Uncertain Data Management
Applications of Uncertain Data

Antoine Amarilli[1], **Silviu Maniu**[2]

[1]Télécom ParisTech

[2]Université Paris-Sud

January 11th, 2016

Web Crawling

Obtaining Data on the Web

Crawling: the operation of obtaining a "picture" of the pages on the Web.

Obtaining Data on the Web

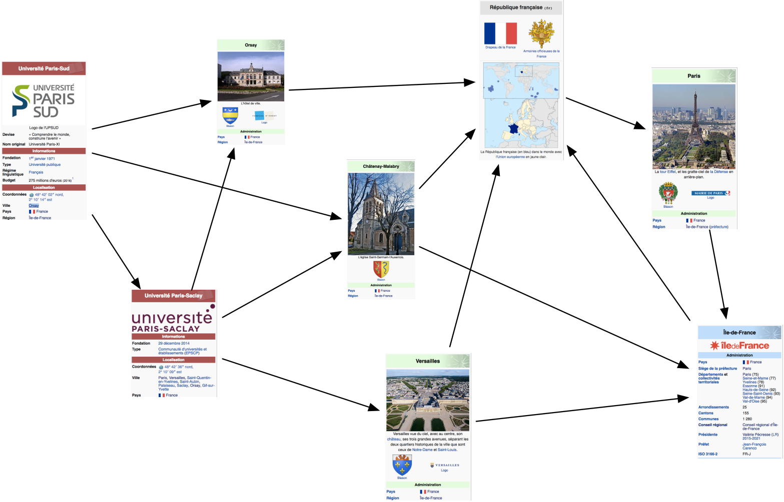
Crawling: the operation of obtaining a "picture" of the pages on the Web.

An iterative process:

1. get a set of pages on the Web called seeds, and process their outgoing links,
2. for each outgoing link, extract it from the Web and process its outgoing links,
3. repeat step 2 until no pages are left.

The set of pages to be processed is called the frontier.

Crawling: Illustration



Focused Crawling

When we have a budget and objective – focused crawling:

- budget – limited Web API calls (Twitter, Foursquare, Facebook), limited money
- objective – crawl only the news related to a subject, obtain the pages that are relevant to a query, etc.

Applications: Web crawling, deep Web mining, social network querying, peer-to-peer gossip.

Algorithms for Focused Crawling

As opposed to classical crawling (BFS is enough), there must be a way to estimate the worth of each node to be crawled.

Algorithms for Focused Crawling

As opposed to classical crawling (BFS is enough), there must be a way to estimate the worth of each node to be crawled.

Estimation algorithm amount to probabilistic processing: estimating the worth of each node (topic centered PageRank), or probabilistically choosing the best nodes (multi-armed bandits).

PageRank

Estimating Node Worth

Nodes on the Web: pages (sites, Wikipedia, ...), users (Twitter, Facebook), etc.

Estimating Node Worth

Nodes on the Web: pages (sites, Wikipedia, ...), users (Twitter, Facebook), etc.

To crawl/use the nodes that are more “interesting” than others, we have to estimate the worth of each node.

PageRank: the algorithm used by Google to rank pages on the Web.

PageRank: Ranking Nodes in A Graph

Definition 1: The important nodes are the nodes that are linked to by other important nodes (recursive).

PageRank: Ranking Nodes in A Graph

Definition 1: The important nodes are the nodes that are linked to by other important nodes (recursive).

Definition 2 – the random surfer model, where the surfer walks on the graph:

1. the surfer starts at a node (*e.g.*, Google) and chooses randomly an outgoing node (*e.g.*, a page in the search results),
2. the surfer behaves in the same manner for other nodes,
3. at each step the surfer has a probability $1 - \alpha$ (damping factor) of jumping elsewhere randomly.

The importance of a page $=$ the stationary probability that the surfer is on a page at time ∞.

PageRank: Ranking Nodes in A Graph

Definition 1: The important nodes are the nodes that are linked to by other important nodes (recursive).

