Controlling False Discoveries During Interactive Data Exploration

Problem:
Visual analytics & data exploration are prone to false discovery.

Challenge:
Infer hypotheses from visualizations automatically.
Control false discovery interactively with theoretical guarantee.

System:
QUDE: Quantifying Uncertainties in Data Exploration.
Demo: Safe Visual Data Exploration

Session 11, Continental B, 11am.

Z Zhao, L De Stefani, E Zgraggen, C Binnig, E Upfal and T Kraska
Computer Science Department, Brown University
MacroBase: Prioritizing Attention in Fast Data
Peter Bailis; Edward Gan; Samuel Madden; Deepak Narayanan; Kexin Rong; Sahaana Suri

Problem:
Massive fast data streams :: i) relational analytics not enough
ii) no ML systems for automatically reducing streams at scale

Solution:
New engine combining streaming classification + explanation
This paper: architecture, unsupervised estimation, sketching
Exciting results from production and open source use

This is the next major challenge for dataflow-based analytics
Data Canopy: Accelerating Exploratory Statistical Analysis

Statistics are everywhere! Repetitive statistics and data access

Data Canopy synthesizes statistics from basic ingredients

Abdul Wasay | Xinding Wei | Niv Dayan | Stratos Idreos
Problem: training a probabilistic model from sampled answers to Boolean queries

Main Contribution: a novel, tuple-independent probabilistic DB with Beta priors
Previous

40 Years of DB research: Repetition of "Query → Answer"

Our Work

DB Learning: More queries → Fast query processing
(ML: More observation → More Accurate)

*The first work that becomes faster every time by exploiting the answers to past queries*
Staging User Feedback toward Rapid Conflict Resolution in Data Fusion

Romila Pradhan       Siarhei Bykau       Sunil Prabhakar

<table>
<thead>
<tr>
<th>Data Item</th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zootopia</td>
<td>Howard</td>
<td>Spencer</td>
<td>Spencer</td>
<td></td>
</tr>
<tr>
<td>Kung Fu Panda</td>
<td>Stevenson</td>
<td>Nelson</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inside Out</td>
<td>leFauve</td>
<td>Docter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finding Dory</td>
<td></td>
<td>Stanton</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minions</td>
<td>Coffin</td>
<td>Renaud</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rio</td>
<td>Jones</td>
<td>Saldanha</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Predicted Truth
- Spencer
- Nelson
- Docter
- Stanton
- Coffin
- Saldanha

Ground Truth
- Howard
- Stevenson
- Docter
- Stanton
- Coffin
- Saldanha

How to be most effective with user feedback?
Discovering Your Selling Points: Personalized Social Influential Tag Exploration

Yuchen Li; Ju Fan; Dongxiang Zhang; Kian-Lee Tan
We propose a reduction strategy, scalable algorithm, and analysis framework for Coarsening Massive Influence Networks for Scalable Diffusion Analysis.

Naoto Ohsaka, Tomohiro Sonobe, Sumio Fujita, Ken-ichi Kawarabayashi

Solution for the input graph

Solution for the small graph

Accurancy guarantee 1 hour for billion edges 2–30× faster
Debunking the **Myths** of Influence Maximization

**Myths and Mis-claims**
- Bugs and undocumented assumptions

**Categorization**
- Quality
- Memory Footprint
- Efficiency

"One Size Doesn’t Fit All"
No Single State of the Art!

Akhil Arora* | Sainyam Galhotra* | Sayan Ranu
Interactive Mapping Specification with Exemplar Tuples

Give me a few tuples, I'll get you a mapping

A. Bonifati, U. Comignani, E. Coquery, R. Thion
Motivation

Current data transformation methods:
1) consume too much user effort
2) require high expertise
3) programming-skills

Results
1. Save about 60% of user effort than state-of-art Programming By Demonstration data transformation tool; requires little expertise.
2. Can handle about 90% real-world data transformation tasks

Programming-By-Example approach synthesizing data transformation programs

“Somewhat” Structured Raw Data

Synthesized Program
QIRANA: A Framework for Scalable Query Pricing
Shaleen Deep, Paris Koutris
University of Wisconsin-Madison

Problem: Design scalable pricing framework
- increased demand for data markets
- need for pricing systems with formal guarantees

Proposed Solution: QIRANA
- scalable
- arbitrage-free data pricing framework
- allows customizability for seller
- history-aware pricing

Talk@Session 14, Lake Michigan
Access Path Selection in Main-Memory Optimized Data Systems:

Should I Scan or Should I Probe?

Scan

Index

Scan is best

Index is best

Concurrent Read Queries

Selectivity

Historical Division

Our Proposal

Michael S. Kester  Manos Athanassoulis  Stratos Idreos
Problem: Optimization of disjunctive predicates

\[(p_{1,1} \land \ldots \land p_{1,n}) \lor \ldots \lor (p_{m,1} \land \ldots \land p_{m,n})\]

Current optimization schemes for disjunctive predicates are based on heuristics which produce poor plans:
- Query is transformed into either DNF or CNF and then optimized.

