New Trends on Exploratory Methods for Data Analytics

Davide Mottin, Matteo Lissandrini,
Yannis Velegrakis, Themis Palpanas
Who we are

Davide Mottin
Graph Mining, Novel Query Paradigms, Interactive Methods
https://hpi.de/en/mueller/team/davide-mottin.html

Matteo Lissandrini
Knowledge Graphs, Novel Query Paradigms, Graph Mining
https://disi.unitn.it/~lissandrini

Yannis Velegrakis
Big Data Management & Analytics, Information Integration
https://velgias.github.io

Themis Palpanas
Data Series Indexing & Mining, Data Management, Data Analytics
http://www.mi.parisdescartes.fr/~themisp/

Big data – Easy value?
Exploring

Traditional

On data
Data exploration

Cleaning and profiling

Visualization

Analysis

D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
Data exploration software

Tableau: analysis and statistics

Trifacta

OpenRefine: data preparation and cleanup
Traditional data exploration methods

Efficiently extracting knowledge from data even if we do not know exactly what we are looking for

```
SELECT avg(system-stars)
FROM Universe
WHERE system-stars > 10
GROUP BY galaxy
```

Not easy for novices
Declarative Exploratory methods

Simple query (exploratory)

SELECT galaxy_name
FROM Universe.Galaxy

Over generic
100 billions results

Complex query
(for data experts)

SELECT g.galaxy_name, SUM(s.stars) as st_s
FROM Universe.Galaxy AS g
JOIN Universe.Systems AS s
ON g.galaxy_name = s.galaxy_name
WHERE
  g.st_s > 100B
  AND diameter > 100k AND diameter > 180k
  AND has_black_hole = TRUE
GROUP BY g.galaxy_name

Specific
Few results

SELECT * FROM Universe.Galaxy
WHERE...

Specific
Few results
Examples as Exploratory Methods

Is there a galaxy like this?

Answers
Historical perspective: Query-by-example

Specify a query by example tables, or skeletons.

<table>
<thead>
<tr>
<th>Name</th>
<th>Stars</th>
<th>Diameter</th>
<th>Black_hole</th>
<th>Color</th>
<th>Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>P._</td>
<td>&gt; 10B</td>
<td>&gt;100k</td>
<td>TRUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;180k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Intuitive GUI for simple queries
- SQL not required
- Restricted to SQL semantics
- Not example-based
Tutorial’s goals

- Exploratory methods using examples
- Algorithms for retrieving data without using query languages
- Interactive methods and user-in-the-loop feedback
- Machine learning for adaptive, online methods

But NOT

- Declarative query methods
- User interfaces and visualization
- Optimizations for fast data access
- Dynamic data
Tutorial structure

- Relational databases (25 min)
- Textual data (10 min)
- Graph and networks (25 min)
- Challenges and Remarks

Challenges and Remarks (10 min)
Example-based methods

- Query suggestion using examples
- Reverse engineering queries

- Entity extraction by example text
- Web table completion using examples
- Search by example

- Community-based Node-retrieval
- Entity Search
- Path and SPARQL queries
- Graph structures as Examples

D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
Where we are

Relational databases

Textual data

Graphs and networks

Challenges and Remarks

D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
Reverse engineering queries (REQ)

Given a set of examples, find the query that generated that set of tuples

Example tuples

How do you find such queries?

SELECT galaxy_name
FROM Universe.Galaxy

SELECT g.galaxy_name, SUM(s.stars) AS st_s
FROM Universe.Galaxy AS g
JOIN Universe.System AS s
ON g.galaxy_name = s.galaxy_name
WHERE
  g.st_s > 100B
  AND diameter > 100k AND diameter > 180k
  AND has_black_hole = TRUE
GROUP BY g.galaxy_name
Reverse engineering queries (REQ)

**Exact**
- One-shot
  - Query by output - TALOS
  - REQ SPJ queries from examples
- Interactive
  - Query From examples (QFE)
  - Interactive inference of join queries

**Approximate**
- Minimal
  - Discovering Queries based on Examples
- Top-k
  - S4: Top-k Spreadsheet style
Main idea: Find the set of queries that exactly return a set of examples

Two queries Q and Q’ are instance equivalent on a database D, if the results of Q are the same as the results of Q’.
TALOS

(a) Master

<table>
<thead>
<tr>
<th>pID</th>
<th>name</th>
<th>country</th>
<th>weight</th>
<th>bats</th>
<th>throws</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>A</td>
<td>USA</td>
<td>85</td>
<td>L</td>
<td>R</td>
</tr>
<tr>
<td>P2</td>
<td>B</td>
<td>USA</td>
<td>72</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>P3</td>
<td>C</td>
<td>USA</td>
<td>80</td>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>P4</td>
<td>D</td>
<td>Germany</td>
<td>72</td>
<td>L</td>
<td>R</td>
</tr>
<tr>
<td>P5</td>
<td>E</td>
<td>Japan</td>
<td>72</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

(b) Batting

<table>
<thead>
<tr>
<th>pID</th>
<th>year</th>
<th>stint</th>
<th>team</th>
<th>HR</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>2001</td>
<td>2</td>
<td>PIT</td>
<td>40</td>
</tr>
<tr>
<td>P1</td>
<td>2003</td>
<td>2</td>
<td>ML1</td>
<td>50</td>
</tr>
<tr>
<td>P2</td>
<td>2001</td>
<td>1</td>
<td>PIT</td>
<td>73</td>
</tr>
<tr>
<td>P2</td>
<td>2002</td>
<td>1</td>
<td>PIT</td>
<td>40</td>
</tr>
<tr>
<td>P3</td>
<td>2004</td>
<td>2</td>
<td>CHA</td>
<td>35</td>
</tr>
<tr>
<td>P4</td>
<td>2001</td>
<td>3</td>
<td>PIT</td>
<td>30</td>
</tr>
<tr>
<td>P5</td>
<td>2004</td>
<td>3</td>
<td>CHA</td>
<td>60</td>
</tr>
</tbody>
</table>