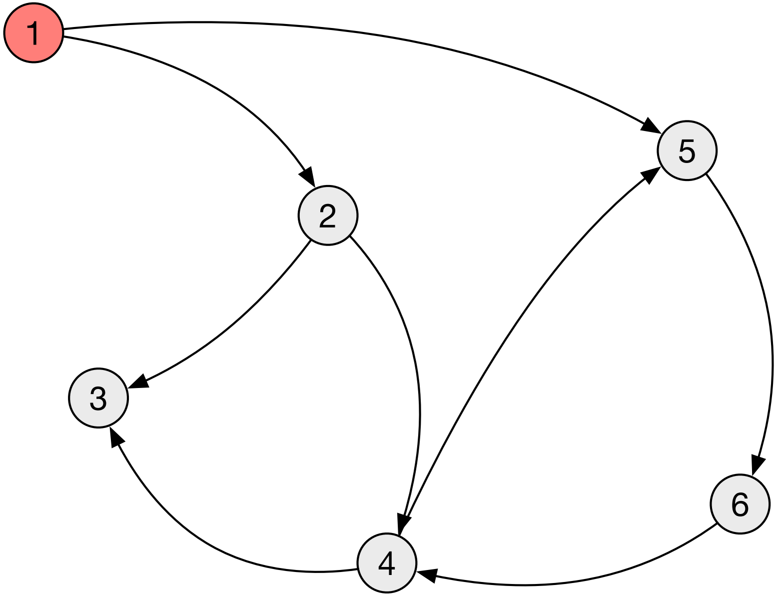
Definition 2 – the random surfer model, where the surfer walks on the graph:

1. the surfer starts at a node (*e.g.*, Google) and chooses randomly an outgoing node (*e.g.*, a page in the search results),
2. the surfer behaves in the same manner for other nodes,
3. at each step the surfer has a probability $1 - \alpha$ (damping factor) of jumping elsewhere randomly.

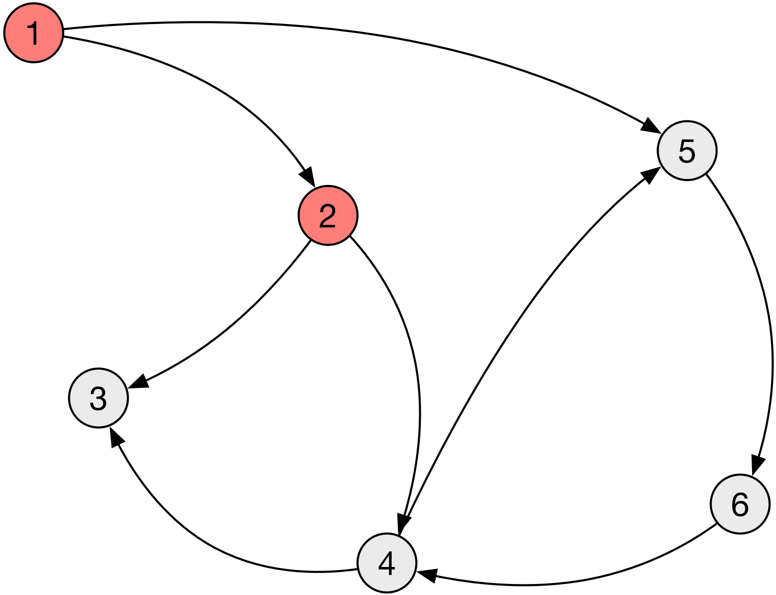
The importance of a page $=$ the stationary probability that the surfer is on a page at time ∞.

The two definitions are equivalent.

