Query Explanations

HDI - Fall 2021

Overview

- Background + Problem Statement
- Scorpion
- DIFF

Feel free to interrupt with questions!

Motivation

Explaining trends in high-volume data is a fundamental challenge for data analysts.

- Example: Tracking user engagement in an app.
  - Product manager (PM) notices daily active users declined in the last week. **Why?**

Discuss in groups of 3-4:

- How would you explain why using only relational tools?
- If you had a specific tool to help you, what would you want it to tell you?

Motivation

Explaining trends in high-volume data is a fundamental challenge for data analysts.

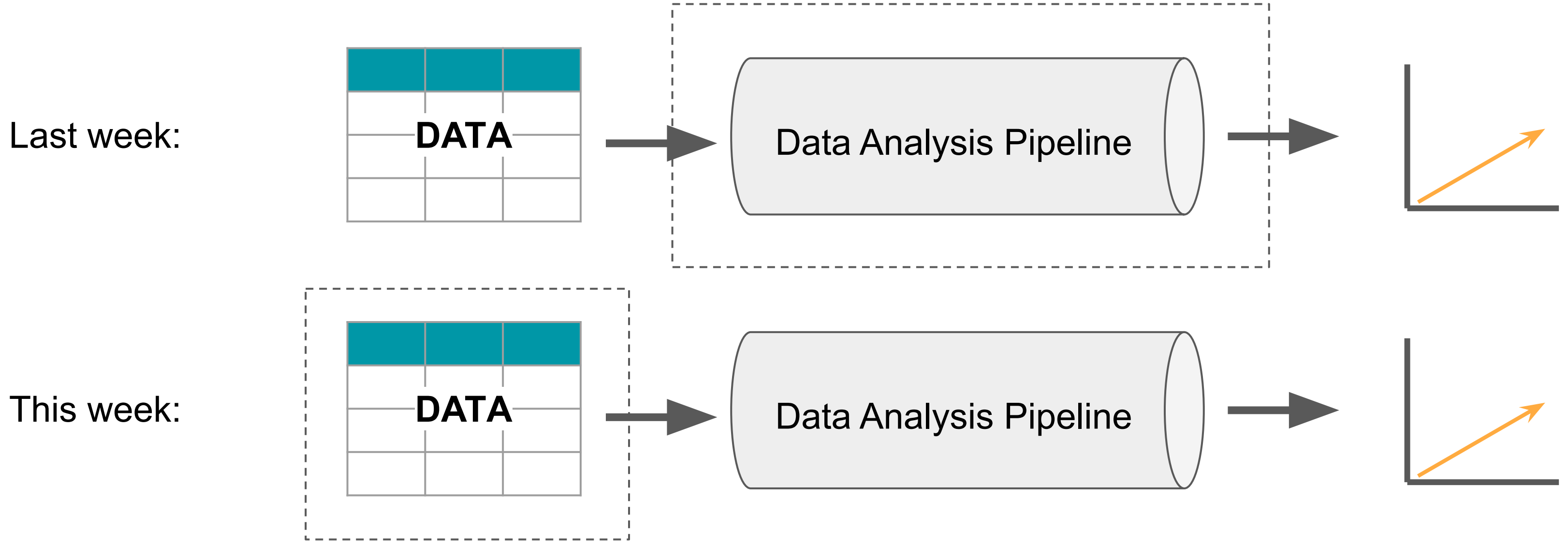
- Example: Tracking user engagement in an app.
  - Product manager (PM) notices daily active users declined in the last week. **Why?**

- Using traditional OLAP and BI tools, PM manually searches through different causes using GROUP BY and CUBE queries.
  - Must inspect many different potential causes: device location, temporal / seasonal factors, user demographics, etc.
  - All combinations of these causes too.

(slide inspired by DIFF VLDB talk)

Query Explanation vs. ML Interpretation

- **Explanation engines** are tools which try to automate this search process.