(c) Team

<table>
<thead>
<tr>
<th>team</th>
<th>year</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIT</td>
<td>2001</td>
<td>7</td>
</tr>
<tr>
<td>PIT</td>
<td>2002</td>
<td>4</td>
</tr>
<tr>
<td>CHA</td>
<td>2004</td>
<td>3</td>
</tr>
</tbody>
</table>

Input

- B: PIT
- E: CHA

Join graph computation

\[ J = \text{Master} \Join \text{Batting} \Join \text{Team} \]

<table>
<thead>
<tr>
<th>name</th>
<th>bat</th>
<th>throw</th>
<th>stint</th>
<th>HR</th>
<th>team</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>A</td>
<td>L</td>
<td>R</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>t_2</td>
<td>A</td>
<td>L</td>
<td>R</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>t_3</td>
<td>C</td>
<td>R</td>
<td>L</td>
<td>2</td>
<td>35</td>
</tr>
<tr>
<td>t_4</td>
<td>D</td>
<td>L</td>
<td>R</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>t_5</td>
<td>B</td>
<td>R</td>
<td>R</td>
<td>1</td>
<td>73</td>
</tr>
<tr>
<td>t_6</td>
<td>B</td>
<td>R</td>
<td>R</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>t_7</td>
<td>E</td>
<td>R</td>
<td>R</td>
<td>3</td>
<td>60</td>
</tr>
</tbody>
</table>
Idea: treat the problem as a binary classification

1. **Strict**: all tuples must be captured
2. **At-Least-one**: one tuple for example must be captured

\[
Gini(S) = 1 - (f^2 + f^-^2)
\]

Positive and negative tuples in S

\[
Gini(S_1, S_2) = \frac{|S_1|Gini(S_1) + |S_2|Gini(S_2))}{|S_1| + |S_2|}
\]
How complex is exact REQ?

Relational Operators:
- σ selection \{=, \neq, \geq, \leq\}
- π projection
- \bowtie natural join

Database \( D \)

\( E^+ \) Positive examples
\( E^- \) Negative examples

REQ

\( Q \) such that results contain
- All positive examples
- No negative example

How difficult is to find:
- A bounded size \( Q \)?
- An unbounded \( Q \)?
## Complexity - No parameters

[Weiss et al., 2017]

<table>
<thead>
<tr>
<th>Operator</th>
<th>Unbounded Queries</th>
<th>Bounded Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\pi)</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(\bowtie)</td>
<td>P</td>
<td>NPC</td>
</tr>
<tr>
<td>(\sigma)</td>
<td>P</td>
<td>NPC</td>
</tr>
<tr>
<td>(\sigma, \bowtie)</td>
<td>P</td>
<td>NPC</td>
</tr>
<tr>
<td>(\pi, \sigma)</td>
<td>NPC</td>
<td>NPC</td>
</tr>
<tr>
<td>(\sigma, \bowtie)</td>
<td>DP</td>
<td>DP</td>
</tr>
<tr>
<td>(\pi, \sigma, \bowtie)</td>
<td>DP</td>
<td>DP</td>
</tr>
</tbody>
</table>

### Complexity Results

- Only projections: **Easy**
- Unbounded selections: **Easy**
- Unbounded selections: **HARD**
- Combination of operators: **HARD!!!**
### Unbounded Select

#### Possible queries?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>✗</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>✗</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>✓</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>✓</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

A = 1   AND
B ≥ 1   AND   B ≤ 5   AND
C ≥ 2   AND   C ≤ 4   AND
D ≥ 1   AND   D ≤ 4   AND   D ≠ 4
E ≥ 3   AND   E ≤ 5   AND   E ≠ 4

[Weiss et al., 2017]
**Bounded select**

**INPUT:** Database $D$, Examples $E$, Query size $k$

**OUTPUT:** Does there exist a query satisfying $D$ and $E$, of size at most $k$?

$U = \{1,2,3,4,5\}$  
$S = \{ \{1,2,3\}, \{2,4\}, \{3,4\}, \{4,5\} \}$

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
# Complexity - Parameters

<table>
<thead>
<tr>
<th>No param</th>
<th>Schema Example</th>
<th>No param Query</th>
<th>Schema Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\pi)</td>
<td>(P)</td>
<td>(P)</td>
<td>(P)</td>
</tr>
<tr>
<td>(\land)</td>
<td>(P)</td>
<td>(NPC \quad \frac{P}{W[2]C})</td>
<td>(FPT) (NPC)</td>
</tr>
<tr>
<td>(\sigma)</td>
<td>(P)</td>
<td>(NPC \quad \frac{P}{W[2]C})</td>
<td>({=} : P, {\neq} : NPC \quad {=} : FPT)</td>
</tr>
<tr>
<td>(\sigma, \land)</td>
<td>(P)</td>
<td>(NPC \quad \frac{P}{W[2]C})</td>
<td>({=} : P, {\neq} : NPC \quad {=} : FPT)</td>
</tr>
<tr>
<td>(\pi, \sigma)</td>
<td>(NPC \quad {=} : P, {\neq} : NPC \quad \frac{\pi}{W[3]C})</td>
<td>(NPC \quad \frac{P}{W[3]C})</td>
<td>({=} : P, {\neq} : NPC \quad {=} : FPT)</td>
</tr>
</tbody>
</table>
Main idea: Interactively remove candidate queries proposing a new set of query results from a modified database

Use QBO

Reverse engineered Queries Q’

Query Results

REQ

Database Refinement

Modified database and results

[Li et al., 2015]

D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
Database Refinement

[Li et al., 2015]