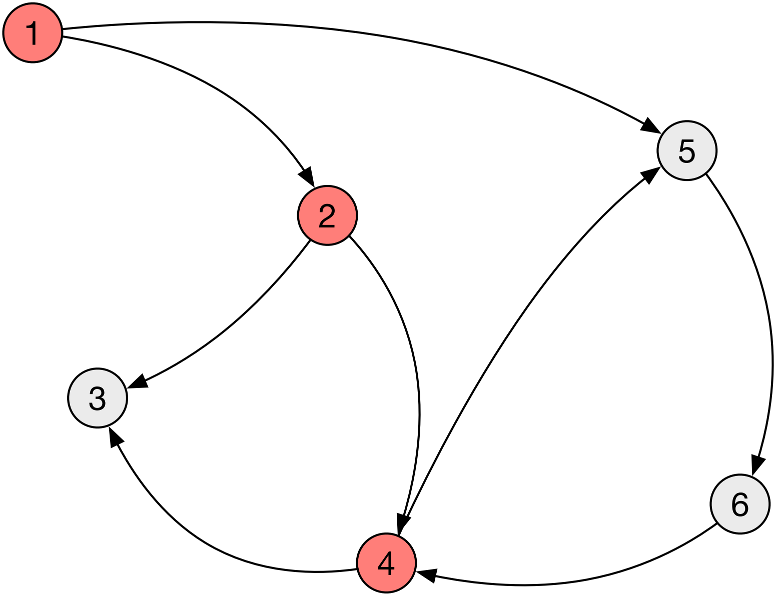
PageRank: Random Surfer



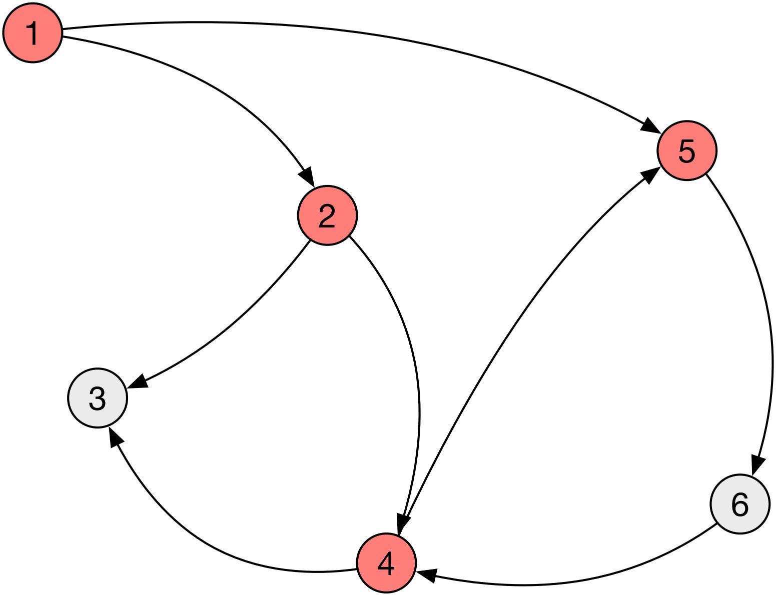
PageRank: Random Surfer



PageRank: Random Surfer



PageRank: Random Surfer



PageRank Equation and Algorithm

For a graph G with n nodes, where each node i has the incoming neighbours I_i and outgoing neighbours O_i:

$$p(i) = \alpha \sum_{j \in I_i} \frac{p(j)}{|O_j|} + \frac{1-\alpha}{n}.$$

PageRank Equation and Algorithm

For a graph G with n nodes, where each node i has the incoming neighbours I_i and outgoing neighbours O_i:

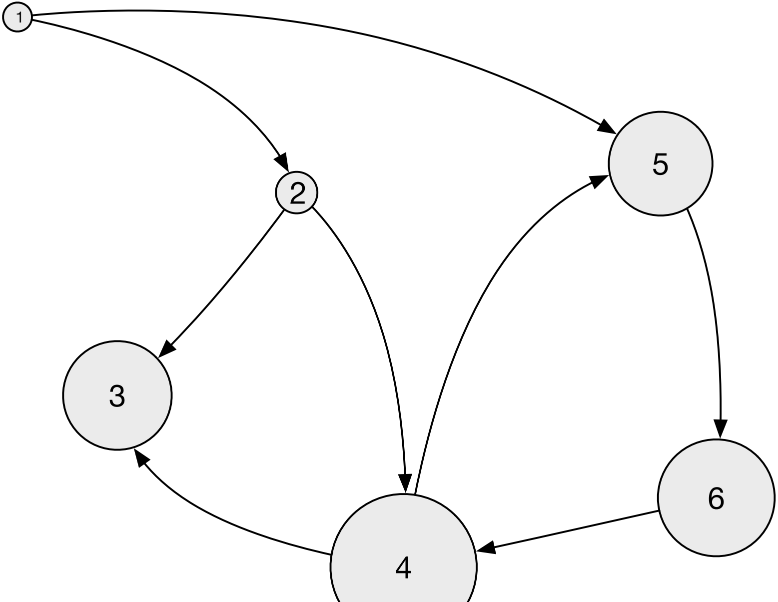
$$p(i) = \alpha \sum_{j \in I_i} \frac{p(j)}{|O_j|} + \frac{1-\alpha}{n}.$$

Algorithm for computing $p(i)$:

1. start with initial values of $p(i) = \frac{1}{n}$,
2. iteratively apply the equation for each node i,
3. stop when the probabilities converge (stationary).

Monte-Carlo approximation: simulate N walks and take $p(i) = \frac{v_i}{N}$, where v_i number of visits of page i among N walks.

