**Introduction**

The aim of this work is to address the exponential increase in data and computational times by Approximate Query Processing (AQP). However, instead of a sampling based approach (S-AQP) [2] we use a Query-Driven Learning (QDL) [1] approach. We train Machine Learning (ML) models that are able to estimate the answers of future queries using historical workloads.

**Contributions:**
1. Offer a light-weight, complimentary aggregate estimation engine that can be stored locally.
2. Agnostic to data backend. Can be used alongside relational databases, S-AQP etc.
3. Highly accurate and efficient estimations

**Query-Driven Methodology**

QDL uses past and incoming queries to learn query patterns and be able to build ML models that can estimate the results of new queries.

**Query Representation:**
- Each Aggregate Query (AQ) is represented as a vector by extracting its filtering parameters.
- Any aggregate (COUNT/AVG/SUM, etc) is supported...
- Each query is represented as \( q = (m, y) \)

**Partitioning (Clustering):**
- The set of queries, \( C = \{(m, y)\}_{i=1}^n \), is partitioned for better result estimation.
- \( C = \{C_1, \ldots, C_j\} \) where \( C_i \cap C_j = \emptyset, i \neq j \)
- Each subset has a representative \( W = \{w_1, \ldots, w_j\} \)

**ML Model Association**
- Every subset is used to train a supervised regression model.
- A set of ML models is created \( M = \{f_1, \ldots, f_k\} \) which are associated with the representatives \( W \)

**Answer Prediction**
- A prediction is made based on an ensemble scheme incorporating the predictions of the closest representative.

\[
\hat{y} = \sum_{k=1}^{K} \mathbb{I}_k(f_k(m))
\]
- Where \( \mathbb{I}_k \) is an indicator function evaluating to true if \( \hat{w}_k = \arg \min_{w_k} \|m - w_k\| \)

**System Architecture**

![Figure 1. Complete System Architecture – Showing how models are trained using parsed queries and how predictions are served through models.](image1)

**Evaluation Results**

- **Datasets Used:** Crimes and TPC-H (1GB)
- **Partitioner:** K-Means - Model: XGBoost
- **Experiments ran single threaded on Linux Ubuntu 16.04 using an i7 CPU at 2.2GHz with 6GB RAM**

**Relative Error**

- For crimes: Over 4000x faster than SAQP 0.01
- For TPC-H: 39000x

**Bibliography**


**Contact**

Fotis Savva
School of Computing Science, University of Glasgow
Email: f.savva.1@research.gla.ac.uk