Skyline Queries in Crowd-Enabled Databases

Christoph Lofi, Kinda El Maarry, Wolf-Tilo Balke

National Institute of Informatics, Tokyo
Institut für Informationssysteme, Braunschweig
Skyline queries are a popular and effective personalization technique

"Select all tuples which are not dominated by any other tuple."

Introduction

- Skyline queries are a popular and effective personalization technique

- "Select all tuples which are not dominated by any other tuple."
Introduction

• But: Skyline queries suffer when faced with incomplete data
  – Incomplete data is unfortunately very common
    • Results from federation, integration, unreliable information extraction, etc.
  – Skyline algorithms require complete knowledge!
Crowd-enabled databases promise to elicit missing data **during runtime**

- Integrate dynamic crowd-sourcing into skyline processing!

**Crowd-Sourcing:**

- Cognitive micro tasks issued to a large worker pool
- Usually, users are paid, e.g. Amazon Mechanical Turk
• But: Don’t crowd-source all missing data!
  – Too expensive and too slow, not needed for result
• Goal of this talk
  – Design a *cost-aware workflow* for crowd-enabled skyline queries on incomplete data
  – Illustrate the *design space* of such a work flow
  – Demonstrate a *heuristic* for assessing the *risk* associated with each tuple
• **Basic workflow**
  – Automatically select a suitable **value imputation technique**
  – **Impute** all missing values
  – Assess **risk** for each tuple
  – **Crowd-source** those tuple with the highest risk
  – Compute **skyline** using crowd-sourced and imputed values
Value Imputation

• Predict missing values using statistics and values of other tuples
  – Several algorithms available
  – We considered
    • Median imputation
    • KNN imputation (k-nearest neighbor)

• Select best algorithm by **sampling**
  – From complete dataset, randomly remove values (mimicking full incomplete dataset)
  – Measures per attribute:
    • Mean Squared Error, Error Standard Deviation
  – Select algorithm with lowest error
How can prediction errors impact the skyline?

- **False Positives:**
  - Tuple is considered to be in the skyline, although it doesn’t belong there

- **False Negatives:**
  - A tuple is excluded from the skyline, but it actually should be in

- **True Positives and True Negatives:**
  - Tuple treated correctly despite missing information
• Error Measure based on informedness

\[
\text{error}(\text{skyline}^{\text{imputed}}, \text{skyline}^{\text{real}}) = 2
\]

\[
\frac{\text{truePositives}(..)}{\text{truePositives}(..) + \text{falseNegatives}(..)}
\]

\[
\frac{\text{trueNegatives}(..)}{\text{trueNegatives}(..) + \text{falsePositives}(..)}
\]
Skyline Errors

- \(\times\) complete tuples
- \(\times\) skyline tuple (excluding predicted values)
- \(\rightarrow\) skyline frontier
- \(\bigcirc\) tuples with predicted value
- \(\bigcirc\) skyline tuples with predicted value
- \(\rightarrow\) error lines

![Skyline Errors Diagram]

- \(s_1, s_2, s_3, s_4, s_5, s_6\)
- \(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\)
• Task: **Rank predicted tuples** wrt. their potential for introducing errors
  – Crowd-Source most risky tuples!

• Base Assumption: **Normal error behavior**
  – i.e. all predicted values are within standard error bounds, no severe outliers
    • Attribute statistics already elicited during algorithm selection
  – Observe virtual tuples:
    • $t^P$: predicted tuple
    • $t^+$: optimistic interpretation
    • $t^-$: pessimistic interpretation
False Positives

• For each tuple consider impact on skyline of complete subset
  – Find set of false positives $fp(t^p)$

• Four cases for false positives
False Positives

a) False positives

b)
Skyline Queries in Crowd-Enabled Databases
False Negatives

• Also, four cases for false negatives
  – Find set of false negatives $f_n(t^p)$

• Again, four cases
False Negatives

Skyline Queries in Crowd-Enabled Databases
False Negatives

Skyline Queries in Crowd-Enabled Databases
• **Rank** all tuples and create **HITs**
  – Size of set of false negatives / positives with different weighting factor $\alpha$
  – Crowd-source most risky $k$ tuples
  – Group multiple tuples in one HIT
    • Optionally: Re-rank after each HIT

• How to obtain good values for HIT size, $\alpha$, and $k$?
  – User provided values directly
  – Rely on simulation using test sample
• **NBA Basketball Dataset**
  – 21,961 tuples, 6 attributes, skyline size 75 tuples

• **Notebooks**
  – 1,597 tuples, 8 attributes, skyline size 35 tuples

• **Cars**
  – 7,755 tuples, 6 attributes, skyline size 268 tuples

• Default assumption: 20% missing values
How well do imputation algorithms perform?

- KNN-imputation always better

**Cars: \( e^{-2} : 0.077 \)**
- Attributes have mean error between 0.2 and 0.25
- **Skyline error: 0.62**

**Notebooks: \( e^{-2} : 0.093 \)**
- Mean attribute error between 0.15 and 0.35
- **Skyline error: 0.28**

**NBA: \( e^{-2} : 0.0510 \)**
- \( games\_played \) at mean error 0.35
- All other attributes have mean error \(~0.05\)
- **Skyline error: 0.07**
Error – KNN Imputation

20% missing values
Missing Values and Costs

• Cars Dataset

- Skyline Queries in Crowd-Enabled Databases

  - Missing Values and Costs
    - KNN imputation, 28% Gold questions
Summary & Outlook

• **Hybrid crowd-sourcing** approach
  – Rely on **heuristics** / prediction / imputation in most cases
  – Identify and crowd-source those tuples with the **highest risk** to the final skyline result
  – Approach fully **self-tuning** with respect to data

• Result quality improves very quickly, only few crowd-sourcing HITs necessary