PageRank: The Final Graph



Variants of PageRank

Depending where the surfer teleports with probability $1 - \alpha$, we have different variants of PageRank:

- classic PageRank: the surfer can jump to any node.
- personalized PageRank: the surfer can only jump to their start page.
- topic-sensitive PageRank: the surfer can only jump to a set of same-topic pages.

Variants of PageRank

Depending where the surfer teleports with probability $1 - \alpha$, we have different variants of PageRank:

- classic PageRank: the surfer can jump to any node.
- personalized PageRank: the surfer can only jump to their start page.
- topic-sensitive PageRank: the surfer can only jump to a set of same-topic pages.

For focused crawling, one can use topic-sensitive PageRank – one has to estimate the values for each node during the crawl.

Influence Maximization

Social Influence

Social Influence: important problem in social network, with applications in marketing, computational advertising

Social Influence

Social Influence: important problem in social network, with applications in marketing, computational advertising

Objective: given a promotion budget of k social network users, maximize the expected influence spread given some influence or propagation model

Social Influence

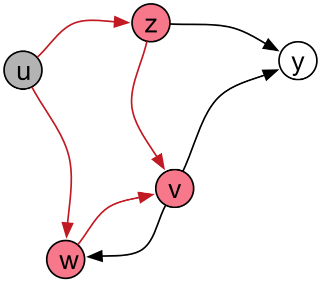
Data Model: an uncertain graph $G(V, E, p)$

Social Influence

Data Model: an uncertain graph $G(V, E, p)$

- V and E are the social network
- p is, on each edge, the influence probability

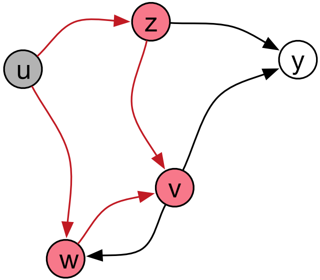
Influence Spread via Cascades



Independent Cascade Model:
discrete time model of propagation

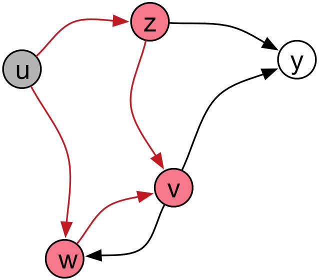
1. at time 0, activate seed u
2. for a node i activated at time t: activate at time $t+1$ each neighbour v with probability p_{iv}
3. once a node is activated, it cannot be activated again or de-activated

Influence Spread via Cascades



We wish to compute the expected spread from a seed seed set S, $\sigma(S)$

Influence Spread via Cascades



We wish to compute the expected spread from a seed seed set S, $\sigma(S)$

By linearity of expectation:

$$\sigma(u) = \sum_{v \in V} \Pr(u \to v)$$

- for a seed set S, more complicated
- same hardness as reachability

Maximizing the Influence

Influence maximization is computationally hard

Two sources of hardness:

1. computing $\sigma(S)$ is #P-hard (as we seen before, it is equivalent to reachability) – Monte Carlo with additive approximations
2. computing the selection of k seeds in S is NP-hard – maximization of a submodular function

Maximizing the Influence

Influence maximization is computationally hard
Two sources of hardness:

1. computing $\sigma(S)$ is #P-hard (as we seen before, it is equivalent to reachability) – Monte Carlo with additive approximations
2. computing the selection of k seeds in S is NP-hard – maximization of a submodular function

Submodular function: the influence spread is submodular:

$$\sigma(S \cup \{u\}) - \sigma(S) \geqslant \sigma(T \cup \{u\}) - \sigma(T) \quad \text{if} \quad S \subseteq T$$

Influence Maximization: Greedy Algorithm

We can obtain a $(1 - \frac{1}{\epsilon})$-approximation factor for influence maximization by using the greedy algorithm

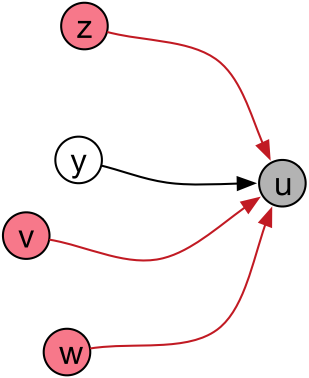
Influence Maximization: Greedy Algorithm

We can obtain a $(1 - \frac{1}{\epsilon})$-approximation factor for influence maximization by using the greedy algorithm

Steps:

1. initialize $S = \emptyset$
2. choose the user u that maximizes $\sigma(S \cup \{u\}) - \sigma(S)$
3. $S = S \cup u$
4. repeat steps 2 and 3 k times
5. **return** S