Scorpion

- An explanation engine which targets *outliers* — one or more aggregate values which behave differently than others.
- Finds a boolean predicate that when applied to the input data set (before the aggregation is computed), will cause outliers to look normal, while having minimal effect on the points that the user indicates are normal.
  - Conjunction of range clauses over continuous attributes and set containment clauses over the discrete attributes
  - SELECT * FROM table WHERE 5 <= c_attr <= 10 AND d_attr in (‘NY’, ‘NJ’)
- *Why predicates?*

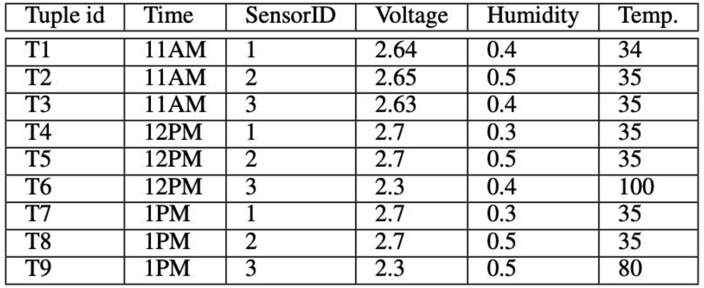
Problem Statement

- Environment:
  - Single relation D
  - Group by query Q
- User Inputs:
  - Result tuples labeled as outliers O
  - (Optional) Error direction vector V
  - Result tuples labeled as hold-out / normal H
- Goal: Find the predicate p that has the maximum “influence”

$$p^* = \arg \max_{p \in P_{A_{rest}}} inf(p)$$

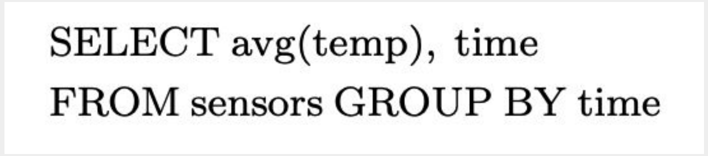
Problem Statement Example

Environment



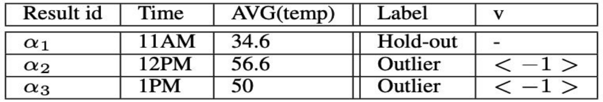
Tuple id	Time	SensorID	Voltage	Humidity	Temp.
T1	11AM	1	2.64	0.4	34
T2	11AM	2	2.65	0.5	35
T3	11AM	3	2.63	0.4	35
T4	12PM	1	2.7	0.3	35
T5	12PM	2	2.7	0.5	35
T6	12PM	3	2.3	0.4	100
T7	1PM	1	2.7	0.3	35
T8	1PM	2	2.7	0.5	35
T9	1PM	3	2.3	0.5	80

Table 1: Example tuples from sensors table



```
SELECT avg(temp), time
FROM sensors GROUP BY time
```

User Inputs



Result id	Time	AVG(temp)	Label	v
α_1	11AM	34.6	Hold-out	-
α_2	12PM	56.6	Outlier	$< -1 >$
α_3	1PM	50	Outlier	$< -1 >$

Table 2: Query results (left) and user annotations (right)

Predicate Influence

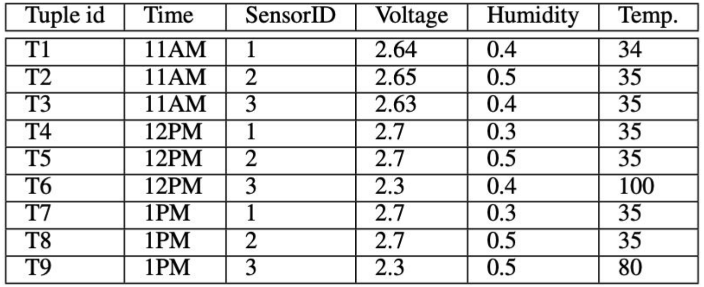
- Let g_o be the set of input tuples which were used to compute the outputs, *agg* be the aggregation function, and $|p(g_o)|$ be the number of tuples a predicate deletes.
- Basic influence:

$$\Delta_{agg}(o,p) = agg(g_o) - agg(g_o - p(g_o))$$

$$inf_{agg}(o,p) = \frac{\Delta o}{\Delta g_o} = \frac{\Delta_{agg}(o,p)}{|p(g_o)|}$$

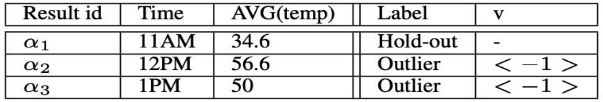
Predicate Influence

- Consider the predicate p = “SELECT * WHERE Tuple id = T6”
- Basic influence example:



Tuple id	Time	SensorID	Voltage	Humidity	Temp.
T1	11AM	1	2.64	0.4	34
T2	11AM	2	2.65	0.5	35
T3	11AM	3	2.63	0.4	35
T4	12PM	1	2.7	0.3	35
T5	12PM	2	2.7	0.5	35
T6	12PM	3	2.3	0.4	100
T7	1PM	1	2.7	0.3	35
T8	1PM	2	2.7	0.5	35
T9	1PM	3	2.3	0.5	80

