- **Key Idea**
  - Truth Inference: Resolve conflicts among crowd

Which picture visualizes the best SFU Campus?

Pair-wise Voting

- **A**: A > B
- **B**: 2
- **C**: 1
- **D**: 3

- **Ranking Inference over conflicts among crowd**
  - Max Likelihood Inference
  - NP-hard

S. Guo et al.: So who won?: dynamic max discovery with the crowd. SIGMOD Conference 2012: 385-396
Crowdsourced TopK/Sort

Key Idea

- Task Selection: Most beneficial for getting the top-k results

What are the top-2 picture that visualizes the best SFU Campus?

Rank by computers

The most beneficial task: Difficult to computers vs. Xiaohang Zhang, Guoliang Li, Jianhua Feng: Crowdsourced Top-k Algorithms: An Experimental Evaluation. PVLDB 2016
Crowdsourced Collection

- **Objective**
  - Collecting a set of new items

- **Key Idea**
  - **Truth Inference:** Inferring item coverage

- **Species Estimation Algo.**
  - Observing the rate at which new species are identified over time
  - Inferring how close to the true number of species you are

---

B. Trushkowsky et al.: Crowdsourced enumeration queries. ICDE 2013: 673-684
Crowdsourced Collection

- **Key Idea**
  - Task Assignment: satisfying result distribution

- **Diverse distributions among workers**
  - E.g., collecting movies with publishing decades

- **Worker 1**
  - Old Fashioned

- **Worker 2**
  - New Fashioned

---

J. Fan et al.: Distribution-Aware Crowdsourced Entity Collection. TKDE 2017
Crowdsourced Fill

- **Objective**
  - Filling missing cells in a table

- **Key Idea: Task Design**
  - Microtask vs. partially-filled table with voting
  - Real-Time collaboration for concurrent workers
  - Compensation scheme with budget

<table>
<thead>
<tr>
<th>name</th>
<th>nationality</th>
<th>position</th>
<th>caps</th>
<th>goals</th>
<th>name</th>
<th>nationality</th>
<th>position</th>
<th>caps</th>
<th>goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lionel Messi</td>
<td>Argentina</td>
<td>FW</td>
<td>83</td>
<td></td>
<td>Empty</td>
<td>Empty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ronaldinho</td>
<td>Brazil</td>
<td>MF</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neymar</td>
<td>Brazil</td>
<td>FW</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iker Casillas</td>
<td>Spain</td>
<td>FW</td>
<td>150</td>
<td>0</td>
<td>Empty</td>
<td>Empty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ronaldinho</td>
<td>Brazil</td>
<td>FW</td>
<td>Empty</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Crowdsourced Count

- **Objective**
  - Estimating number of certain items

- **Key Idea**
  - Task Design: Leveraging crowd to estimate

How many are female? 2
# Take-Away for Crowd Operators

<table>
<thead>
<tr>
<th></th>
<th>CrowdSelect</th>
<th>CrowdJoin</th>
<th>CrowdSort</th>
<th>CrowdCollect</th>
<th>CrowdFill</th>
<th>CrowdCount</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Truth Inference</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Task Assignment</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Answer Deduction</strong></td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Task Selection</strong></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Round Reduction</strong></td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Interface Design</strong></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## System Comparison

<table>
<thead>
<tr>
<th>Crowd Powered Operators</th>
<th>CrowdDB</th>
<th>Qurk</th>
<th>Deco</th>
<th>CDAS</th>
<th>CDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrowdSelect</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdJoin</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdSort</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdTopK</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdMax</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdMin</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdCount</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdCollect</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CrowdFill</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## System Comparison