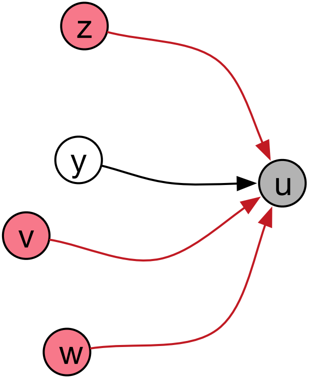
Learning Propagation Probabilities



The probability that v is influenced by its neighbours

$$\Pr(v) = 1 - \prod_{u} (1 - p_{uv})$$

Learning Propagation Probabilities



The probability that v is influenced by its neighbours

$$\Pr(v) = 1 - \prod_{u} (1 - p_{uv})$$

Given a log of actions $A = \{(\text{act}, u, v), \dots\}$:

1. maximum likelihood: $p_{vu} = \frac{A_{vu}}{A_v}$
2. Jaccard similarity: $p_{vu} = \frac{A_{vu}}{A_{u|v}}$

References

1. D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," KDD, 2003, pp. 137–146.
2. A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," WSDM, 2010, pp. 241–250.

Crowdsourcing

Some tasks cannot be performed effectively by computers
(*Which?*)

Some tasks cannot be performed effectively by computers (*Which?*)

Crowdsourcing: asking the answers to data from Internet workers, and not from computers

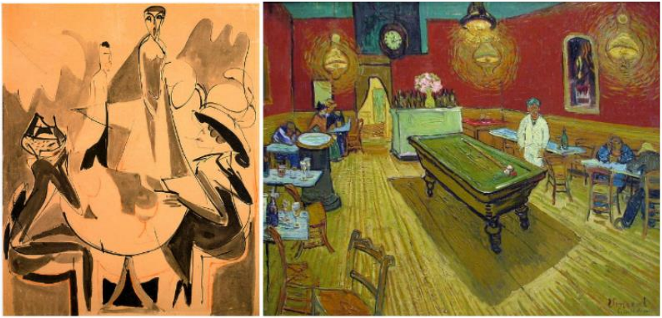
Some tasks cannot be performed effectively by computers (*Which?*)

Crowdsourcing: asking the answers to data from Internet workers, and not from computers

Applications:

- image recognition
- entity resolution
- data cleaning

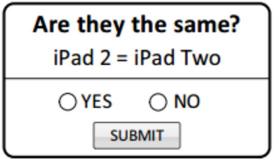
Image Recognition

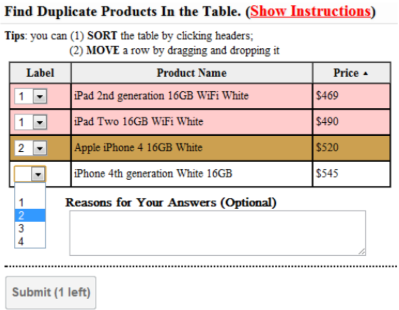


How similar is the artistic style in the paintings above?

- Very similar
- Somewhat similar
- Neither similar nor dissimilar
- Somewhat dissimilar
- Very dissimilar

Entity Resolution



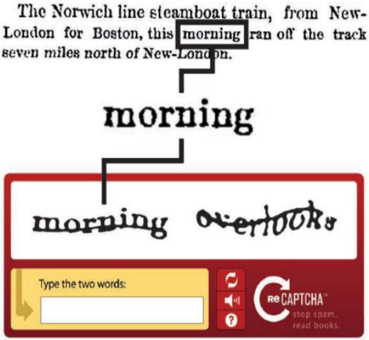


CAPTCHA

CAPTCHA

Completely Automated Public Turing test to tell Computers and Humans Apart

ReCAPTCHA



Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham and Manuel Blum. ReCAPTCHA: Human-Based Character Recognition via Web Security Measures. Science, 321: 1465-1468, 2008

Crowdsourcing on the Internet

Crowdsourcing Terms

Workers: users, bloggers, Merchanical Turk workers
Requesters: persons who need their data cleaned or need new knowledge

Crowdsourcing Terms

Workers: users, bloggers, Merchanical Turk workers

Requesters: persons who need their data cleaned or need new knowledge

Tasks – also known as HITs (human interface tasks): questions, comments, Wikipedia edits,

Crowdsourcing Terms

Workers: users, bloggers, Merchanical Turk workers
Requesters: persons who need their data cleaned or need new knowledge

Tasks – also known as HITs (human interface tasks): questions, comments, Wikipedia edits,

Incentives: usually money, but can be reputation, recognition in the community

Tasks

Types of tasks:

- binary questions: is Paris the capital of France?
- open questions: what is the address of Télécom?
- comparisons: which image is "better"

Data Issues in Crowdsourcing

Answers from crowds are unreliable, due to the workers' answers
Why?

