Enhancing Usability and Explainability of Data Systems

Anna Fariha

Advisor: Alexandra Meliou
Democratization of data systems

Explainability

Usability

Trust
Makes data systems accessible to non-expert users.

- **Applications**
  - Data access
    - Querying relational databases
  - Data integration
  - Data transformation
  - Data visualization
  - Data summarization
    - Text document summarization
Enhances people’s confidence towards data systems.

• Applications
  • Artificial intelligence and machine learning
    • Model predictions
  • Novel interaction mechanisms
    • Programming by example
Explainability

Increases *transparency* of data systems.

- **Applications**
  - Machine learning
    - Model predictions
  - Distributed systems
    - Concurrent applications
  - Data evolution
    - Why/how two databases differ?
  - Fairness in algorithms/software
Dissertation outline

Query by Example (QBE)
[VLDB 2019]
[SIGMOD 2018] (demo)

Comparative User Study: QBE vs SQL
[CHI 2020]*

Data Summarization by Example
[VLDB 2020] (demo)

Adaptive Interventional Debugging
[SIGMOD 2020]

Data Change Explanation

Conformance Constraints: Trusted ML
[SIGMOD 2021]*

Explaining Tuple Non-conformance
[SIGMOD 2019] (demo)

* under submission/revision
Are data systems accessible to non-experts?

Who are our most valuable customers?
How to express complex task specifications?
Programming by example (PBE)

- A step towards democratization of computational power.
- Enhances usability for both non-experts and experts.

User provides examples

System “guesses” intent

Program synthesis

Result delivery
Querying relational databases by example

SQuID

Semantic similarity-aware Query Intent Discovery
• Alice wants to find all Funny Actors from the IMDb database.
Challenge 1: understanding the schema
Challenge 2: SQL expertise

SELECT person.name
FROM person, castinfo, movietogenre, genre
WHERE person.id = castinfo.person_id
  AND castinfo.movie_id = movietogenre.movie_id
  AND movietogenre.genre_id = genre.id
  AND genre.name = 'Comedy'
GROUP BY person.id
HAVING count(*) >= 40
Query by example (QBE)
Expectation vs reality

Eddie Murphy
Robin Williams
Jim Carrey

All actors
Humans use context

NOT SURE

NEED CONTEXT
Discovering semantic similarity

There is no “funny” attribute in the data
Discovering semantic similarity

Jim Carrey

Robin Williams

Eddie Murphy
SQuID

Semantic Similarity-aware Query Intent Discovery
SQuID Outline

Modeling Semantic Context
Query Intent Discovery
Real-time Performance
Evaluation
Semantic context: basic

- Directly affiliated with an entity.

- Person
  - Birth year: 1962
  - Gender: Male
  - Height: 6’ 2”
  - Age: 57
Semantic context: derived

- **Aggregate** over a basic property of an associated entity.
  - number of comedy movies an actor appeared in.
Filters

• Encode semantic context.

```
SELECT
  person
FROM
  people
WHERE
  color = orange
```
Intended or co-incidental?

- Male
- Born in North America
- Appeared in 80+ Hollywood movies
- Appeared in 40+ comedy movies
- Appeared in 20+ drama movies
- Height above 5 feet
- Born after 1940
- …

Eddie Murphy
Robin Williams
Jim Carrey
Abduction

• Most likely explanation of an observation.
• Most likely query given the examples.

Maximum likelihood estimation is abduction!
Problem definition

Query intent discovery: given a Database and Example, find Query such that:

\[ \text{Example} \subseteq \text{Query(Database)} \]

\[ \text{Query} = \arg \max_q P(q | \text{Example}) \]
Probabilistic abduction model

\[ P(\text{Query} | \text{Example}) = \frac{P(\text{Context} | \text{Query}) P(\text{Query})}{P(\text{Context})} \]
\[
\frac{P(\text{Context} | \text{Query})}{P(\text{Query})} = \frac{P(\text{Query})}{P(\text{Context})}
\]

**Domain selectivity**

- SELECT * FROM p WHERE 25 <= age <= 30
- SELECT * FROM p WHERE 5 <= age <= 90

**Association strength**

- Outlier

**Outlier**
\[
P(\text{Context} | \text{Query}) \frac{P(\text{Query})}{P(\text{Context})}
\]

- Data selectivity

<table>
<thead>
<tr>
<th>country</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>CAN</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>CAN</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
</tbody>
</table>

USA: 80%

CAN: 20%
\[ P(\text{Context}|\text{Query}) = \frac{P(\text{Query})}{P(\text{Context})} \]

\[ P(\text{USA}|\text{country} = \text{USA}) = 1 \]

<table>
<thead>
<tr>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>CAN</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>CAN</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>USA</td>
</tr>
</tbody>
</table>
\[ \frac{P(Context|Query)P(Query)}{P(Context)} \]

\[ P(USA| country = USA) = 1 \]

\[ P(USA | No Filter) = 0.8 \]