Results

REQs =
- $Q_1 = \sigma_{gender=M}(D)$
- $Q_2 = \sigma_{salary>3700}(D)$
- $Q_3 = \sigma_{dept=IT}(D)$
Cost model

\[
\text{cost}(D') = \text{edit}(D, D') + \beta \cdot n + \sum_{i=1}^{k} \text{edit}(R, R_i) + N \cdot \frac{\text{edit}(D, D')}{\mu} + \beta + \frac{2}{k} \sum_{i=1}^{k} \text{edit}(R, R_i)
\]

Main idea: Find a refined db D’ and results \( R_1, \ldots, R_k \) with:
1. Minimum number of results k
2. Minimum differences i the database
3. The query are balanced (less interactions)

[Li et al., 2015]
Minimal Project Join REQ

Main idea: Find the set of queries that approximately return a set of examples

Partial query table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mike</td>
<td>ThinkPad</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>iPad</td>
</tr>
<tr>
<td>3</td>
<td>Bob</td>
<td>Dropbox</td>
</tr>
</tbody>
</table>

Minimal PJ

Queries Q’

- **valid**: every tuple is present in query results
- **minimal**: any removal in query tree gets to an invalid query
Candidate Query Generation

- Use candidate network generation algorithm (Hristidis 2002)

1. Generate join tree $J$
2. Generate mapping $\phi$
3. Check minimal:
   - Every leaf node contains a column that is mapped by an input column

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mike</td>
<td>ThinkPad</td>
<td>Office</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>iPad</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Bob</td>
<td>Dropbox</td>
<td></td>
</tr>
</tbody>
</table>

[Shen et al., 2014]
Validity verification

Naïve: check all candidate queries singularly if they return ALL examples

Better: exploit substructures in candidate queries for pruning

Best: adaptively select the substructures to have the min number of evaluations

NP-hard

[Shen et al., 2014]
Minimal Project Join REQ

Main idea: Allow missing rows/columns and rank the k best queries

Partial query table

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>2</td>
<td>Jill</td>
<td>Hans</td>
</tr>
</tbody>
</table>

Output: Top-k PJ Queries
Ranking score

Linear combination of row score and column score

\[
\alpha \cdot score_{row}(Q) + (1 - \alpha) \cdot score_{col}(Q)
\]

\[\frac{Q}{|Q|}\]

- \(\alpha = 1\) penalizes missing rows
- \(\alpha = 0\) penalizes missing columns

Row score

<table>
<thead>
<tr>
<th>Name</th>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
<td>Xbox</td>
</tr>
<tr>
<td>Jill</td>
<td>Hans</td>
<td>Surface</td>
</tr>
</tbody>
</table>

Column score

<table>
<thead>
<tr>
<th>Name</th>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
<td>Xbox</td>
</tr>
<tr>
<td>Jill</td>
<td>Hans</td>
<td>Surface</td>
</tr>
</tbody>
</table>

\[\alpha \cdot \text{score}_{row}(Q) + (1 - \alpha) \cdot \text{score}_{col}(Q)\]

\[\frac{Q}{|Q|}\]

\[\alpha \cdot \text{score}_{row}(Q) + (1 - \alpha) \cdot \text{score}_{col}(Q)\]

\[\frac{Q}{|Q|}\]

\[\alpha \cdot \text{score}_{row}(Q) + (1 - \alpha) \cdot \text{score}_{col}(Q)\]

\[\frac{Q}{|Q|}\]

\[\alpha \cdot \text{score}_{row}(Q) + (1 - \alpha) \cdot \text{score}_{col}(Q)\]

\[\frac{Q}{|Q|}\]
S4 Optimizations

Upper bound
Row score is always bounded by the column score
(row containment is more restrictive)
Exploit inverted indexes on columns/rows

Early termination
Stop when current upper bound score is less than the k-th ranked evaluated query
Scan queries on decreasing upper bound

Caching
Reuse common subparts in the candidate queries
Reverse engineering queries (REQ)

Exact

One-shot
- Query by output - TALOS
- REQ SPJ queries from examples

Interactive
- Query From examples (QFE)
- Interactive inference of join queries

Approximate

Minimal
- Discovering Queries based on Examples

Top-k
- S4: Top-k Spreadsheet style

Lack of user models!
Main idea: Allow interactive navigation of the query space in a hierarchy
Examples for query suggestion: Blaeu

[Sellam et al., 2016]

Given a result of an example query Q, explore the data through data maps = partitions

Output: Set of query refinements

Problem: User utility is unknown

- Cluster analysis for result exploration
- Zoom and projection operations
- User model

User utility

\[ u: DB \rightarrow \{-1,1\}, U(Q) = \sum_{t \in Q} u(t) \]
Examples for query suggestion: Blaeu

Find the partition $\mathcal{C} = \{C_1, ..., C_n\}$ of the results of $Q$ such that exists $C_j \in \mathcal{C}: U(C_j) > U(Q)$

**Solution:** interesting tuples are close to each other within a maximum separation threshold $\theta(\mathcal{C})$
Where we are

Relational databases

Textual data

Graphs and networks

Challenges and Remarks

Machine learning
Examples for textual data

Entity Extraction [Hanafi 2017]

Web table completion [Yakout 2013]

Search by example
  - Serendipitous search [Bordino 2013]
  - Using example queries [Zhu 2014]

Few methods for textual data using examples
  - Snowball [Agichtein 2000]
  - DIPRE [Brin 1999]
Main idea: Create rules to extract wanted information from documents using examples

Output: Extraction rules

P: Percentage = 1.0 = 1.0
D: {5, 6} = 0.4
D: {percent, %} = 0.4 = 0.4
R: [0-9]+ = 0.2
D: {percent, %} = 0.4 = 0.3

Entity extraction by-example (SEER)