Data Issues in Crowdsourcing

Answers from crowds are unreliable, due to the workers' answers
Why?

- the workers' answers have to be biased by their reliability (*how to measure?*)
- the data has to be stored and processed in databases (*what kinds of databases?*)

Data Issues in Crowdsourcing

Answers from crowds are unreliable, due to the workers' answers
Why?

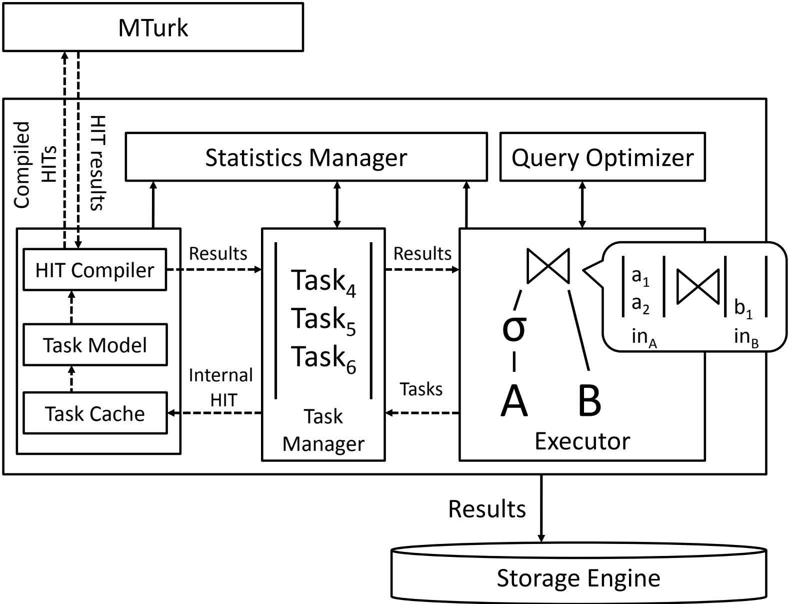
- the workers' answers have to be biased by their reliability (*how to measure?*)
- **the data has to be stored and processed in databases** (*what kinds of databases?*)

Qurk

For tasks on Amazon Mechanical Turk, they can be expressed as an workflow:

- SQL queries on the data existing in the database
- UDFs (User Defined Functions) on missing data

Qurk


MTurk
Compiled HITs
HIT results
Statistics Manager
Query Optimizer
HIT Compiler
Task Model
Task Cache
Results
Internal HIT
Task4
Task5
Task6
Task Manager
Results
Tasks
σ
A
B
Executor
a1
a2
inA
b1
inB
Results
Storage Engine

Qurk

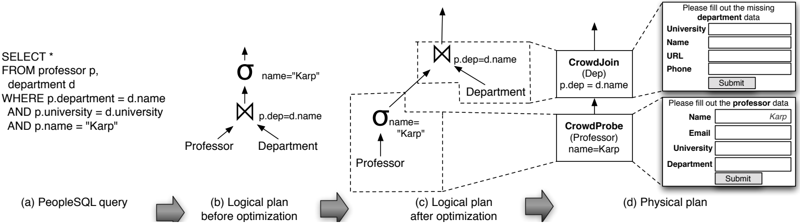
Users give different and conflicting answers – *how can we solve this?*

Qurk

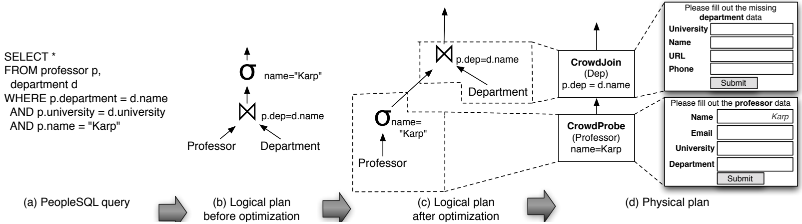
Users give different and conflicting answers – *how can we solve this?*