<table>
<thead>
<tr>
<th>country</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>CAN</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>CAN</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>USA</td>
</tr>
</tbody>
</table>

Table: country |     |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>USA</td>
</tr>
</tbody>
</table>

Context
Predicate
in Query
\[
P(\text{Context} | \text{Query}) \frac{P(\text{Query})}{P(\text{Context})}
\]

\[
P(\text{USA} | \text{country} = \text{USA}) = 1
\]

\[
P(\text{USA} | \text{No Filter}) = 0.8 \times 0.8
\]

\[
= 0.64
\]
\[ P(\text{Context}|\text{Query}) \frac{P(\text{Query})}{P(\text{Context})} \]

\[ P(\text{USA}| \text{country} = \text{USA}) = 1 \]

\[ P(\text{USA} | \text{No Filter}) = 0.8 \times 0.8 \times 0.8 = 0.51 \]
SQuID algorithm: to pick or drop filters?

\[ P(\text{Context}|\text{Filter})P(\text{Filter}) \]

With filter

Without filter
Real-time performance
Abduction-ready database

- **Offline Module**
  - Inverted indexing
  - Derived relation materialization
  - Filter selectivity precomputation

- **αDB abduction-ready database**
  - Derived relations
  - Semantic property statistics

- **Query Intent Discovery**
  - Entity disambiguation
  - Semantic context discovery
  - Query abduction

- Example tuples
- Result tuples
- SQL query
- Real-time
Evaluation

1. How efficient is SQuID for large datasets and many examples?
2. Does SQuID infer the right query?
3. Can alternative techniques be effective in intent discovery?
   • Query Reverse Engineering (TALOS, 2014)
   • Positive and Unlabeled Learning (Elkan et al., 2008)

• Query run-time comparison
• Case studies
Datasets

633 MB

15 relations
• person: 6M rows
• movies: 1M rows
• castinfo: 14M rows

16 benchmark queries

5 benchmark queries

20 benchmark queries
Experiment settings

Benchmark Query → Result → Sample → Inferred Query → Compare → SQuID Result

Ground Truth → Example → SQuID Result
How does SQuID perform with large datasets or many examples?

Abduction time is practical

Linear in example size

Abduction time is practical

Logarithmic in DB size

Abduction time (s)
SQuiD works with few examples

---

**Accuracy Metric**

- IQ1
- IQ2
- IQ3
- IQ4
- IQ5
- IQ6
- IQ7
- IQ8

**Precision**

- IQ9
- IQ10
- IQ11
- IQ12
- IQ13
- IQ14
- IQ15
- IQ16

**Recall**

**F-score**

---

**# Examples**

- IQ1
- IQ2
- IQ3
- IQ4
- IQ5
- IQ6
- IQ7
- IQ8

---

**Modeling Semantic Context**

---

**Query Intent Discovery**

---

**Real-time Performance**

---

**Evaluation**
Query reverse engineering (QRE)

Input

QRE TALOS

Reverse Engineered Query

Output

Exact match required
SQuID outperforms QRE

Benchmark Query with Cardinality

<table>
<thead>
<tr>
<th>Query</th>
<th>Actual</th>
<th>SQuID</th>
<th>TALOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ1</td>
<td>(113)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ2</td>
<td>(20)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ3</td>
<td>(1531)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ4</td>
<td>(1374)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ5</td>
<td>(12)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ6</td>
<td>(36)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ7</td>
<td>(35)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ8</td>
<td>(71)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ9</td>
<td>(23)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ10</td>
<td>(84)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ11</td>
<td>(291)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ12</td>
<td>(394)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ13</td>
<td>(57)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ14</td>
<td>(22)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ15</td>
<td>(2512)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ16</td>
<td>(207)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Modeling
Semantic Context

Query Intent Discovery
Real-time Performance
Evaluation

Log scale

Log scale
SQuID outperforms QRE

Animation movies produced by Pixar
Original Query

```
SELECT DISTINCT movie.title
FROM movie, production, company, movietogenre, genre
WHERE movie.id = production.movie_id AND
  production.company_id = company.id AND
  company.name LIKE '%Pixar%' AND
  movie.id = movietogenre.movie_id AND
  movietogenre.genre_id = genre.id AND
  genre.name = 'Animation';
```

SQuID outperforms QRE
### Original Query

```sql
SELECT DISTINCT movie.title
FROM movie, production, company, movietogenre, genre
WHERE movie.id = production.movie_id
AND production.company_id = company.id
AND company.name LIKE '%Pixar%'
AND movie.id = movietogenre.movie_id
AND movietogenre.genre_id = genre.id
AND genre.name = 'Animation';
```

### SQuID Query

```sql
SELECT DISTINCT movie.title
FROM movie, production, company, movietogenre, genre
WHERE movie.production_year >= 1984 AND movie.production_year <= 2021 AND movie.country = USA AND genre.name = 'Animation' AND company.name = 'Pixar' AND movie.id = movietogenre.movie_id AND genre.id = movietogenre.genre_id AND movie.id = movietoproduction.movie_id AND company.id = movietoproduction.company_id;
```
SQuID outperforms QRE

