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, 1 page. https://doi.org/10.475/123_4

1 INTRODUCTION

It's a story as old as time: Student gathers data, makes a graph with the data, writes a paper about the data, and then graduates. With the student gone, the data languishes. Without so much as a wiki page or README file documenting it, anyone who wants to re-use the data needs to spend hours, days, or even weeks reverse-engineering it. If we're lucky, that person documents their efforts. If not, the entire process repeats.

In this paper we propose *Label Once, and Keep It* (**LOKI**), a data-ingest middleware for incremental, re-usable schema recovery. When first pointed at un- or poorly labeled CSV data, **LOKI** uses an internal knowledge base to provide suggestions for schema elements like column names. Users ensure the labeling of a fragment of the schema that they are interested in,

Users identify a subset of the schema that they are interested in: table names, and other metadata documenting the

proposes an initial schema consisting of interpretable column names, a table name, and/or other contextual details.

When first invoked on an unlabeled , **LOKI** suggests a set of

The core of **LOKI** is a knowledge-base of

design and annotation of relational schemas.

We use a case study to evaluate the feasibility of LOKI. Specifically...

One more thought regarding a pitch for the work. We could wrap the idea in the context of a larger system for importing / querying initially unlabeled data. Specifically, when someone first loads an unlabeled (or only partially labeled) CSV file into a database/spark, they have two problems:

1) They need to label a subset of the columns that pertain to the specific analysis they want to do now. 2) They don't need to label *all* of the columns (might be 10s, 100s, or 1000s of columns that they don't care about).

However, at some point in the future, more labeling might be helpful. For example: 1) They pose a query and randomly discover that they are missing a column that *could* potentially exist in the source data. 2) Someone else wants to use the same data set, but with a different selection of columns. 3) The knowledge-base is updated and more automatic labelings become available.

I'm going to suggest that we present our contribution in the context of a system that: 1) Auto-suggests names for columns based on existing heuristics 2) Saves labeling efforts, making it possible to incrementally label a data-set and re-use effort across analyses 3) Allows you to ask whether a particular column name *could* exist in a given data set, and identify the data column that most-likely represents it.

Specifically, in this paper, we're conducting a case study evaluating one particular approach to task (1).

2 SYSTEM DESIGN

3 SKETCH SIMILARITY

4 EXPERIMENTS

Experiments

5 RELATED WORK

Related work

6 FUTURE WORK

REFERENCES

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