<table>
<thead>
<tr>
<th>Optimization</th>
<th>CrowdDB</th>
<th>Qurk</th>
<th>Deco</th>
<th>CDAS</th>
<th>CDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Latency</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Quality</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objectives</th>
<th>CrowdDB</th>
<th>Qurk</th>
<th>Deco</th>
<th>CDAS</th>
<th>CDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth Inference</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Task Assignment</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Answer Reasoning</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Task Design</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Latency Reduction</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
16. Adam Marcus, David R. Karger, Samuel Madden, Rob Miller, Sewoong Oh: Counting with the Crowd. PVLDB 2012.
Outline

◦ Crowdsourcing Overview (30min)
  – Motivation (5min)
  – Workflow (15min)
  – Platforms (5min)
  – Difference from Other Tutorials (5min)

◦ Fundamental Techniques (100min)
  – Quality Control (60min)
  – Cost Control (20min)
  – Latency Control (20min)

◦ Crowdsourced Database Management (40min)
  – Crowdsourced Databases (20min)
  – Crowdsourced Optimizations (10min)
  – Crowdsourced Operators (10min)

Challenges (10min)
The 6 Crowdsourcing Challenges

- Benchmarking
- Scalability
- Truth Inference
- Privacy
- Macro-Tasks
- Mobile Crowdsourcing
1. Benchmarking

- Database Benchmarks
  - TPC-C, TPC-H, TPC-DI, ...

- Crowdsourcing
  - No standard benchmarks

- Existing public datasets (link) are inadequate
1. Benchmarking

- Existing public datasets are inadequate, because:
  - Each task often receives 5 or less answers
  - Most tasks are single-label tasks
  - Very few numeric tasks
  - Lack ground truth
    - Expensive to get ground truth for 10K tasks
2. Scalability

- Hard to Scale in Crowdsourcing to tackle the 3Vs of Big Data?

- (1) workers are expensive;
  (2) answers can be erroneous;
  (3) existing works focus on specific problems, e.g., active learning (Mozafari et al. VLDB14), entity matching (Gokhale et al. SIGMOD14).
2. Scalability: Query Optimization

- Query Processing in Traditional RDBMS

1. **Parser**
2. **Logical Query Plan**
3. **Query Rewriter**
4. **Physical Query Plan**
5. **Query Optimization**
2. Scalability: Query Optimization

- Query optimization in crowdsourcing is challenging:
  
  1. handle 3 optimization objectives
  2. humans are more unpredictable than machines
3. Truth Inference

- Not fully solved (Zheng et al. VLDB17)

- We have surveyed 20+ methods:
  
  1. No best method;

  2. The oldest method (David & Skene JRSS 1979) is the most robust;

  3. No robust method for numeric tasks (the baseline “Mean” performs the best !)
4. Privacy

(1) **Requester**

Wants to protect the **privacy of their tasks** from workers

e.g., tasks may contain sensitive attributes, e.g., *medical data.*
4. Privacy

(2) Workers

Want to have privacy-preserving requirement & worker profile

e.g., personal info of workers can be inferred from the worker’s answers, e.g., location, gender, etc.
5. Macro-Tasks

- Existing works focus on simple micro-tasks

Is Bill Gates currently the CEO of Microsoft?
- Yes  - No

Identify the sentiment of the tweet: ……
- Pos  - Neu  - Neg

- Hard to perform big and complex tasks, e.g., writing an essay

(1) macro-tasks are hard to be split and accomplished by multiple workers;
(2) workers may not be interested to perform a time-consuming macro-task.
6. Mobile Crowdsourcing

- Emerging mobile crowdsourcing platforms e.g., gMission (HKUST), ChinaCrowd (Tsinghua)

- Challenges
  1. Other factors (e.g., spatial distance, mobile user interface) affect workers’ latency and quality;
  2. Different mechanisms: traditional crowdsourcing platforms: workers request tasks from the platform; for mobile crowdsourcing platform: only workers close to the crowdsourcing task can be selected.
Thanks!

Q & A

Guoliang Li  Yudian Zheng  Ju Fan  Jiannan Wang  Reynold Cheng
Tsinghua University  Hong Kong University  Renmin University  SFU  Hong Kong University