Original Query
SELECT DISTINCT movie.title FROM movie, production, company, movietogenre, genre WHERE movie.id = production.movie_id AND production.company_id = company.id AND company.name LIKE '%Pixar%' AND movie.id = movietogenre.movie_id AND movietogenre.genre_id = genre.id AND genre.name = 'Animation';

SQuID Query
SELECT DISTINCT movie.title FROM movie, production, company, movietogenre, genre WHERE movie.production_year >= 1984 AND movie.production_year <= 2021 AND movie.country = 'USA' AND genre.name = 'Animation' AND company.name = 'Pixar' AND movie.id = movietogenre.movie_id AND genre.id = movietogenre.genre_id AND movie.id = movietoproduction.movie_id AND company.id = movietoproduction.company_id;

TALOS Query
SELECT DISTINCT movie.title FROM movie, production, company, movietogenre, genre WHERE movie.production_year >= 1984 AND movie.production_year <= 2021 AND movie.country = 'USA' AND genre.name = 'Animation' AND company.name = 'Pixar' AND movie.id = movietogenre.movie_id AND genre.id = movietogenre.genre_id AND movie.id = movietoproduction.movie_id AND company.id = movietoproduction.company_id;

Query Reverse Engineering overfits
SQuID outperforms machine learning

Generic machine learning cannot model RDBMS specific assumptions

**PU learning does not scale**

**PU learning requires >= 70% data as example**

Modeling
Semantic Context
Query Intent Discovery
Real-time Performance
Evaluation
Comparative user studies: QBE vs SQL
SQuID increased user efficiency
Overall, SQuID generated more accurate results
SQuiD was easier to use
Participants were satisfied with SQuID results.
SQuID or SQL?

![Bar chart showing preferences between SQuID and SQL]

- Definitely SQuID
- Probably SQuID
- No preference
- Probably SQL
- Definitely SQL

# Users
Anecdotal comments

“Even if I forget about syntax . . . figuring out how to go about writing the pseudo-code query for funny actors [is difficult]”

“Vague tasks are generally a lot more open to interpretation. Coding up a query that meets someone’s vague specifications [is] hard . . . It was very hard to nail down what the correct definition of funny is.”
Personalized text document summarization

SuDocu: Summarizing Documents by Example
Personalized summarization

Document → Personalized summary
Summarization by example

Example Summaries

Automatic Summaries
In 1957, Utah created the Utah State Parks Commission with four parks. Today, Utah State Parks manages 43 parks and several undeveloped areas totaling over 95,000 acres of land and more than 1,000,000 acres of water. Utah's state parks are scattered throughout Utah, from Bear Lake State Park at the Utah/Idaho border to Edge of the Cedars State Park Museum deep in the Four Corners region and everywhere in between. Utah State Parks is also home to the state's off highway vehicle office, state boating office and the trails program.

The state of Utah relies heavily on income from tourists and travelers visiting the state's parks and ski resorts. Today, Utah State Parks manages 43 parks and several undeveloped areas totaling over 95,000 acres of land and more than 1,000,000 acres of water. With five national parks (Arches, Bryce Canyon, Canyonlands, Capitol Reef, and Zion), Utah has the third most national parks of any state after Alaska and California. Temperatures dropping below 0 °F (−18 °C) should be expected on occasion in most areas of the state most years.
Dissertation outline

Query by Example (QBE)
- [VLDB 2019]
- [SIGMOD 2018] (demo)

Comparative User Study: QBE vs SQL
- [CHI 2020]*

Data Summarization by Example
- [VLDB 2020] (demo)

Adaptive Interventional Debugging
- [SIGMOD 2020]

Data Change Explanation

Conformance Constraints: Trusted ML
- [SIGMOD 2021]*

Explaining Tuple Non-conformance
- [SIGMOD 2019] (demo)

* under submission/revision
Part 2:
Trust in Data Systems
To trust or not to trust?

IBM's Watson AI suggested 'often inaccurate' and 'unsafe' treatment recommendations for cancer patients, internal documents show.
To trust or not to trust?

Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam
Conformance constraints: trusted machine learning
## Trusting ML predictions

<table>
<thead>
<tr>
<th>Training data</th>
<th>New data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
</tr>
<tr>
<td></td>
<td>Green</td>
</tr>
<tr>
<td></td>
<td>Pink</td>
</tr>
<tr>
<td></td>
<td>Orange</td>
</tr>
</tbody>
</table>
Trusting ML predictions

Training data

New data

- Red
- Yellow
- Green

- Pink
- Orange

Non-conforming
Non-conformance = untrustworthy prediction

Detection

Is it non-conforming?
A real-world example: airlines dataset

Regression task: predict arrival delay

<table>
<thead>
<tr>
<th></th>
<th>dep_date</th>
<th>dep_time</th>
<th>arr_time</th>
<th>duration (minute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>May 2</td>
<td>14:30</td>
<td>18:20</td>
<td>230</td>
</tr>
<tr>
<td>...</td>
<td>July 22</td>
<td>09:05</td>
<td>12:15</td>
<td>195</td>
</tr>
<tr>
<td>...</td>
<td>June 6</td>
<td>10:20</td>
<td>20:00</td>
<td>582</td>
</tr>
<tr>
<td>...</td>
<td>May 19</td>
<td>11:10</td>
<td>13:05</td>
<td>117</td>
</tr>
<tr>
<td>...</td>
<td>April 7</td>
<td>22:30</td>
<td>06:10</td>
<td>458</td>
</tr>
</tbody>
</table>

DAYTIME flights

OVERNIGHT flight
A real-world example: airlines dataset

- Trained with DAYTIME flights only
- Constraints observed in DAYTIME flights
  - “departure time is earlier than arrival time”
  - “their difference is very close to flight duration”

- OVERNIGHT flights
  - violate DAYTIME flights’ constraints
  - incur high regression error

Constraint violation correlates with high regression error
Conformance constraints (CCs)

ML pipelines drop low-variance dimensions to achieve dimensionality reduction.

ML models assume that training data’s constraints/properties will continue to hold during serving.
Conformance constraints

- **constraints** that the data satisfies
- capture the **invariants** of the data
Conformance constraints

- Encode linear arithmetic relationship over multiple attributes.

\[-\epsilon \leq (60 \cdot \text{arr\_hour} + \text{arr\_min}) - (60 \cdot \text{dep\_hour} + \text{dep\_min}) - \text{duration} \leq \epsilon\]
Conformance constraints: example

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
</tbody>
</table>

-\[ 10 \leq \text{BMI} \leq 40 \]

-\[ -40 \leq (28 \times \text{Height} - \text{Weight}) \leq 30 \]
Violation of conformance constraint

\[ 10 \leq \text{BMI} \leq 40 \]

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
<tr>
<td>6 feet</td>
<td>170 lbs</td>
<td>231</td>
</tr>
</tbody>
</table>
### Degree of violation

10 ≤ BMI ≤ 40

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
<tr>
<td>6 feet</td>
<td>170 lbs</td>
<td>231</td>
</tr>
</tbody>
</table>

Note: The BMI value of 231 is outside the normal range.
Projection

\[-\epsilon \leq (60 \cdot \text{arr\_hour} + \text{arr\_min}) - (60 \cdot \text{dep\_hour} + \text{dep\_min}) - \text{duration} \leq \epsilon\]
What are “good” projections?

- Infinitely many projections possible
  - Pick the \textbf{low-variance} projections.
  - Because?
    - They more useful in detecting trends in the data.

- Do we pick all low-variance projections?
  - Pick a set of projections with \textbf{low pair-wise correlations}.
  - Because?
    - They complement each other.
Low-variance projections
Projections with small mutual correlation
Discovering projections: PCA

- Principal Component Analysis (PCA)
  - Produces projections with small mutual correlations
    - Intuition: principal components are orthogonal to each other
  - Computing violation
    - Weigh CCs with low variance projections more
    - Weigh CCs with high variance projections less
Disjunctive conformance constraints

- Divide the dataset into **disjoint** partitions.
- Learn CCs for each partition.
- Compute **disjunctive** CCs.

\[ \psi_2 : M = \text{“May”} \Rightarrow -2 \leq AT - DT - DUR \leq 0 \]

\[ \lor \quad M = \text{“June”} \Rightarrow 0 \leq AT - DT - DUR \leq 5 \]

\[ \lor \quad M = \text{“July”} \Rightarrow -5 \leq AT - DT - DUR \leq 0 \]
Complexity analysis

• Runtime
  • Linear in number of tuples in the dataset
  • Cubic in number of attributes
  • Highly parallelizable

• Memory
  • Quadratic in number of attributes
Experimental results: two applications

• Trusted Machine Learning
  • Is there a relationship between CC violation and the ML model’s prediction accuracy?