[Hanafi et al., 2017]
Learning rules

Example: 5 percent up

1. **Enumerate** possible primitives per example token

   - **5**
     - P: Number
     - percent
     - L: ‘percent’
     - R: [0-9]+

   - **5 percent**
     - P: Integer
     - L: ‘5’
     - R: [A-Za-z]+

   - **percent**
     - L: ‘percent’
     - R: [0-9]+

2. **Assign** scores to primitives

   - **Dubai**
     - T: 0-1
     - < Pre-built
       - < Dictionary
         - < Literal
           - < Token gap
             - Regex

   - **Dubai**
     - T: 0-1
     - < P: City
Learning rules (cont’d)

3. Generate rules

Example: 5 percent

Tokens:

Tree:

Rule:

Example: 6%

Tokens:

Tree:

Rule:

4. Merge

Intersection: [5 percent, 6%]
Web tables completion (InfoGather)

Main idea: Complete tables using partial information about tuples

[Yakout et al., 2012]

Incomplete table

<table>
<thead>
<tr>
<th>Model</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>S80</td>
<td></td>
</tr>
<tr>
<td>A10</td>
<td></td>
</tr>
<tr>
<td>GX-1S</td>
<td></td>
</tr>
<tr>
<td>T1460</td>
<td></td>
</tr>
</tbody>
</table>

Complete table

<table>
<thead>
<tr>
<th>Model</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>S80</td>
<td>Benq</td>
</tr>
<tr>
<td>A10</td>
<td>Innostream</td>
</tr>
<tr>
<td>GX-1S</td>
<td>Samsung</td>
</tr>
<tr>
<td>T1460</td>
<td>Benq</td>
</tr>
</tbody>
</table>

Web tables

<table>
<thead>
<tr>
<th>Model</th>
<th>Brand</th>
<th>Part No</th>
<th>Mfg</th>
</tr>
</thead>
<tbody>
<tr>
<td>S80</td>
<td></td>
<td>DSC W570</td>
<td>Sony</td>
</tr>
<tr>
<td>A10</td>
<td></td>
<td>T1460</td>
<td>Benq</td>
</tr>
<tr>
<td>GX-1S</td>
<td></td>
<td>Optio E60</td>
<td>Pentax</td>
</tr>
<tr>
<td>T1460</td>
<td></td>
<td>S8100</td>
<td>Nikon</td>
</tr>
</tbody>
</table>

InfoGather
Augmentation framework

Direct Match Approach (DMA)

- Traditional schema matching techniques using the attribute names and the values in the column

\[
S_{DMA}(T) = \begin{cases} 
\frac{|T \cap K \cap Q|}{\min(|Q|, |T|)} & \text{if } Q.A \approx T.B \\
0 & \text{otherwise}
\end{cases}
\]

[Yakout et al., 2012]
Ranking tables using PageRank

- PageRank

\[ \pi_u(v) = \epsilon \delta_u(v) + (1 - \epsilon) \sum_{\{w|(w,v)\in E\}} \pi_u(w) \alpha_{w,v} \]

- Personalized PageRank (PPR)

- Topic Sensitive Pagerank (TSP)

\[ \pi^\beta(v) = \epsilon \beta + (1 - \epsilon) \sum_{\{w|(w,v)\in E\}} \pi^\beta(w) \alpha_{w,v} \]
Serendipitous search

Main idea: Use related entities and query logs to find serendipitous searches

[Serendipitous search diagram]

Document

Francisco Pizarro
Rafting
Amazon

Serendipitous Search

Connected entities

America
Peru
Machu Picchu

Query Logs

rafting excursion down the urubamba river
el dorado temple of sun
indios quechuas
map of peru
sapa inca

Searches related to
Document content

[Extracts from related entities and query logs]

[Whole page text]

Serendipitous search

[Bordino et al., 2013]
Find queries using entity-query graph

Query-flow graph with entity nodes

Three types of arcs:

1. query to query:

\[ w_Q(q_i \rightarrow q_j) = w_{QFG}(q_i \rightarrow q_j) \]

2. entity to query

\[ w_{EQ}(e \rightarrow q) = \frac{f(q)}{\sum_{q_i \mid e \in X_{E}(q_i)} f(q_i)} \]

3. entity to entity

\[ w_{E}(e_u \rightarrow e_v) = 1 - \prod_{i=1,\ldots,r} \left( 1 - p_{q_{i_s} \rightarrow q_{i_t}}(e_u \rightarrow e_v) \right) \]

Idea: Run Personalized PageRank on entity-query graphs

[Bordino et al., 2013]
Search by multiple examples

Main idea: Document examples are used to find topics

[Zhu et al., 2014]

Action Movies
- Mission impossible
- Die Hard
- ...

Action Actors
- Bruce Willis
- Tom Cruise
- ...

Related topics and documents

Chuck Norris
Arnold Schwarzenegger

Search by examples
Nearest neighbor approach

Main Idea:
The similarity is an aggregation over the distances between document $D_i$ and its nearest query example.

