Table Augmentation in Data Lakes
Federico Dassereto, Giovanna Guerrini

Relational-like operations at large scale

Motivation and Context

- **Data Lakes** are large repositories of both structured and unstructured data, among which tables without any schema information [1].
- Tables are extremely valuable due to their origin, since enterprises and administrative offices publish them daily.
- The lack of a common schema inside Data Lakes makes it difficult to efficiently perform traditional data management operations, such as joins over different tables.
- Existing approaches only focus on retrieving tables whose columns are the best option for joinability, without considering the amount of information that can be added but actually materializing the join [2].

We focus on a discovery scenario, recognizing as relevant joinability, unionability and augmentation operations.

The Problem

Given a data lake \( L = \{C_1, \ldots, C_k\} \), a query table \( Q \), a join column \( J \), a target column \( Q.T \), the goal is to provide a ranking of tables \( C_i \in L \) such that:

1. \( \exists j \in C_i \) s.t. \( Q.J \bowtie A_j \)
2. \( 3.A_i \in C_i \) s.t. \( corr(A_i, Q.T) \)

Where \( \bowtie \) denotes a fuzzy join over sets and \( corr(\cdot, \cdot) \) denotes a positive correlation among two columns and represents the augmentation of information. The output is a ranking \( R = \{C_1, \ldots, C_k\} \) in which tables are ordered by augmentation of information obtained by probing an index structure.

Key Concepts

| Table-as-a-Query | Paradigm for data lake exploration, in which the query is a complete table rather than a string representation |
| Data Lake        | Large repository of tables sharing no schema information |
| Joinability Search | Searching of tables that can be horizontally concatenated; the two (or more) tables must share a common column |
| Augmentation     | A more specific joinability search where the most relevant columns are not the most overlapping but the ones augmenting the information the most concerning Machine Learning tasks such as Classification on Regression |

Design Choices

The ideal solution to augment would be checking every possible pair of columns.

Limitations:

- This approach is inefficient due to the size of data lakes.
- Computing every possible pairs is unfeasible, so we search for approximate answers, by probing an indexing structure based on hashing functions.

Solutions:

- The index store information for each column with a single hash signature.
- The index allows for a fast retrieval of candidate tables joining with the join column of the query table.
- Probing the index results in a ranking of tables order by augmentation of information, where augmentation is quantified in terms of information theory.
- We allow for one-to-one joins over columns acting as key in their tables.

Augmentation

\[ H(Q.T | C_i) < H(Q.T | C_j) \]

Where \( C_i \) augments the information the most, since \( C_i.A_i \) helps predict whether the cities in the join column \( Q.J \) are in Europe or not. Augmentation is here expressed in terms of conditional entropy.

Forthcoming Research

- Implementation of ad-hoc hashing functions to preserve and capture similarities in columns, depending on their types.
- A priori pruning on joinability search and augmentation evaluation to speed up the execution time.
- Extension to different kind of joins, one-to-many and many-to-many.

References