• Data-drift
  • Can CCs be used to quantify data drift?
Trusted machine learning: airlines dataset

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Serving</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Daytime</td>
<td>Overnight</td>
</tr>
<tr>
<td>Average violation</td>
<td>0.02%</td>
<td>0.02%</td>
</tr>
<tr>
<td>MAE</td>
<td>18.95</td>
<td>18.89</td>
</tr>
</tbody>
</table>
Data drift: EVL benchmark (1/4)

- CD-MKL
- CD-Area
- PCA-SPLL (25%)
- CCSynth

1CDT
Data drift: EVL benchmark (2/4)

- CD-MKL
- CD-Area
- PCA-SPLL (25%)
- CCSynth
Data drift: EVL benchmark (3/4)

- CD-MKL
- CD-Area
- PCA-SPLL (25%)
- CCSynth

MG-2C-2D

[Diagram showing scatter plots and line graphs]
Data drift: EVL benchmark (4/4)
Dissertation outline

Query by Example (QBE)
- [VLDB 2019]
- [SIGMOD 2018] (demo)

Comparative User Study: QBE vs SQL
- [CHI 2020]*

Data Summarization by Example
- [VLDB 2020] (demo)

Adaptive Interventional Debugging
- [SIGMOD 2020]

Data Change Explanation

Conformance Constraints: Trusted ML
- [SIGMOD 2021]*

Explaining Tuple Non-conformance
- [SIGMOD 2019] (demo)

* under submission/revision
Part 3: Explanation Frameworks
Why ML models fail for certain tuples?

How is this different?
Why do systems (sometimes) behave unexpectedly?

Why did the system crash?
ExTuNe
Explaining Tuple Non-conformance
Conformance constraints

Detection

ExTuNe

Explanation

Is it non-conforming?

Why is it non-conforming?
Tuple-level explanation
## Tuple-level explanation

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
<tr>
<td>6 feet</td>
<td>170 lbs</td>
<td>231</td>
</tr>
</tbody>
</table>
## Tuple-level explanation

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
<tr>
<td>6 feet</td>
<td>170 lbs</td>
<td>231</td>
</tr>
</tbody>
</table>
### Intervention reveals causality

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
<tr>
<td>6 feet</td>
<td>170 lbs</td>
<td>25.9</td>
</tr>
</tbody>
</table>

Mean = 25.9

change
## Intervention reveals causality

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
<tr>
<td>6 feet</td>
<td>170 lbs</td>
<td>25.9</td>
</tr>
</tbody>
</table>
Intervention reveals causality

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 feet</td>
<td>142 lbs</td>
<td>19.3</td>
</tr>
<tr>
<td>5 feet</td>
<td>170 lbs</td>
<td>33.2</td>
</tr>
<tr>
<td>5 feet</td>
<td>130 lbs</td>
<td>25.4</td>
</tr>
<tr>
<td>16 feet</td>
<td>70 lbs</td>
<td>25.9</td>
</tr>
</tbody>
</table>

Blame!
ExTuNe principles: actual causality

Actual Causality

“When $K$ other events are removed, then $C$ is a counterfactual cause of $E$”

- $C$ is an actual cause of $E$
- $C$’s responsibility is $1/(K + 1)$
ExTuNe interface

Upload reference data

Learn Data Invariants

Upload test data:

Show aggregated attribute-responsibility

<table>
<thead>
<tr>
<th>id</th>
<th>Violation</th>
<th>age</th>
<th>gender</th>
<th>height</th>
<th>weight</th>
<th>ap_hi</th>
<th>ap_lo</th>
<th>cholesterol</th>
<th>gluc</th>
<th>smoke</th>
<th>alco</th>
<th>active</th>
</tr>
</thead>
<tbody>
<tr>
<td>21321</td>
<td>0.27</td>
<td>22652</td>
<td>2</td>
<td>163</td>
<td>70</td>
<td>200</td>
<td>180</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18398</td>
<td>0.21</td>
<td>21770</td>
<td>1</td>
<td>161</td>
<td>84</td>
<td>196</td>
<td>182</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2318</td>
<td>0.2</td>
<td>18961</td>
<td>1</td>
<td>158</td>
<td>74</td>
<td>200</td>
<td>170</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8643</td>
<td>0.15</td>
<td>15086</td>
<td>2</td>
<td>190</td>
<td>165</td>
<td>160</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15682</td>
<td>0.15</td>
<td>21300</td>
<td>2</td>
<td>176</td>
<td>98</td>
<td>240</td>
<td>110</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>29677</td>
<td>0.15</td>
<td>16615</td>
<td>2</td>
<td>196</td>
<td>180</td>
<td>130</td>
<td>80</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14276</td>
<td>0.14</td>
<td>19798</td>
<td>2</td>
<td>174</td>
<td>75</td>
<td>240</td>
<td>120</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24399</td>
<td>0.14</td>
<td>15117</td>
<td>2</td>
<td>180</td>
<td>200</td>
<td>150</td>
<td>90</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>17058</td>
<td>0.13</td>
<td>21897</td>
<td>1</td>
<td>152</td>
<td>87</td>
<td>220</td>
<td>110</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10556</td>
<td>0.13</td>
<td>21721</td>
<td>2</td>
<td>173</td>
<td>116</td>
<td>190</td>
<td>80</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
ExTuNe evaluation: case studies
Anomaly in COVID dataset

Conformance constraint: \(#\text{positive} + \#\text{negative} = \#\text{total}\)