[Zhu et al., 2014]
Where we are

- Relational databases
- Textual data
- Graphs and networks
- Challenges and Remarks

Challenges and Remarks
Graphs

Fact Graph

Arnold Schwarzenegger

actedIN

Terminator

Ontology Tree

is A

Person

subClassOf

Actor

<table>
<thead>
<tr>
<th>Release</th>
<th>1984</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget</td>
<td>$6.4M</td>
</tr>
<tr>
<td>Length</td>
<td>1h 48m</td>
</tr>
</tbody>
</table>
Graphs

RDF

(subject, predicate, object)

(Arnold_Schwarzenegger, is A, Person)
(Actor, subClassOf, Person)
(Arnold_Schwarzenegger, actedIn, Terminator)
Exemplar Queries

Input: $Q_e$, an example element of interest
Output: set of elements in the desired result set

Exemplar Query Evaluation

- **evaluate** $Q_e$ in a database D, finding a sample S
- **find** the set of elements $A$ similar to $S$ given a similarity relation

[Mottin et al., 2014]
Exemplar Queries

Input: $Q_e$, an example element of interest
Output: set of elements in the desired result set

Exemplar Query Evaluation

• evaluate $Q_e$ in a database $D$, finding a sample $S$
• find the set of elements $A$ similar to $S$ given a similarity relation
• [OPTIONAL] return only the subset $A^R$ that are relevant

[Mottin et al., 2014]
SIMILARITY

Nodes

Connectivity
- Mediator Nodes [Ruchansky'15]
- Clusters [Perozzi'14]

Properties
- Entity Search [Metzger'13, Sobczak'15]

Queries
- Path Queries [Bonifati'15]
- SPARQL [Arenas'16]

(Edge-)Labels
- Entity Tuples [Jayaram'15]
- Graph Structures [Mottin'14]

CHALLENGE: DISCOVER USER PREFERENCE
CHALLENGE: EFFICIENT SEARCH
The Minimum Wiener Connector Problem

Model: Unlabeled Undirected Graph
Query: A set of Nodes $Q$
Similarity: Shortest-Path distance
Output: A Set of Connector Nodes $H$

“explains” connections in $Q$

Connectors: Nodes with **HIGH closeness** to **ALL** the inputs

Similar to a Steiner-Tree but **overall pairwise distances** are optimized

Case: Infected Patients $\rightarrow$ Culprit/Other Infected
Case: Target Audience $\rightarrow$ Influencers

[Ruchansky, et al., 2015]
The Minimum Wiener Connector Problem

Model: Unlabeled Undirected Graph
Query: A set of Nodes Q
Similarity: Shortest-Path distance
Output: A Set of Connector Nodes H

Called: Wiener Index.

\[ \min \sum_{(u,v) \in H} d(u, v) \]

\( d(u, v) \) is the shortest-path distance

[ tradeoff between size and average distance ]

W=1+2+1 = 4
W=1+1+1 = 3

Sometimes The Best Solution is NOT A Tree

NP-Hard

[Ruchansky et al., 2015]
Approximate minimum Wiener Index Connector

**Choose** \( r \) & \( \lambda \in \left[1, \log_{1+\beta} |V| \right] \)

- All Pairwise Distances
  - **Distances from a root** \( r \)

- Measure distance in \( H \)
  - **Precomputed distance in** \( G \)

**Edge Weights**

\[
w(u, v) = \lambda + \frac{\max\{d_G(r, u), d_G(r, v)\} - \lambda}{\lambda}
\]
Focused Clustering and Outlier Detection

Model: Unlabeled Undirected Graph with Node Attributes

Query: A set of Nodes $Q$

Similarity: Attribute Values & Connectivity (to be inferred)

Output: Clusters of Nodes: Dense & Coherent + Cluster Outliers

Case: Target Users $\rightarrow$ Community with same interests

Case: Products $\rightarrow$ Co-purchased products with similar features

[Perozzi et al., 2014]
Focused Clustering and Outlier Detection

**TASK:** Infer “FOCUS”, important attributes attribute weights $\beta$

1. Set of similar pairs, PS (from Q)
2. Set of dissimilar pairs, PD (random sample)
3. Learn a distance metric between PS and PD
   
   (Distance Metric Learning, inverse Mahalanobis distance: Xing, et al 2002)
Focused Clustering and Outlier Detection

**TASK:** Extract Clusters on Focused Graph

attribute weights $\beta \rightarrow$ Edge Weight

1. Find Starting Set of Candidates
   1.a Drop low-weight edges
   1.b Extract **Strongly Connected Component** $C_1, C_2, \ldots$

2. Grow Clusters around Candidates
   2.a Compute conductance of $C$: $\phi^{(w)}(C, G)$
   2.b Select node to add to $C'$: best improvement to $\Delta\phi^{(w)}(C, C')$ (greedy)
   2.c Prune Underperforming nodes

3. Detect Outliers: High *unweighted* conductance

[Perozzi et al., 2014]
Nodes

Connectivity
- Mediator Nodes [Ruchansky’15]
  - Clusters [Perozzi’14]

Properties
- Entity Search [Metzger’13, Sobczak’15]

(Edge-)Labels
- Path Queries [Bonifati’15]
  - SPARQL [Arenas’16]

Structures

Queries
- Entity Tuples [Jayaram’15]

Graph Structures [Mottin’14]
iQBEES: Entity Search by Example

Model: Knowledge Graph
Query: A set of Entities $Q$
Similarity: shared semantic properties
Output: A Set of Similar Entities ranked

Case: Products $\rightarrow$ Find Similar Products
Case: Social Media $\rightarrow$ User recommendation

[Metzger et al., 2013, Sobczak et al., 2015]
Maximal Aspects

[Metzger et al., 2013, Sobczak et al., 2015]

- Adding any aspect: \( E(A) = \{\text{Arnold}\} \)
- Include Typical Types
- Use most specific type
- REPEATABLE
  Update Q

Prune generic aspects

- \(?x\) type BodyBuilder
- \(?x\) type AmericanActor

Rank Set of aspects

- \(?x\) type AmericanActor
- \(?x\) type GovernorCalifornia
- \(?x\) hasHeight 1.88m
- \(?x\) type Entity

- \(?x\) type AmericanActor
- \(?x\) actedIn TheExpendables
- \(?x\) type ActionActor

?(x type BodyBuilder
?), (x type AmericanActor
), (x type GovernorCalifornia
), (x hasHeight 1.88m
), (x type Entity
), (x type AmericanActor
), (x type ActionActor
), (x actedIn TheExpendables
)
SIMILARITY

