Alternate *ACM* SIG Proceedings Paper in LaTeX Format

ABSTRACT

Abstract

1. INTRODUCTION

Desiderata of PQL

- Flexibility: PQL is tolerant of ambiguity by design, so that partial queries can be used to explore underdeveloped hypothesis.
- Expressivity: PQL must support a large class of adhoc queries, so that analysts are not constrained when exploring the types of questions they might want to ask.
- Ability to suggest possible actions: The suggestion can come in two forms: 1) Given a partial query, suggest the possible queries that are *simmilar in functionality* or 2) Given a data example or resultset, suggest possible next steps or actions that reflect *simmilar data relationships*.
- Ability to generate explainations:

The first two are language features of PQL and the latter two requirements are related to the PQL engine.

2. MOTIVATION

Popular theories in data sensemaking have suggested two different symbiotic modes of operations for understanding data. Pirolli and card describbes On the other hand, the Data/Frame theory suggests that —-. Our key insight is identifying that these two —- , represent two fundamentally —- : data or bottom up approaches are initiated through singleton examples, whereas framing comes from working with distribution or relationships.

3. USAGE SCENARIO

A team of analyst is a given a dataset that consist of the reported income of all the villager living in their county. They wanted to study whether there is any evidence of institutional bias worker salaries. Given this large dataset that contain hundreds of attributes containing personal information about each individual, they are not sure where to start. The only thing they know is that their measure attribute of interest the Income column, so they specify this in PQL. In this inital step, PQL tries to provide as much support information as it can by displaying a sample table of data records arranged in the order of increasing income, the attributes are displayed in the order of most to least importance, where importance is defined by how much the feature have an impact on the measure variable of interest. By convention, the primary key is always in the first column.

Given this information, analyst A was still not able to infer anything about any particular column to ask a new question, but he notice that Charlie has an incredible low income compared to the average income. Using PQL, he can searches for other records simmilar to Charlie. He notices that all of the returned records are all young children who work part-time jobs.

Analyst B skims through the records and notices that despite Mary having more years of working experience than Fred, she is getting paid $5000 less than him. He wants to find out whether employers are marginalizing the pay of certain individuals based on gender for the same job done. Given this hypothesis, he specifies a PQL query to find women who are in Sales and have simmilar years of experience as Mary and Fred. The PQL engine generates two support queries on this selected population to display the overview statistics and histograms of the selected subpopulation and its counterpart (i.e. men who are in Sales with simmilar years of experience).

4. PQL DESIGN

Given the prior work —- and usage scenario, the PQL design tries to support both example-driven and idea-driven queries to offer suggestions and explanations to the users.

"Idea-driven" partial queries (IDPQ): IDPQs is the conventional approach to querying a database where a user starts with a pre-existing idea of what he is looking for. The way that analyst B approached the problem is an example of a IDPQ. However, formulating SQL queries that maps user's high-level intentions (as highlighted by the second example) to specific query statements is challenging.

Recent work in natural language querying have tried to address this by parsing adjectives and quantifiers and asking

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

the users for additional information to resolve the ambiguity through form-based interface if needed. However, these system are often limited in their expressiveness as the linguistic and conceptual coverage of how users would express complex queries is often unbounded.

Instead of approaching this problem from the perspective of ambiguity resolution, we recognize that sometimes users themselves may not have a clear cut specification of what they are asking for. As a result, we introduce the EXPLORE operator and the special value ? to increase the tolerance of ambiguity inherently in the PQL language design, then the engine takes this partial specification decide what would be the best answers or suggestions to return as an output.

"Example-driven" partial queries (EDPQ): EDPQs are used when the users have no pre-existing idea of what he wants to do with the data. EDPQs enable users to select the information that they have already seen and use that in their query. Given a EDPQ, the system infers and suggest potential explanations or queries based on the data the user provides as a query.

Existing work in the area of Query-by-Examples (QBE) tries to do this by asking users to provide I/O examples of the query to be synthesized. However, if the user does not know what they are querying for, then they would not be able to come up with such an example.

A crucial difference that separates our work and QBE is that the goal of EDPQ is to generate potential explanations or query recommendations rather than generating a query that will address a predefined question. As a result, EDPQ has a more relaxed set of constraints compared to typical scenario in the query synthesis problem, which leads to a larger search space potential queries that offer further opportunities for optimization.

In addition, EDPQ accepts a more easy-to-come-up-with input modalities such as visualization distributions or an existing singleton record as examples to query. These example inputs to queries comes from the results of support queries. Support queries are queries that are non-essential in answering the queries, but provide auxcillary information that could help users jumpstart in their hypothesis generation. Examples of support queries includes requests to overview representatives, outliers, and other statistics, or generate different types of visualizations. These support queries can be either system generated (as part of suggestions and explainations) or defined by the user.

PQL engine generates *support queries*.

the ability to work with individual record-level examples and compare