Table 1: Example tuples from sensors table



Result id	Time	AVG(temp)	Label	v
α_1	11AM	34.6	Hold-out	-
α_2	12PM	56.6	Outlier	$< -1 >$
α_3	1PM	50	Outlier	$< -1 >$

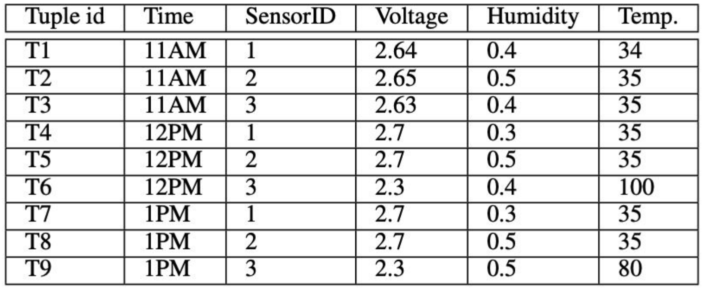
Table 2: Query results (left) and user annotations (right)

$$\Delta_{agg}(o,p) = agg(g_o) - agg(g_o - p(g_o))$$ = 56.6 - 35 = 21.6

$$inf_{agg}(o,p) = \frac{\Delta o}{\Delta g_o} = \frac{\Delta_{agg}(o,p)}{|p(g_o)|}$$ = 21.6 / 1 = 21.6

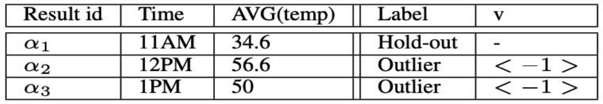
Predicate Influence

- Consider the predicate p = “SELECT * WHERE Tuple id IN (T5, T6)”
- Basic influence example:



Tuple id	Time	SensorID	Voltage	Humidity	Temp.
T1	11AM	1	2.64	0.4	34
T2	11AM	2	2.65	0.5	35
T3	11AM	3	2.63	0.4	35
T4	12PM	1	2.7	0.3	35
T5	12PM	2	2.7	0.5	35
T6	12PM	3	2.3	0.4	100
T7	1PM	1	2.7	0.3	35
T8	1PM	2	2.7	0.5	35
T9	1PM	3	2.3	0.5	80

Table 1: Example tuples from sensors table



Result id	Time	AVG(temp)	Label	v
α_1	11AM	34.6	Hold-out	-
α_2	12PM	56.6	Outlier	$< -1 >$
α_3	1PM	50	Outlier	$< -1 >$

Table 2: Query results (left) and user annotations (right)

$$\Delta_{agg}(o,p) = agg(g_o) - agg(g_o - p(g_o))$$ = 56.6 - 35 = 21.6

$$inf_{agg}(o,p) = \frac{\Delta o}{\Delta g_o} = \frac{\Delta_{agg}(o,p)}{|p(g_o)|}$$ = 21.6 / 2 = 10.8

Predicate Influence

- Full influence formula:

$$inf_{agg}(O, H, p, V) = \lambda \frac{1}{|O|} \sum_{o \in O} inf_{agg}(o, p, v_o) - (1-\lambda) \max_{h \in H} |inf_{agg}(h, p)|$$

- λ controls the amount we prioritize removing outliers vs. not affecting holdouts.
- We average over multiple outliers and take the maximum over holdouts.
- *Is this formula missing anything?*

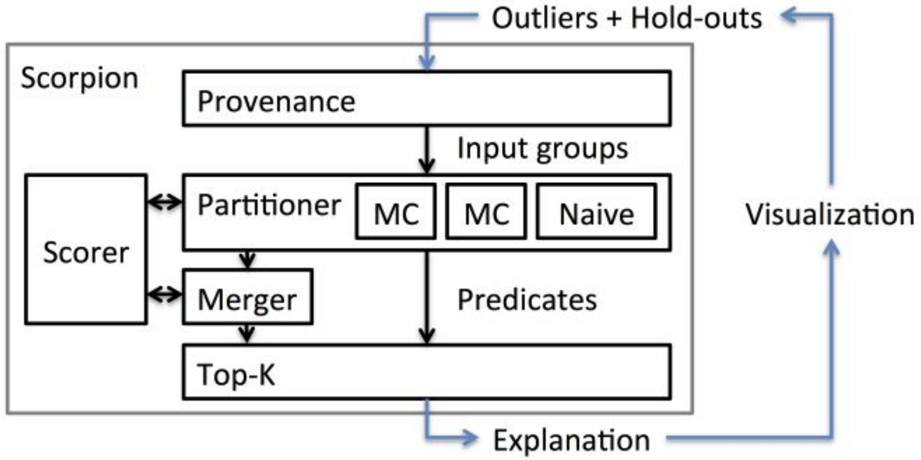
Finding the Most Influential Predicate

1. Compute the output provenance g_o.
   a. Already solved! (Smoke, ProvSQL, GProM, PERM)
2. Find the responsible subset.
   a. Potentially really slow. Naively, we need to try every possible subset.
3. Generate predicates.
   a. Must be done in conjunction with finding the responsible subset.