Nodes

- Connectivity
  - Mediator Nodes [Ruchansky’15]
  - Clusters [Perozzi’14]

- Properties
  - Entity Search [Metzger’13, Sobczak’15]

Structures

- Queries
  - Path Queries [Bonifati’15]
  - SPARQL [Arenas’16]

- (Edge-)Labels
  - Entity Tuples [Jayaram’15]
  - Graph Structures [Mottin’14]
Learning Path Queries on Graphs

Model: Edge Labeled Graph

Query: 2 sets of Entities $Q^+$, $Q^-$

Positive, Negative

Similarity: common path query (RegExp) $(bus|tram)^* Cinema$

Output: A Set of Nodes Satisfying some paths($Q^+$) but NOT paths($Q^-$)

Case: Proteins $\rightarrow$ Similar interactions/co-expression

Case: Tasks Initiator $\rightarrow$ Similar Processes/Behaviours

[Bonifati et al., 2015]
Learnability of Path Queries

**Query:** \( Q^+ \) & \( Q^- \) (Positive & Negative examples)

**Consistency:** \( \forall v \in Q^+. \text{paths}_G(v) \not\subseteq \text{paths}_G(Q^-) \)

1. Selecting the Smallest Consistent Paths
   - Infinite Paths? Fix maximal length \( K \) but...
   - When to use Kleene star * ?
     \[
     C \mid (A \cdot B \cdot C) \rightarrow (A \cdot B)^* \cdot C
     \]

2. Generalize SCP
   - a. Construct Prefix-Tree Acceptor
   - b. Generalize into DFA with Merge

**Consistency Check:** PSPACE-complete

- Enumerate Paths Up to Fixed distance
- For paths of Length \( N \)
  \( K = 2 \times N + 1 \)

[Bonifati et al., 2015]
Reverse engineering SPARQL queries

Model: Knowledge Graph

Query: Set of ANSWERS*

Similarity: common AND/OPT/FILTER query

Output: A SPARQL QUERY/RESULT

Case: Open Data → Query Unknown Schema

Case: Novice User → Avoid SPARQL

<table>
<thead>
<tr>
<th></th>
<th>?e1</th>
<th>?e2</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Mexico</td>
<td>Spanish</td>
</tr>
<tr>
<td>M2</td>
<td>Haiti</td>
<td></td>
</tr>
<tr>
<td>M3</td>
<td>Jamaica</td>
<td>English</td>
</tr>
</tbody>
</table>
Reverse engineering SPARQL queries

<table>
<thead>
<tr>
<th>Query:</th>
<th>Set of Variable Mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>?X</td>
</tr>
<tr>
<td>M1</td>
<td>John</td>
</tr>
<tr>
<td>M2</td>
<td>Mary</td>
</tr>
<tr>
<td>M3</td>
<td>Lucy</td>
</tr>
</tbody>
</table>

Enumerate all possible SPARQL queries satisfied by the mappings

\[ \Sigma^p_2 \text{— complete} \]
\[ \text{coNP—complete} \]

Build tree-shaped SPARQL queries IMPLIED by the mappings

\[ (\textcolor{red}{?X, type, Person}) \]
\[ ?X \neq \text{me} \]

\[ (\textcolor{red}{?X, email, ?Y}) \]
\[ (\textcolor{red}{?X, addr, ?Z}) \]

[ Arenas et al., 2016 ]
Reverse engineering SPARQL queries

Query: Set of Variable Mappings $\Omega$

<table>
<thead>
<tr>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>a2</td>
<td>a3</td>
<td>a4</td>
</tr>
</tbody>
</table>

$\Omega = \{M1, M2, M3, M4\}$

$\{M2, M4\}$

$\{M2, M4\}$

$\{M3, M4\}$

$\{M4\}$

Greedy: keep just enough to cover all variables

[Arenas et al., 2016]
Nodes
Connectivity
- Mediator Nodes [Ruchansky’15]
- Clusters [Perozzi’14]

Properties
- Entity Search [Metzger’13, Sobczak’15]

Queries
- Path Queries [Bonifati’15]
- SPARQL [Arenas’16]

(Edge-)Labels
- Entity Tuples [Jayaram’15]
- Graph Structures [Mottin’14]

Structures

SIMILARITY
Exemplar Queries

Model: Knowledge Graph

Input: Example Structure

Similarity: Isomorphism/Simulation

Output: A set of Graphs

[Query: Knowledge Graph]

Knowledge Graph

Mottin et al., 2014

D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
Computing exemplar queries

[Pruning technique:]
- Compute the neighbor labels of each node
- Prune nodes not matching query nodes neighborhood labels
- Apply iteratively on the query nodes

\[ W_{n,a,i} = \{n_1 \mid l(n_1, n_2) = a \ \forall \ N_{i-1}(n) \} \]

NP-complete (subgraph isomorphism)

\[ O(|V|^4) \] (simulation)

\[ v \text{ neighborhood} = \{(B,1)\} \]
\[ \notin \]
\[ u \text{ neighborhood} = \{(A,1)\} \]

Labels at distance 1

[v, A]

[v, A]

[u, A]

[u, A]

[u, A]

[u, A]

No Match
Computing exemplar queries

Approximation:
- Nodes closed to the sample are more important
- Use Personalized PageRank with a weighted matrix
  \[ \mathbf{v} = (1 - c) \mathbf{A} \mathbf{v} + c \mathbf{p} \]
  
  - Weight edges: frequency of the edge-label
    \[ I(e_{ij}^\ell) = I(\ell) = \log \frac{1}{P(\ell)} = -\log P(\ell) \]
    \[ P(\ell) = \frac{|E^\ell|}{|E|} \]
Combination of two factors

1. Structural: similarity of two nodes in terms of neighbor relationships
2. Distance-based: the PageRank already computed
Graph query by example (GQBE)