When optimizing disjunctive predicates, the true optimization potential cannot be achieved by means of traditional plans:
- We can fill this gap by means of Bypass Processing.

We propose a new algorithm which exploits Bypass processing:
- Our experiments show an improvement in plan quality by an average factor of over 2000 vs. heuristics used in RDBMSs.
A Top-Down Approach to Achieving Performance Predictability in Database Systems

Jiamin Huang; Barzan Mozafari; Grant Schoenebeck; Thomas F. Wenisch

Q1. How to automatically identify root causes of performance variance in a complex codebase?

Q2. How to make database systems more predictable but also faster?

Q3. How our techniques improved MySQL and are deployed on 2M+ servers today?
Teaser Talks (Second Part)
Problem: Estimate join size
  ● Selection predicates given at query time

Our solution: Sampling
  ○ Two-Level Sampling
  ○ One Pass, Unbiased, Smaller Error
  ○ Beats previous sampling methods

Applications
  ○ Query optimization (join order)
  ○ Approximate query answering (COUNT, SUM, AVG)
By the end of my talk I hope to convince you to replace all your Bloom filters with a new data structure, the counting quotient filter (CQF).

- Problem: Bloom filters lack features that many applications need.
  - Can’t count, merge, resize, delete, scale out of RAM, etc.
  - Applications are forced to work around the limitations.
- Solution: Counting quotient filter (CQF).
  - Supports counting, merging, scaling out of RAM, etc.
  - Counts skewed input distributions efficiently.
  - Faster and smaller than a Bloom filter.
- Several computational biology and streaming applications teams are already replacing Bloom filters with CQFs in their code.
BePI: Fast and Memory-Efficient Method for Billion-Scale Random Walk with Restart

Jinhong Jung; Namyong Park; Sael Lee; U Kang

Seoul National Univ. and The State Univ. of New York Korea

Problem: Random Walk with Restart
- RWR measures relevance scores between nodes in graphs
- How can we compute RWR scores quickly in very large graphs?

Proposed method: BePI
- Fast and scalable by taking the advantages of both preprocessing and iterative approaches

Experimental Results
- Process 100x larger graphs and requires 130x less memory space than existing preprocessing methods
- Compute RWR scores up to 9x faster than its competitors
Determining the Impact Regions of Competing Options in Preference Space

1 Bo Tang; 2 Kyriakos Mouratidis; 1 Man Lung Yiu
1 The Hong Kong Polytechnic Univ., 2 Singapore Management Univ.

Who are our target customer?

How about our market share?

Where am I most competitive?

(Session 17, 14:00-14:25, Continental C)
Regret-ratio minimizing set can serve for (approximately) answering maxima queries when convex hull is large.

We make several fundamental theoretical as well as practical advances in developing such a compact set.

- In 2D: we develop an optimal linearithmic time algorithm by leveraging the ordering of skyline tuples.
- In HD: we develop an approximation algorithm that runs in linearithmic time and guarantees a regret ratio, within any arbitrarily small user-controllable distance from the optimal regret ratio.
FEXIPRO: Fast and Exact Inner Product Retrieval in Recommender Systems

1 Hui Li; 2 Tsz Nam Chan; 2 Man Lung Yiu; 1 Nikos Mamoulis
1 The University of Hong Kong and 2 Hong Kong Polytechnic University

- **Problem:** top-k inner product retrieval
  - Matrices \( Q \) and \( P \) come from matrix factorization
  - Large \( q^T p \) indicates a possible recommendation

- **Our Framework FEXIPRO**
  - Use Thin SVD, Integer Approximation and Monotonicity Reduction to manipulate data
  - Orthogonal to existing systems

- **Experiments**
  - At least one order of magnitude faster than existing methods
  - Single-thread FEXIPRO is faster than multi-thread Intel MKL and requires much less memory
Feedback-Aware Social Event-Participant Arrangement
Jieying She; Yongxin Tong; Lei Chen; Tianshu Song

● Background & Motivation
  ○ Event arrangement in EBSN
  ○ Existing studies
    ■ The satisfaction scores are hard to learn
    ■ Users may not accept the arrangements

● Solutions
  ○ Multi-arm Bandit (MAB) based framework
    ■ Thompson Sampling based solution
    ■ Upper Confidence Bound (UCB) based solution

● Experiments
  ○ The **Thompson Sampling** based solution does not perform well under FASEA
  ○ The **UCB** based solution is the best in overall by extensive experiments on both real and synthetic datasets
Exploiting Common Patterns for Tree-Structured Data
Zhiyi Wang Shimin Chen
(Institute of Computing Technology, Chinese Academy of Sciences)