Scorpion’s contribution!

Scorpion Architecture

- Naive partitioner computes all predicates over single attributes, then all combinations of single attribute predicates.
- Continuous attributes are split into discrete equal sized intervals.
- Merger sorts the predicates from partitioner by score and greedily combines them.



Properties of Simple Aggregates

Certain aggregate functions have nice properties which allow us to improve over the naive algorithm.

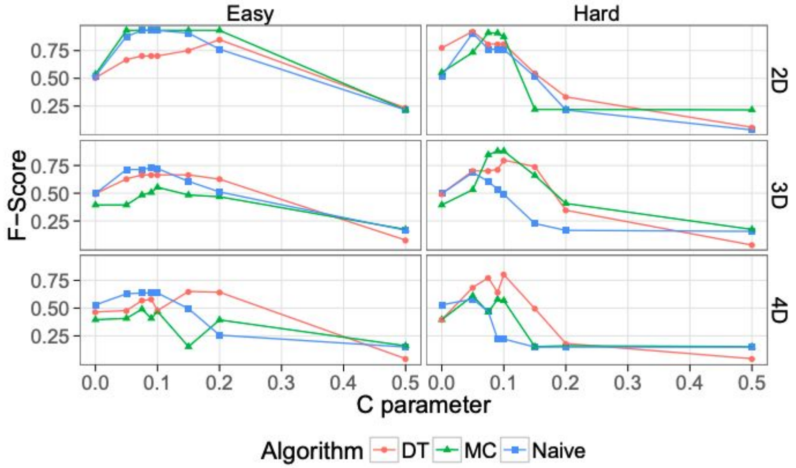
- Incrementally removable
  - The result of removing a subset, s, from the inputs, D, can be computed only by reading s. E.g. SUM(D - s) = SUM(D) - SUM(s)
- Independence
  - If a single tuple t1 is less influential than a tuple t2, adding t1 to a set of tuples will be less influential than adding t2 to the same set.
  - Adding a tuple more influential than the minimum influence can’t decrease a set’s influence.
- Anti-monotonic influence
  - The amount a predicate influences an aggregate result is greater than or equal to the influence of any predicate contained within.

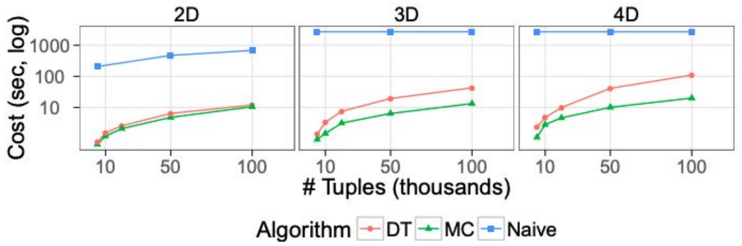
Other Partitioners + Optimizations

- Decision Tree (DT) Partitioner and DT Merger
    - Based on regression trees, the attribute space of an input group is recursively split to create a set of predicates. Requires *independence*.
- Bottom-Up (MC) Partitioner
    - Searches for influential single attribute predicates then intersects them to construct multi-attribute predicates. Requires *independence* and *anti-monotonicity*.
- Dimensionality Reduction
    - Removes non-informative attributes to reduce the search space.

Evaluation

- DT and MC compute similarly accurate results in applicable queries roughly 2 orders of magnitude faster than the naive method.





Evaluation

- For two real world datasets (INTEL and EXPENSE), Scorpion correctly identified sources of outlier readings in aggregate queries.
- *Are there any other experiments that may have been informative?*

Motivating DIFF

- Scorpion and other explanation engines are standalone products — data must be imported into these engines and analyzed separately rather than part of an existing OLAP workflow.
- Not designed with scalability in mind.

Solution: Create a new relational operator, DIFF, which combines the semantics of different explanation engines.

DIFF

Finds sets of attribute values {A1 = X, A2 = Y, ...} which increase the likelihood or risk of a certain point being an outlier.