Model: Knowledge Graph
Input: Entity Tuples
Similarity: Isomorphism
Output: A set of Tuples

In GQBE Input is a set of (disconnected) entity mention tuples

Q = (Google, S. Mateo)

Results = (Yahoo, S. Clara)
(CBS, New York)

[Jayaram et al., 2015]
GQBE: Maximum Query Graph

Q = (v₁, v₂)

1. Find the maximum query graph
   - Graph with M edges having the maximum weight

2. Answers subgraph-isomorphic to the query graph
   [NP-hard]

3. Return top-k

**Answer score:**
- Sum of query graph weights
- Similarity match between edges in the answer and the query (shared nodes take extra credit)

\[
\text{match}(e, e') = \begin{cases} 
    \frac{w(e)}{|E(u)|} & \text{if } u = f(u) \\
    \frac{w(e)}{|E(v)|} & \text{if } v = f(v) \\
    \frac{w(e)}{\min(|E(u)|, |E(v)|)} & \text{if } u = f(u), v = f(v) \\
    0 & \text{otherwise}
\end{cases}
\]
Multiple query tuples

Find answers using a lattice obtained removing edges from the union graph

GQBE finds answers for multiple query tuples
1. Compute a re-weighted union graph of the individual query graphs

[Jayaram et al., 2015]
SIMILARITY

Nodes

Connectivity
- Mediator Nodes [Ruchansky’15]
- Clusters [Perozzi’14]

Properties
- Entity Search [Metzger’13, Sobczak’15]

Queries
- Path Queries [Bonifati’15]
- SPARQL [Arenas’16]

Structures

Do not include User Feedback
Where we are

- Relational databases
- Textual data
- Graphs and networks
- Challenges and Remarks

Machine learning

D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
**Online exploration of datasets**

**Main idea:** Learn the items to show online as more points are acquired

Two ways of learning: passive and active
MindReader

Main idea: learn an implicit query from user examples and optional scores

Searching “mildly overweighted” patients

- The doctor selects examples by browsing patient database
- The examples have “oblique” correlation
- We can “guess” the implied query

Ishikawa et al., 1999

Height

Weight

✓ : good
✓ : very good

Searching “mildly overweighted” patients
Learning an ellipsoid distance

[Ishikawa et al., 1999]

\[ D(x, q) = (x - q)^T M (x - q) \]

Implicit query

\[ D(x, q) = \sum_{j}^{n} \sum_{k}^{n} m_{jk} (x_j - q_j) (x_k - q_k) \]

Learn the query minimizing the penalty = weighted sum of distances between query point and sample vectors

\[ \text{minimize} \sum_{i} (x_i - q)^T M (x_i - q) \]

subject to \( \text{det}(M) = 1 \)
Learning the distance

Query point is moved towards “good” examples — Rocchio formula in IR

\[ Q_0 : \text{query point} \]
\[ \cdot : \text{retrieved data} \]
\[ \checkmark : \text{relevance judgments} \]
\[ Q_1 : \text{new query point} \]

Learning can be done online!!!
Active learning for online query systems

Main idea: the system “query” the user to understand her preferences

Learn unknown preferences and minimize the number of questions to the user

[Vanchinathan et al., 2015]
Learning unknown preferences

**Problem:** Find a set $S$ that maximize the user preference within a budget (e.g., number of interactions)

\[
\arg \max \sum_{v \in S} \text{pref}(v)
\]

subject to $\text{Cost}(S) \leq \text{budget}$

User preferences

Cost for the set $S$
Background: Gaussian processes

[Bishop et al., 2006]

Idea: Model the user preferences as a Gaussian Process

A Gaussian Process (GP) is an infinite set of variables, any subset of this is Gaussian

$$P(f|\Sigma, \mu) = \frac{1}{\sqrt{2\pi}\Sigma^{1/2}} \exp\left(-\frac{1}{2} (f - \mu)^T \Sigma^{-1} (f - \mu)\right)$$

Gaussian prior

Specified only by mean and covariance

Given observations \( \{x, y\}_{i=1}^n \) over an unknown function \( f \) drawn from a Gaussian prior, the posterior is Gaussian

$$P(f|y) \propto \int dx \ P(f, x, y)$$


D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
GP-Select

Algorithm 1 GP-SELECT

Input: Ground Set $V$, kernel $\kappa$ and budget $B$
Initialize selection set $S$
for $t = 1, 2, \ldots, B$ do
    Model Update:
    $[\mu_{t-1}(:, \cdot), \sigma^2_{t-1}(:, \cdot)] \leftarrow \text{GP-Inference}(\kappa, (S, y_{1:t-1}))$
    Item Selection:
    Set $v_t \leftarrow \arg\max_{v \in V \setminus \{v_{1:t-1}\}} \mu_{t-1}(v) + \beta_{t}^{1/2} \sigma_{t-1}(v)$
    $S \leftarrow S \cup \{v_t\}$
    Receive feedback $y_t = f(v_t) + \epsilon_t$
end for

Vanchinathan et al., 2015

- Learn posterior
- Trade off exploration and exploitation
- Ask user feedback

- Exploration: select items with high-variance
- Exploitation: select items with high-value
Active learning on graphs – which prior? [Ma et al., 2015]

Idea: Use the graph structure to infer the node classes

Use graph Laplacian as prior
\[ L = D - A, \text{ A is the adjacency matrix} \]

\[ p(f) \sim \mathcal{N}(0, L^{-1}) \]

Laplacian: higher probability of having the same class if two nodes are connected
Explore-by-Example: AIDE

[Dimitriadou et al., 2015]
The AIDE algorithm

1. Divide the space into d-dimensional cubes
2. Find the sample points in the cubes (medoids)
3. Train the classifier
4. Refine the training sampling from neighbors of misclassified points
5. Boundary refinement