<table>
<thead>
<tr>
<th>Violation</th>
<th>date</th>
<th>state</th>
<th>positive</th>
<th>negative</th>
<th>pending</th>
<th>hospitalized</th>
<th>death</th>
<th>total</th>
<th>population</th>
<th>hospital_beds</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>0.350000</td>
<td>NY</td>
<td>10356</td>
<td>35081</td>
<td>0</td>
<td>1603</td>
<td>44</td>
<td>45437</td>
<td>19453561</td>
<td>52524</td>
</tr>
<tr>
<td>36</td>
<td>0.350000</td>
<td>NY</td>
<td>25665</td>
<td>65605</td>
<td>0</td>
<td>3234</td>
<td>210</td>
<td>91270</td>
<td>19453561</td>
<td>52524</td>
</tr>
<tr>
<td>256</td>
<td>0.290000</td>
<td>NY</td>
<td>7102</td>
<td>25325</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>32427</td>
<td>19453561</td>
<td>52524</td>
</tr>
<tr>
<td>4</td>
<td>0.240000</td>
<td>CA</td>
<td>2102</td>
<td>13452</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>27654</td>
<td>39512223</td>
<td>71122</td>
</tr>
<tr>
<td>59</td>
<td>0.240000</td>
<td>CA</td>
<td>1733</td>
<td>12567</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>26400</td>
<td>39512223</td>
<td>71122</td>
</tr>
<tr>
<td>311</td>
<td>0.220000</td>
<td>NY</td>
<td>4152</td>
<td>18132</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>22284</td>
<td>19453561</td>
<td>52524</td>
</tr>
<tr>
<td>91</td>
<td>0.200000</td>
<td>NY</td>
<td>20875</td>
<td>57414</td>
<td>0</td>
<td>2635</td>
<td>114</td>
<td>78289</td>
<td>19453561</td>
<td>52524</td>
</tr>
<tr>
<td>88</td>
<td>0.190000</td>
<td>NJ</td>
<td>2844</td>
<td>359</td>
<td>94</td>
<td>0</td>
<td>27</td>
<td>3297</td>
<td>8882190</td>
<td>21317</td>
</tr>
<tr>
<td>471</td>
<td>0.130000</td>
<td>NJ</td>
<td>178</td>
<td>120</td>
<td>20</td>
<td>0</td>
<td>2</td>
<td>218</td>
<td>8882190</td>
<td>21317</td>
</tr>
<tr>
<td>389</td>
<td>0.130000</td>
<td>CA</td>
<td>483</td>
<td>7981</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>8407</td>
<td>39512223</td>
<td>71122</td>
</tr>
<tr>
<td>51</td>
<td>0.130000</td>
<td>WA</td>
<td>2221</td>
<td>31712</td>
<td>0</td>
<td>0</td>
<td>110</td>
<td>33933</td>
<td>7614893</td>
<td>12945</td>
</tr>
</tbody>
</table>

178 + 120 = 298

178 + 120 ≠ 218
Explaining data systems’ failure

AID: Causality-guided Adaptive Interventional Debugging
DBMS are complex and contain bugs

- concurrent
- parallel
- asynchronous
Intermittent failure

sometimes **succeeds** sometimes **fails**

```json
{
    name = "SIGMOD",
    venue = "Portland",
    year = 2020
}
```

**Input**

**DBMS**

**Runtime conditions**
- Thread scheduling
- Timing
Motivation and goal

Can’t reproduce!

Help me debug!

Investigate root causes of intermittent failure
Npgsql intermittent failure
[ADO.NET data provider for PostgreSQL]

Race condition in PoolManager.TryGetValue #2485

thetrerman opened this issue on May 29, 2019 · 3 comments

Steps to reproduce
I've created a test that can reproduce the issue. All you have to do is fill in the values for the
connection string. The test is VolatileTest as seen here:
https://github.com/thetrerman/npgsql/pull/1/files

The issue
Could be related to: #2146

In our production code, we are running into issues when trying to create a new Postgres
connection (Specifically when we call: var connection = new NpgsqlConnection(ConnectionString);).

This can intermittently occur when we are trying to start our service on a server which can contain
large amounts of database resources.

Assignees
thetrerman

Labels
bug

Projects
None yet

Milestone
4.0.8

Linked pull requests
**Npgsql intermittent failure**

- **Thread 1**
  - `Find(key):`
  - `localPools = pools`

- **Thread 2**
  - `Add(key):`
Npgsql intermittent failure

- **pools** (shared):
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
- **index** (shared):
  |   |   |   |   |   | 5 |   |   |   |   |
- **localPools** (local):
  | 0 | 1 | 2 | 3 | 4 |

**Thread 1**

Find(key):
1. `localPools = pools`

**Thread 2**

Add(key):
2. `if pools_is_filled:`
   3. `pools = ResizeDouble(pools)`
   4. `last_slot ++`
   5. `pools[last_slot] = key`
Npgsql intermittent failure