Is the crash caused by {os = 8}?

RID	app_version	device_type	os	crash
1	v1	iPhone X	11	FALSE
2	v2	iPhone X	8	TRUE
3	v3	Galaxy S9	8	FALSE
4	v1	Galaxy S9	11	TRUE
5	v2	Galaxy S9	11	FALSE
6	v3	HTC ONE	8	FALSE

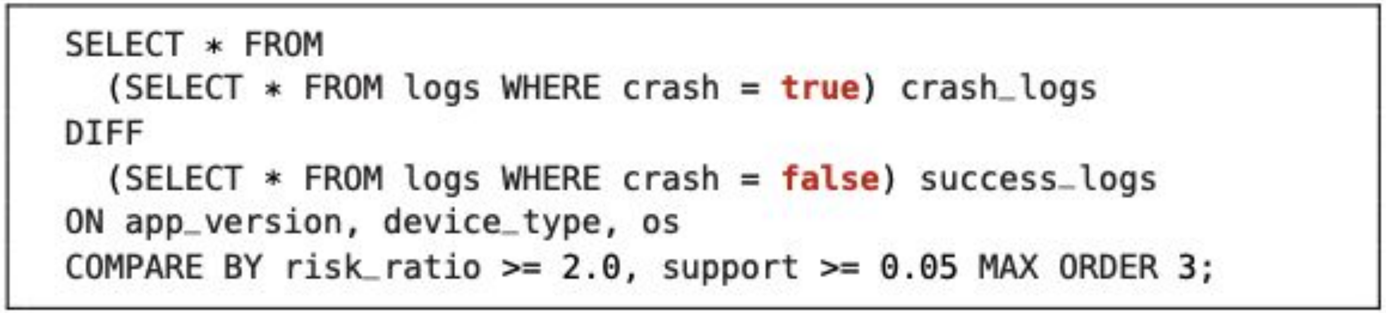
Example Workflow

In Scorpion:

1. SELECT COUNT(*), crash FROM logs GROUP BY crash;
2. Mark crash=False as “normal” and crash=True as “outlier”
3. Solve for predicates over input data.

Example Workflow

Using DIFF:



```
SELECT * FROM
  (SELECT * FROM logs WHERE crash = true) crash_logs
DIFF
  (SELECT * FROM logs WHERE crash = false) success_logs
ON app_version, device_type, os
COMPARE BY risk_ratio >= 2.0, support >= 0.05 MAX ORDER 3;
```

Metrics

$$\gamma_{\text{support}} \coloneqq \begin{cases} \mathcal{F} = \texttt{COUNT(*)} \\ h = \dfrac{\mathcal{F}^R_{\text{attrs}}}{\mathcal{F}^R_{\text{global}}} \end{cases}$$

$$\gamma_{\text{risk\_ratio}} \coloneqq \begin{cases} \mathcal{F} = \texttt{COUNT(*)} \\ h = \dfrac{\dfrac{\mathcal{F}^R_{\text{attrs}}}{\mathcal{F}^R_{\text{attrs}} + \mathcal{F}^S_{\text{attrs}}}}{\dfrac{\mathcal{F}^R_{\text{global}} - \mathcal{F}^R_{\text{attrs}}}{(\mathcal{F}^R_{\text{global}} - \mathcal{F}^R_{\text{attrs}}) + (\mathcal{F}^S_{\text{global}} - \mathcal{F}^S_{\text{attrs}})}} \end{cases}$$

$$\gamma_{\text{influence}} \coloneqq \begin{cases} \mathcal{F} = \{f,\ g\} \\ h = \lambda \dfrac{\text{remove}(f^R_{\text{global}}, f^R_{\text{attrs}})}{g^R_{\text{attrs}}} \\ \qquad -(1-\lambda)\dfrac{\text{remove}(f^S_{\text{global}}, f^S_{\text{attrs}})}{g^S_{\text{attrs}}} \end{cases}$$

The Power of Relational Operators

Logical Optimizations

- DIFF-JOIN Pushdowns
- Leveraging functional dependencies

Physical Optimizations

- Packed integers, bitsets, fast algorithms

Discussion

- Explanation engines are powerful tools for gaining insights into the *data* that is contributing to unexpected results.
- Scorpion returns boolean predicates with the highest influence.
- DIFF generalizes to user defined importance metrics and finds attribute values that fit those metrics.
- *What features discussed are these two explanation systems missing?*
- *How important is interactivity (running in interactive time)?*

Thanks!