[Dimitriadou et al., 2015]
### Classification & Query Formulation

[Dimitriadou et al., 2015]

**Sample** | **Red** | **Green** | **Relevant**
--- | --- | --- | ---
Object A | 13.67 | 12.34 | Yes
Object B | 15.32 | 14.50 | No
.. | .. | .. | ..
Object X | 14.21 | 13.57 | Yes

**Decision Tree Classifier**

```sql
SELECT * FROM galaxy WHERE red<=14.82 AND red>=13.5 AND green<=13.74
```

### Sample Red Green Relevant

<table>
<thead>
<tr>
<th>Object</th>
<th>Red</th>
<th>Green</th>
<th>Relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object A</td>
<td>13.67</td>
<td>12.34</td>
<td>Yes</td>
</tr>
<tr>
<td>Object B</td>
<td>15.32</td>
<td>14.50</td>
<td>No</td>
</tr>
<tr>
<td>Object X</td>
<td>14.21</td>
<td>13.57</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Decision Tree**

SELECT * FROM galaxy WHERE red<=14.82 AND red>=13.5 AND green<=13.74
Misclassified Sample Exploitation

[Dimitriadou et al., 2015]

Sampling Areas

Red wavelength

Green Wavelength
Clustering-based Sampling

Red wavelength

Clusters-Sampling Areas

Green Wavelength

Idea: Use a k-medoid approach to find sampling areas

[Dimitriadou et al., 2015]
Where we are

- Relational databases
- Textual data
- Graphs and networks

Challenges and Remarks
Example-based methods

- Query suggestion using examples
- Reverse engineering queries
- Entity extraction by example text
- Web table completion using examples
- Search by example
- Community-based Node-retrieval
- Entity Search
- Path and SPARQL queries
- Graph structures as Examples
Example-based methods: takeaways

**Relational**
- Complex search space
- Exact and approximate
- Interactivity can improve the quality
- Limited to query inference

**Textual**
- Allows serendipitous search
- Easier document finding
- Speed up entity matching

**Graph**
- Exploit locality
- Entity attributes are expressive
- Reverse engineering: good approximations
- Large result-sets require ranking

---

D. Mottin, M. Lissandrini, T. Palpanas, Y. Velegrakis
The use of examples

Examples can ease data exploration

• … reduce need for complex queries / simplify user input
• … require no schema knowledge
• … allow uncertainty in search conditions
• … require little data analytics expertise
Where should we invest time

Machine learning

Approximate Methods

User models

Scalability
ADOPT HETEROGENEITY

Need for solutions that operate across different models operate on heterogeneous datastores
PERSONALIZATION

better understand user needs

Meta-data and User Profiles

exploit query log, prior searches, user context

“The Context of Mobile Interaction”

– Nadav Savio
DEMOCRATIZATION

easy access to data

tools that work on commodity hardware, mobile devices

data-exploration for everyday use-cases

D. Mottin, M. Lissandrini
INTERACTIVITY
gradually understand user need

ADAPTIVITY
build indexes and data structures on-the-go

D. Mottin, M. Lissandrini
NATURAL LANGUAGE INTERFACE

flexible, vague, imprecise input

exploration through conversation

D. Mottin, M. Lissandrini
Example is always more efficacious than precept

*Samuel Johnson*, *Rasselas* (1759), Chapter 29.

“New Trends on Exploratory Methods for Data Analytics.”
Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, Themis Palpanas.
*Proceedings of the Conference in Very Large Databases (PVLDB), 10(12), 2017*

**Slides:** [http://j.mp/DataExplore](http://j.mp/DataExplore)
Acknowledgments

We would like to thank the authors of the papers who kindly provided us the slides

Angela Bonifati, Radu Ciucianu, Marcelo Arenas, Gonzalo Diaz, Egor Kostylev, Yaacov Weiss, Sarah Cohen, Fotis Psallidas, Li Hao, Chan Chee Yong, Ilaria Bordino, Mohamed Yakout, Kris Ganjam, Kaushik Chakrabati, Thibault Sellam, Rohit Singh, Maeda Hanafi, Marcin Sydow, Mingzhu Zhu, Yoshiharu Ishikawa, Daniel Deutch, Nandish Jayaram, Bryan Perozzi, Kiriaki Dimitriadou, Yifei Ma, Natali Ruchansky, Quoc Trung Tran, Hastagiri Prakash Vanchinathan
References


A. Bonifati, R. Ciucanu, and A. Lemay. Learning path queries on graph databases. EDBT, 2015.


I. Bordino, G. De Francisci Morales, I. Weber, and F. Bonchi. From machu picchu to rafting the urubamba river: anticipating information needs via the entity-query graph. WSDM, 2013.

References

G. Diaz, M. Arenas, and M. Benedikt. **Sparqlbye: Querying rdf data by example.** PVLDB, 2016.


A. Gionis, M. Mathioudakis, and A. Ukkonen. **Bump hunting in the dark: Local discrepancy maximization on graphs.** ICDE, 2015.

M. F. Hanafi, A. Abouzied, L. Chiticariu, and Y. Li. **Synthesizing extraction rules from user examples with seer.** SIGMOD, 2017.


N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. **Querying knowledge graphs by example entity tuples.** TKDE, 2015.
References


Y. Ma, T.-K. Huang, and J. G. Schneider. **Active search and bandits on graphs using sigma-optimality.** UAI, 2015.


B. Perozzi, L. Akoglu, P. Iglesias Sa´nchez, and E. Müller. **Focused clustering and outlier detection in large attributed graphs.** KDD, 2014.
References


T. Sellam and M. Kersten. **Cluster-driven navigation of the query space.** TKDE, 2016.


References


M. M. Zloof. Query by example. AFIPS NCC, 1975.