<table>
<thead>
<tr>
<th>pools (shared)</th>
<th>index (shared)</th>
<th>localPools (local)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9</td>
<td>5</td>
<td>0 1 2 3 4 5</td>
</tr>
</tbody>
</table>

Thread 1

Find(key):

1. localPools = pools

3. for i in range(0,last_slot+1):
   if (localPools[i] == key):
     return i

return null

Thread 2

Add(key):

2. if pools_is_filled:
   pools = ResizeDouble(pools)

   last_slot ++
   pools[last_slot] = key
Npgsql intermittent failure

pools (shared)

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

index (shared)

5

localPools (local)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Thread 1

Find(key):

1. localPools = pools

3. for i in range(0, last_slot + 1):
   if (localPools[i] == key):
      return i

   return null

Thread 2

Add(key):

2. if pools_is_filled:
   pools = ResizeDouble(pools)

   last_slot ++
   pools[last_slot] = key

Array Index Out of Bound
Npgsql intermittent failure

### Add(key):

```python
if pools_is_filled:
    pools = ResizeDouble(pools)
last_slot ++
pools[last_slot] = key
```

### Find(key):

```python
localPools = pools
for i in range(0, last_slot+1):
    if (localPools[i] == key):
        return i
return null
```
localPools = pools

if pools_is_filled:
    pools = ResizeDouble(pools)
    last_slot ++
    pools[last_slot] = key

for i in range(0, last_slot+1):
    if (localPools[i] == key)
        return i

return null
Investigating Npgsql crash

Root cause

Add() temporally overlaps with Find()

Explanation

Find() attempts to access invalid array index

Find() throws ArrayIndexOutOfBoundsException exception

Failure
Limitations of statistical debugging

- Find() is running too slow
- Find() and Get() temporally overlaps
- Get() is running too fast

... crash
Our goals

Root-cause identification

Find() and Get() temporally overlaps

crash
Our goals

Root-cause identification

Explanation

Find() and Get() temporally overlaps

array access at invalid index

index-out-of-bound exception

crash
AID: Adaptive Interventional Debugging
AID

statistical debugging

causality

group testing

fault injection
Finding candidate predicates

- Step 1: Program instrumentation finds all predicates
- Step 2: Statistical debugging finds correlated predicates

**Always** appear in **failed** executions

```
foo()
bar()
```

**Never** appear in **successful** executions

```
foo()
bar()
```
Cause must **temporally precede** effect

**Temporal precedence graph**
Approximating causality

P1 may cause P8
Approximating causality

P4 cannot cause P9
C is a *counterfactual cause* of E

If C had not occurred
E would not have occurred
Intervention
AID

statistical debugging

causality

group testing

fault injection
Fault injection
AID

statistical debugging

causality

fault injection

group testing
Coronavirus Test Shortages Trigger a New Strategy: Group Screening

Pooling diagnostic samples, and using a little math, lets more people get tested with fewer assays.
Adaptive group testing
AID applies group intervention
AID pruning

Diagram showing nodes labeled P1, P2, P3, P11, P10, and F with arrows and scissors indicating removal of some nodes.
Six real-world bugs

Data race
Use-after-free
Timing-bug

Network
BuildAndTest
HealthTelemetry

Microsoft
Microsoft
Microsoft

Random number collision
Order violation
Race condition
Statistical debugging vs AID

AID produces no false positives

# Predicates

- PostgreSQL
- Kafka
- Azure Cosmos DB
- Network
- BuiltAndTest
- HealthTelemetry

<table>
<thead>
<tr>
<th></th>
<th>Statistical Debugging</th>
<th>AID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Kafka</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Azure Cosmos DB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Network</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BuiltAndTest</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HealthTelemetry</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Adaptive group testing vs AID

AID’s pruning reduces #Interventions

# Interventions

<table>
<thead>
<tr>
<th>Category</th>
<th>Adaptive GT</th>
<th>AID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Npgsql</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Kafka</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>Azure Cosmos DB</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td>Network</td>
<td>80</td>
<td>90</td>
</tr>
<tr>
<td>BuiltAndTest</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>HealthTelemetry</td>
<td>30</td>
<td>40</td>
</tr>
</tbody>
</table>
Theoretical analyses

CPD: Causal Path Discovery
GT: Group Testing
AID: Adaptive Interventional Debugging
TAGT: Traditional Adaptive Group Testing

<table>
<thead>
<tr>
<th></th>
<th>Search space</th>
<th>#Interventions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPD</td>
<td>(B(2^n - 1) + 1)^J</td>
<td>Lower bound: (\frac{JB^n}{JBn + DS_1} \log \left(\frac{JB^n}{D}\right))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Upper bound (AID/TAGT): (J \log B + D \log (Jn) - \frac{D(D-1)S_2}{2Jn})</td>
</tr>
<tr>
<td>GT</td>
<td>(2^{JB^n})</td>
<td>Lower bound: (\log \left(\frac{JB^n}{D}\right))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Upper bound: (D \log B + D \log (Jn) - \frac{D(D-1)}{2JB^n})</td>
</tr>
</tbody>
</table>