- Qurk uses resolution rules, such as majority voting

CrowdDB

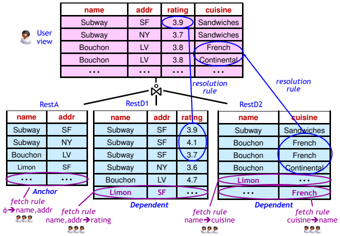

SELECT *
FROM professor p,
department d
WHERE p.department = d.name
AND p.university = d.university
AND p.name = "Karp"
(a) PeopleSQL query
σ name="Karp"
⋈ p.dep=d.name
Professor
Department
(b) Logical plan before optimization
⋈ p.dep=d.name
Department
σ name= "Karp"
Professor
(c) Logical plan after optimization
CrowdJoin
(Dep)
p.dep = d.name
CrowdProbe
(Professor)
name=Karp
Please fill out the missing department data
University
Name
URL
Phone
Submit
Please fill out the professor data
Name
Karp
Email
University
Department
Submit
(d) Physical plan

CrowdDB



- same principle as Qurk, but allows for the generation of new tuples

Deco



- separation between crowd and user views
- defines fetch and resolution rules
- fetch: how data is obtained from the crowd
- resolution: how data is aggregated

Data Issues in Crowdsourcing

Answers from crowds are unreliable, due to the workers' answers
Why?

Data Issues in Crowdsourcing

Answers from crowds are unreliable, due to the workers' answers
Why?

- **the workers' answers have to be biased by their reliability** (*how to measure?*)
- the data has to be stored and processed in databases (*what kinds of databases?*)

Simple Aggregation Rules

Resolution Rules: aggregating the answer from the crowd

Simple Aggregation Rules

Resolution Rules: aggregating the answer from the crowd

What is the capital of France?

worker	answer
Anne	Paris
Richard	Lyon
Jean	Lyon
Pauline	Paris
Benoit	Paris

Simple Aggregation Rules

Resolution Rules: aggregating the answer from the crowd

What is the capital of France?

worker	answer
Anne	Paris
Richard	Lyon
Jean	Lyon
Pauline	Paris
Benoit	Paris

Aggregation rules: majority vote, average, ...

Simple Aggregation Rules

Resolution Rules: aggregating the answer from the crowd

What is the capital of France?

worker	answer
Anne	Paris
Richard	Lyon
Jean	Lyon
Pauline	Paris
Benoit	Lyon

Simple Aggregation Rules

Resolution Rules: aggregating the answer from the crowd

What is the capital of France?

worker	answer
Anne	Paris
Richard	Lyon
Jean	Lyon
Pauline	Paris
Benoit	Lyon

In some cases aggregation rules can fail

Simple Aggregation Rules

Resolution Rules: aggregating the answer from the crowd

What is the capital of France?

worker	answer
Anne	Paris
Richard	Lyon
Jean	Lyon
Pauline	Paris
Benoit	Lyon

Assume that Anne and Pauline give correct answers in 90% of the cases, and Richard, Jean and Benoit only in 50% of the cases – what is the correct answer?

Worker Accuracy

Let us assume labelling questions, where each worker needs to give an answer with only one true value

A simple model: a worker w_i has accuracy π_i – a probability of π_i to give the correct answer and a probability of $1 - \pi_i$ to give the incorrect one

Worker Accuracy

Let us assume labelling questions, where each worker needs to give an answer with only one true value

A simple model: a worker w_i has accuracy π_i – a probability of π_i to give the correct answer and a probability of $1 - \pi_i$ to give the incorrect one

How to get the worker accuracies?:

- estimate their accuracy on a set of control questions
- sometimes, possible to do it without any ground truth input

Example of Crowdsourced Worker Accuracy

worker	Italy	France	U.K.	Spain
Anne	Rome	Paris	London	Madrid
Jean	Milan	Paris	London	Madrid
Pauline	Milan	Lyon	Manchester	Barcelona

Example of Crowdsourced Worker Accuracy

worker	**Italy**	**France**	**U.K.**	**Spain**
Anne	Rome	Paris	London	Madrid
Jean	Milan	Paris	London	Madrid
Pauline	Milan	Lyon	Manchester	Barcelona

What is the correct answer? – **truth discovery**

Truth Discovery in Crowdsourcing

Assume a set of k facts in $\{0, 1\}$, a set of n workers w_i

Every worker answer for every fact:

$$\boldsymbol{a} = \{a_{11}, \cdots, a_{1n}, \cdots, a_{kn}\}$$

Each worker has an accuracy π_i which is the probability that they answer 1 correctly

