Incremental Schema Recovery

Poonam Kumari
University at Buffalo, SUNY
poonamku@buffalo.edu

Gourab Mitra
University at Buffalo, SUNY
gourabmi@buffalo.edu

Oliver Kennedy
University at Buffalo, SUNY
okennedy@buffalo.edu

ABSTRACT

Abstract goes here

KEYWORDS

Stuff

ACM Reference Format:
Poonam Kumari, Gourab Mitra, and Oliver Kennedy. 1997. Incremental Schema Recovery. In *Proceedings of ACM Woodstock conference (WOODSTOCK'97).* ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.475/123_4

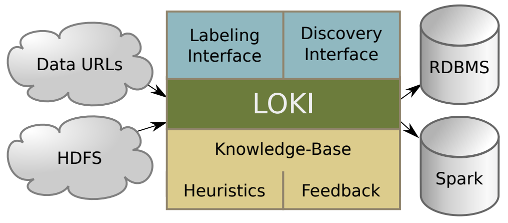

Figure 1: System Overview

1 INTRODUCTION

It's a story as old as time: A student gathers data, makes a graph with the data, writes a paper about the data. Then the student graduates and the data languishes, without so much as a wiki page or README file documenting it. The next student to use the data needs to spend hours, days, or even weeks reverse-engineering it. Then they also graduate and the whole process can start over again.

As a way to break this tragic cycle of data abandonment, we propose *Label Once, and Keep It* (**LOKI**), a data-ingest middleware for incremental, re-usable schema recovery. **LOKI** allows users to assemble schemas on-demand, both (re-)discovering and incrementally refining schema definitions in response to changing data needs. To accomplish this, **LOKI** is built around a knowledge-base of both approximate, as well as exact schema labelings. First, approximate labelings derived from existing open-data sets, user-feedback, and expert-provided heuristics, jump-start the labeling process. When a user first points **LOKI** at a new tabular data set, **LOKI** provides users with a preliminary, default schema. As users confirm and/or override parts of the proposed schema, **LOKI** preserves the labels for the dataset's next user.

In this paper, we detail on our initial efforts to prime the **LOKI** knowledge-base with existing governmental open-data. Specifically...

2 SYSTEM DESIGN

The overall goal of **LOKI** is to streamline the process of developing schemas for existing unlabeled or poorly labeled data sets. As illustrated in Figure 1, **LOKI** lives alongside an existing RDBMS or Spark deployment, and takes as input tabular data in the form of a URL or HDFS file path. **LOKI** provides users with two modes of interaction: (1) A *labeling* interface that assists users in assigning names to existing columns of data, and (2) A *discovery* interface that helps users to search for columns representing particular concepts of interest. Both interfaces are supported by a knowledge-base that combines expert-provided heuristics, learned characteristics, as well as historical feedback gathered from users about already-loaded datasets. Once the user has labeled or discovered a sufficient set of columns, **LOKI** generates appropriate data loading/initialization code (e.g., a `CREATE TABLE` or Spark DataFrame initializer).

2.1 Labeling and Discovery

- Declare notation: Set of columns, Rule for column labeling, etc...
- Express the labeling problem in terms of this
- Express the discovery problem in terms of this

2.2 Approximate Matching Rules

Outline approximate matching rules: Expert heuristics, rules, etc...

- How are approximate KB entries encoded?
- How do we unify different types of heuristics?

2.3 Exact Matching Rules

Outline exact matching rules: Data identity, recording/querying feedback. Merging conflicts.

- How is feedback saved in the KB.
- What happens in response to conflicting feedback.

3 KNOWLEDGE-BASE

Overview sketching data for similarity.

Data type taxonomy

- Numeric Data (Sketch = Distribution)
- Textual Data (N-Gram distribution?,)
- Enum Types (Overlap, Concept-Similarity)
- ...?

3.1 Numeric Data

Focus on the challenges of sketching numeric types

4 EXPERIMENTS

Experiments

WOODSTOCK'97, July 1997, El Paso, Texas USA
2016. ACM ISBN 123-4567-24-567/08/06...$15.00
https://doi.org/10.475/123_4

5 RELATED WORK

1. Gestural Query Specification

Data similarity between two attributes is calculated for a join operation. The number of times that each value from one attribute appears in the other is counted and a histogram is constructed from the counts for all of the values. This histogram is then converted to a multinomial distribution and its probability is measured under a dirichlet distribution.

2. Wrangler

Wrangler infers the data type of a column and highlights errors based on inconsistent data types.

Unfold creates new column headers from data values. Wrangler lets analyst extract data from a column and creates a new column, but the analyst has to name the new column manually.

3. Potter's wheel

Let's user define custom domains and structures data accordingly. The only function required to be implemented is an inclusion function match to identify values in the domain.

Unfold âĂİunflattensâĂİ tables; it takes two columns, collects rows that have the same values for all the other columns, and unfolds the two chosen columns. Values in one column are used as column names to align the values in the other column.

4. Yago

The model must be able to express entities, facts, relations between facts and properties of relations.

5. Data Synthesizer

DataDescriber, investigates the data types, correlations and distributions of the attributes in the private dataset, and produces a data summary.

For each attribute, DataDescriber first detects whether it is numerical, and if so âĂŤ whether it is an integer or a float. If the attribute is non-numerical, DataDescriber attempts to parse it as datetime. Any attribute that is neither numerical nor datetime is considered a string.

6 FUTURE WORK

Future work...

- Meta-queries for columns that *could* be in a query.
- Discovery of meta-data (e.g., units)
- Automatic translation/transformation (units, structure – GPS vs Textual)

REFERENCES