**Dissertation outline**

- **Query by Example (QBE)**
  - [VLDB 2019]
  - [SIGMOD 2018] (demo)

- **Comparative User Study: QBE vs SQL**
  - [CHI 2020]*

- **Data Summarization by Example**
  - [VLDB 2020] (demo)

- **Adaptive Interventional Debugging**
  - [SIGMOD 2020]

- **Data Change Explanation**

- **Conformance Constraints: Trusted ML**
  - [SIGMOD 2021]*

- **Explaining Tuple Non-conformance**
  - [SIGMOD 2019] (demo)

* under submission/revision
Part 4: Proposed Contributions & Tentative Timeline

Data Change Explanation
How did my data change over last couple years?

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Activity</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:50:23</td>
<td>0.5, 69, 11425</td>
<td>&quot;271504218477G&quot;</td>
<td>32, 11, 32, 2018-10-10</td>
</tr>
<tr>
<td>15:50:23</td>
<td>0.5, 69, 11428</td>
<td>&quot;271507491568G&quot;</td>
<td>32, 11, 32, 2018-10-10</td>
</tr>
<tr>
<td>15:50:23</td>
<td>0.5, 69, 11484</td>
<td>&quot;271508481857G&quot;</td>
<td>32, 11, 32, 2018-10-10</td>
</tr>
<tr>
<td>11:35:05</td>
<td>0.05, 134, 11447</td>
<td>&quot;874231098887G&quot;</td>
<td>53, 11, 32, 2018-10-11</td>
</tr>
<tr>
<td>11:35:05</td>
<td>0.05, 134, 11448</td>
<td>&quot;874231135374G&quot;</td>
<td>53, 11, 32, 2018-10-11</td>
</tr>
<tr>
<td>11:35:05</td>
<td>0.05, 134, 11479</td>
<td>&quot;874231461234G&quot;</td>
<td>53, 11, 32, 2018-10-11</td>
</tr>
<tr>
<td>13:43:24</td>
<td>0.05, 34, 11487</td>
<td>&quot;874231676529G&quot;</td>
<td>59, 11, 32, 2018-10-11</td>
</tr>
<tr>
<td>22:25:59</td>
<td>0.75, 73, 14</td>
<td>&quot;271508486757G&quot;</td>
<td>19, 11, 32, 2018-10-08</td>
</tr>
<tr>
<td>22:25:59</td>
<td>0.75, 73, 11512</td>
<td>&quot;874231926046G&quot;</td>
<td>19, 11, 32, 2018-10-08</td>
</tr>
</tbody>
</table>
Prior work

• Existing approaches mostly focus on *syntactic* changes.

• Fail to provide *consumable summary* of changes.
Our goal

• Provide a consumable summary of **semantic** changes that explains how two databases differ.

• Explains database **evolution**.

• Reveals **patterns** in data change.
Evaluating SuDocu

• Data collection

• Tuning SuDocu’s learning algorithm

• Evaluation
  • Against ground-truth summaries
  • Comparison with other baselines
  • User study
Current status

Query by Example (QBE)
[VLDB 2019]
[SIGMOD 2018] (demo)

Comparative User Study: QBE vs SQL
[CHI 2020]*

Data Summarization by Example
[VLDB 2020] (demo)

Adaptive Interventional Debugging
[SIGMOD 2020]

Data Change Explanation

Conformance Constraints: Trusted ML
[SIGMOD 2021]*

Explaining Tuple Non-conformance
[SIGMOD 2019] (demo)

* under submission/revision
Tentative timeline

- October 2020: proposal defense
- November – December 2020: evaluating SuDocu
- January 2020: submit to VLDB 2021
- January – June 2021: work on Data Change Explanation Framework
- July 2021: submit to SIGMOD 2022
- June – August 2021: work on dissertation
- August 2021: final defense
Other project affiliations

• Fair classifiers: experiment and evaluation

• Data profile debugger

• Data sampling by example
Acknowledgements

• Committee

• Mentors and collaborators
Acknowledgements

• Armand Asnani, UMass
• Lucy Cousins, UMass
• Nischal Dave, UMass
• Larkin Flodin, UMass
• Juliana Freire, NYU
• Sainyam Galhotra, UMass
• Maliha Tashfia Islam, UMass
• Eunice Jun, UW
• Beryl Larson, Wellesley College
• Genglin Liu, UMass
• Raoni Lourenço, NYU
• Raj Kumar Maity, UMass
• Kancha Masalia, UMass
• Sheshera Mysore, UMass
• Tony Ohmann, UMass
• Vincent Pun, UMass
• Sheikh Muhammad Sarwar, UMass
• Michael Satanovsky, Hopkins School
• Divesh Srivastava, AT&T
• Zoey Sun, Smith College
• Nishant Yadav, UMass