We want to derive the labels/answers, $\boldsymbol{l}$

Maximum Likelihood

A standard approach to optimize probabilities – computing the likelihood given the answers:

Maximum Likelihood

A standard approach to optimize probabilities – computing the likelihood given the answers:

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\phi} \mid \boldsymbol{a}) = \prod_{i}^{n} \prod_{j}^{w} \phi_i^{l_i} (1 - \phi_i)^{1 - l_i} \pi_j^{y_{ij}} (1 - \pi_j)^{1 - y_{ij}}$$

where

$$y_{ij} = a_{ij} l_i + (1 - a_{ij})(1 - l_i)$$

Maximum Likelihood

A standard approach to optimize probabilities – computing the likelihood given the answers:

$$\mathcal{L}(\boldsymbol{\pi}, \phi \mid \boldsymbol{a}) = \prod_{i}^{n} \prod_{j}^{w} \phi_i^{l_i} (1 - \phi_i)^{1-l_i} \pi_j^{y_{ij}} (1 - \pi_j)^{1-y_{ij}}$$

where

$$y_{ij} = a_{ij} l_i + (1 - a_{ij})(1 - l_i)$$

We want to estimate $\boldsymbol{\pi}$ and ϕ by maximizing the likelihood

Maximum Likelihood

Maximizing it gives us the following estimates

$$\hat{\phi}_i = \frac{\sum_j^n a_{ij}\pi_j + \sum_j^n (1 - a_{ij})(1 - \pi_j)}{n}$$
$$\hat{\pi}_i = \frac{\sum_i^k a_{ij}\phi_i + \sum_i^k (1 - a_{ij})(1 - \phi_j)}{k}$$

Maximum Likelihood Estimation (MLE)

The estimations are recursively defined – to maximize it, we can use the EM algorithm:

1. initialize the facts and the worker accuracies (assume workers are 100% accurate)
2. estimation (E-step) estimate the labels l_i based on the probabilties $\hat{\phi}_i$
3. maximization (M-step) compute the worker and fact probabilities based on the labels
4. iterate 2 and 3 until convergence

Example of Crowdsourced Worker Accuracy

worker	**Italy**	**France**	**U.K.**	**Spain**
Anne	Rome	Paris	London	Madrid
Jean	Milan	Paris	London	Madrid
Pauline	Milan	Lyon	Manchester	Barcelona

Exercise: What is the correct answer?

Using BID Databases

country	capital	answers
France	Paris	7
France	Lyon	3
Italy	Rome	5

0.7

country	capital
France	Paris
Italy	Rome

0.3

country	capital
France	Lyon
Italy	Rome

Using BID Databases

country	**capital**	**prob**
France	Paris	0.7
France	Lyon	0.3
Italy	Rome	1

0.7

country	**capital**
France	Paris
Italy	Rome

0.3

country	**capital**
France	Lyon
Italy	Rome

Add a `REPAIR-KEY` construct to SQL to transform raw answers to probabilistic databases

Using BID Databases

To answer queries like *What is the correct capital of country X?*
we can add a `WHILE` operator / fixpoint operator

Using BID Databases

To answer queries like *What is the correct capital of country X?* we can add a `WHILE` operator / fixpoint operator

- this will result in a Markov chain of instances, for which we need to compute the stationary distribution for a class of queries
- this is a known #P-hard problem

Using BID Databases

To answer queries like *What is the correct capital of country X?* we can add a `WHILE` operator / fixpoint operator

- this will result in a Markov chain of instances, for which we need to compute the stationary distribution for a class of queries
- this is a known #P-hard problem

Approximation:

- additive approximation is PTIME
- multiplicative approximation is NP-hard

References

1. A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing Systems on the World-Wide Web," Comm ACM, vol. 54, no. 4, pp. 86–96, Apr. 2011.
2. M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "CrowdDB: answering queries with crowdsourcing," SIGMOD, 2011, pp. 61–72.
3. H. Park, H. Garcia-Molina, R. Pang, N. Polyzotis, A. Parameswaran, and J. Widom, "Deco: a system for declarative crowdsourcing," PVLDB, vol. 5, no. 12, pp. 1990–1993, Aug. 2012.
4. A. P. Dawid and A. M. Skene, "Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm," Journal of the Royal Statistical Society, vol. 28, no. 1, pp. 20–28, 1979.
5. Q. Liu, M. Steyvers, and A. Ihler, "Scoring Workers in Crowdsourcing How Many Control Questions are Enough?," NIPS, 2013.