- Supports tree-structured data: e.g., JSON, Protocol Buffers, etc.
- Exploits real-world data pattern: simple path optimization
- Achieves $10^{1000}$X speedup compared to state-of-the-art systems

STEED: System for Tree structured Data
Graph-structured data can enable analyses impossible using SQL analytics

Problem: When extracting graphs from RDBMSs, graphs often orders-of-magnitude larger than the input tables. Graph may not fit in memory

Solution: A software layer (GraphGen) over the database that efficiently loads in a condensed-representation, and enables efficient processing through various APIs.
TrillionG: A Trillion-scale Synthetic Graph Generator using a Recursive Vector Model
Himchan Park; Min-Soo Kim

● Motivation
  ○ Large-scale realistic graphs for benchmarks, e.g., RMAT, Kronecker
  ○ Low-level core techniques for rich graph generation, e.g., LDBC, gMark

● Solutions
  ○ A Vertex Scope approach (AVS)
  ○ Recursive Vector Model (RecVec)

● Experiments
  ○ More realistic graphs by adding noises
  ○ Schema-driven rich graph
  ○ A trillion edges graph within 2 hours using just 10 PCs
Mission: Find any sign of life on Earth.

People represent same data using different schemas.

We want to learn same accurate answers over all possible schemas for the same data.

The result of current learning algorithms depend on the schema.

Castor: schema independent, accurate and efficient. It leverages concepts of schema design.

<table>
<thead>
<tr>
<th>boxes</th>
<th>box</th>
<th>item</th>
<th>color</th>
<th>desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bomb</td>
<td>green</td>
<td>solid</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>plant</td>
<td>blue</td>
<td>wet</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>contains</th>
<th>box</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bomb</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>plant</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color</th>
<th>box</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>green</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>blue</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>description</th>
<th>box</th>
<th>desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>solid</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>wet</td>
<td></td>
</tr>
</tbody>
</table>

```
life(x) :- boxes(y,x,z,wet).
life(x) :- contains(y,x), color(y,z), description(y,wet).
```
Scalable Kernel Density Classification using Threshold-Based Pruning
Edward Gan; Peter Bailis

Complex Distributions
Expensive Estimators
Classification Results

ML + Predicate Pushdown: Asymptotic (1000x) Speedups
The BUDS Language for Distributed Machine Learning
Zekai “Jacob” Gao; Shangyu Luo; Luis Perez; Chris Jermaine

Our results: the BUDS optimized implementations have competitive performance compared to the hand-coded SQL implementations.
Improving the Life of a Data Scientist

Zoi Kaoudi; Jorge Quiane; Sara Thirumuruganathan; Sanjay Chawla; Divy Agrawal

What people think he does
What he thinks he does
What he actually does

What people think he does

What he thinks he does

What he actually does

Best performance without sacrificing accuracy

ML TASK

Model

RUN CLASSIFICATION ON training.txt

![Image of data scientist's role and tasks]

Data scientist

Select algorithm

Best performance without sacrificing accuracy

Hyperparameter tuning

Implementation

Big Data

Select classifier

Clustering/classification

K-means

DBScan

Hierarchical

Expectation

Mean-shift

K-nearest neighbours

Random forests

Decision trees

Naive Bayes

Logistic regression

Neural networks

Drop learning

Best performance without sacrificing accuracy

Training time (sec)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MLLib</th>
<th>SystemML</th>
<th>ML4All</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult</td>
<td>29</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>covtype flagged</td>
<td>45</td>
<td>30</td>
<td>16</td>
</tr>
<tr>
<td>higgs</td>
<td>92</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>svm1</td>
<td>511</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>svm2</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>
The integer array compression problem

- Bitmap compression
- Inverted list compression

Which is better?
OtterTune

Automatic Database Management System Tuning Through Large-scale Machine Learning

OtterTune leverages past experiences to tune new DBMS configurations.

OtterTune outperforms tuning scripts and Amazon RDS.

Can OtterTune match a DBA?

Dana Van Aken; Andrew Pavlo; Geoffrey J. Gordon; Bohan Zhang
Solving the Join Ordering Problem via Mixed Integer Linear Programming
Immanuel Trummer; Christoph Koch

Transforming the Join Ordering Problem into a MILP Problem

Query Optimizer → Optimal Order

Transform

MILP Solver → MILP Solution

Optimizing Queries with Up to 40 Tables ...
Problem: Once storage and compute are decoupled the I/O bottleneck moves to the network.

- Offload redo processing from compute
- Purpose-built scale-out multi-tenant log-structured distributed storage service designed for databases
- Storage volume striped across hundreds of nodes over 3 availability zones (AZ)
- Six copies of data, two in each AZ to protect against correlated AZ+1 failures

Result: Better durability, availability, & jitter … … which are all the same thing (on different time scales)