Synthesizing Data Programs

Michael Cafarella
University of Michigan
Ann Arbor, MI 48103
michjc@umich.edu

1. ABSTRACT

At least two important tasks in modern data management exist outside traditional database models and query languages: *data transformation* and *feature programs*. Data transformation is the informal preprocessing code that transforms raw data into a dataset that is appropriate for import into a relational database for deeper analysis. Feature programs transform raw data into a compact piece of training data that is suitable for use in a statistical training procedure. These two tasks are driven by two applications that are intellectually exciting, economically important, and which rightfully garner substantial attention in the database community: data analytics and machine learning.

Unfortunately, to date, both data transformation and feature programs have largely existed in an *ad hoc* netherworld of Python programs and shell scripts. Consider a stock trader engaged in an analytics task, who is interested in downloading a text file that describes large executive stock sales. The trader's intention is to identify "suspicious stocks" from the executiveSale table described by the text file, then perform a semijoin between the query answer and her preexisting stockHoldings table; the result will identify stocks that she wants to consider selling. But before she can do so, she must do some critical data transformation: she must write a regular expression to capture triples of stockSymbol, executiveName, sharesSold; then multiply the sharesSold number by the share price; then translate numeric values to the correct binary formats; then finally emit the results in a format that can be imported by her RDBMS.

Current practice is for the trader to write the above steps in a small but monolithic program in a mixture of Python, XPath, regular expressions, or perhaps some similar languages. It is perhaps not surprising that engineers use a range of tools here: a large part of the problem is that the input data can come described in almost any model, including informal ones. However justified, the resulting programs remain burdensome to write, easy to write incorrectly, hard to maintain in the face of changes to either the input text or the output schema, and opaque to the database optimizer.

Previous work has addressed some of the problems described above. Wrangler focused on data transformation for spreadsheet-style grids [2]. Wrapper induction work attempted, with some success, to induce input-custom text processors [3]. Researchers have proposed optimizers for text tasks outside the RDBMS [1]. However, none of these solutions has yet obtained widespread popularity. Moreover, it seems a shame to have a series of piecemeal solutions for what might be a single intellectual problem — automatically synthesizing programs that are usually:

1. Quite short.
2. Written in constrained languages, such as regular expressions.
3. Easy to check for correctness, by applying them to a known input and testing the program output. In the case of data transformation programs, we can test if the output matches the target relational schema. In the case of feature programs, we can test if the resulting training data improves a machine learning task.

We propose a system for *synthesizing data programs*. The system takes as input a description of the desired output language, some task-specific training examples, and some optional hints from the user. The system emits a small program in the target language. This system should be adaptable to a huge range of inputs and target languages. In practice, we expect that most users will choose one of several "off the shelf" target languages and only provide the system with data-specific assistance.

Researchers in the programming languages community have had some success with synthesis in similarly constrained settings, such as inferring correct multithreaded locking code [4]. Like those systems, we propose to model program synthesis as a constraint satisfaction problem. We have been building a data program synthesis system and have some promising initial results.

7th Biennial Conference on Innovative Data Systems Research (CIDR '15) January 4-7, 2015, Asilomar, California, USA.

2. REFERENCES

[1] P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. To search or to crawl?: towards a query optimizer for text-centric tasks. In *SIGMOD Conference*, pages 265–276, 2006.

[2] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In *CHI*, pages 3363–3372, 2011.

[3] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artif. Intell.*, 118(1-2):15–68, 2000.

[4] A. Solar-Lezama, C. G. Jones, and R. Bodík. Sketching concurrent data structures. In *PLDI*, pages 136–148, 2008.