Transforming Spreadsheets with Data Noodles

Maria I. Gorinova, Advait Sarkar, Alan F. Blackwell
Computer Laboratory, University of Cambridge
{mig30, advait.sarkar, alan.blackwell}@cl.cam.ac.uk

Karl Prince
Judge Business School, University of Cambridge
k.prince@jbs.cam.ac.uk

***Abstract*—Data wrangling is the term used by data scientists for the work of re-organising data into a new structure, before work starts on reporting or analysis. We present a prototype that applies programming by example methods to data wrangling in spreadsheets. The Data Noodles system guides the user through constructing a simple example that illustrates how they would like their spreadsheet to look. A transformation program is then synthesised and executed to produce the final reshaped spreadsheet.**

I. Introduction

Customisable information systems often result in complex databases, from which users need to extract data in a simpler form for further analysis. In recent work [1], we showed that hospital patient records would benefit from this kind of 'data wrangling' capability. We argued that recent program synthesis methods would allow end-users to select and reorganise data, if only an appropriate interaction metaphor were available. This paper presents a candidate metaphor, which we call Data Noodles.

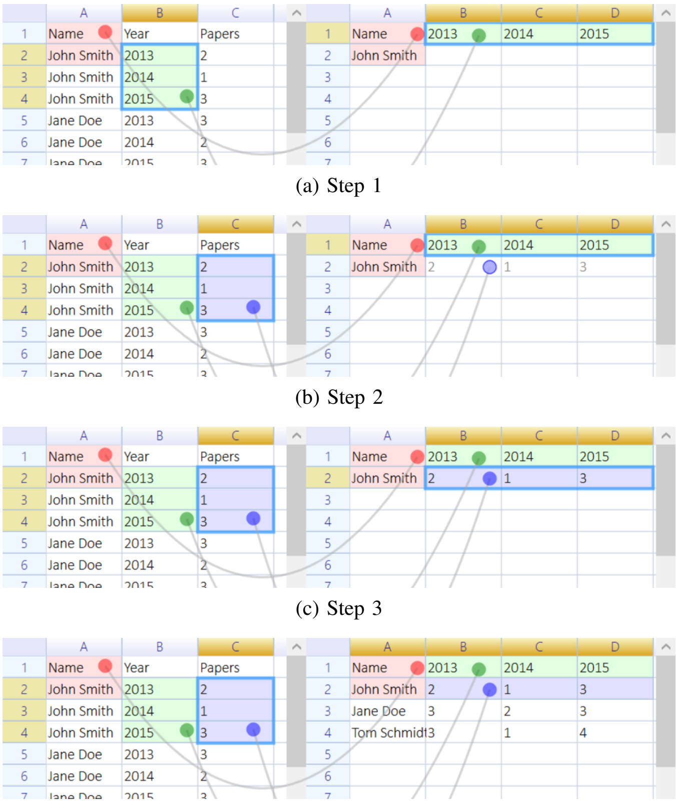
II. Design

The Data Noodles system generates programs to restructure spreadsheets by applying a series of transforms. The user constructs a simple example that illustrates what form they would like the spreadsheet to be in. Based on this example, a spreadsheet transformation program is automatically synthesised and executed to produce a spreadsheet with the desired shape. Below, we describe in more detail those two steps in the implementation of our approach.

A. Specifying the intended program

Menu-driven data wrangling interfaces expect users to select the sequence of individual transformations to be applied. However, users then must understand in advance the scope and effects of the available transformations. We reduce this barrier by asking the user only to describe what the overall result of the transformation should be, rather than individual steps. The user interacts with a split screen showing two spreadsheets – original input data on the left, and an output area on the right. The user first selects a range of cells on the left, producing a 'noodle' (hanging wire) that can be dragged to the right to specify the required transformation.

Fig. 1 demonstrates the process of constructing an example transformation. Each row in the input spreadsheet gives the number of papers published by an academic in a particular year. The desired output is to have only one row per person, with number of papers for each year in separate columns.



	A	B	C
1	Name	Year	Papers
2	John Smith	2013	2
3	John Smith	2014	1
4	John Smith	2015	3
5	Jane Doe	2013	3
6	Jane Doe	2014	2
7	Jane Doe	2015	3

	A	B	C	D
1	Name	2013	2014	2015
2	John Smith			
3				
4				
5				
6				
7				

(a) Step 1

	A	B	C
1	Name	Year	Papers
2	John Smith	2013	2
3	John Smith	2014	1
4	John Smith	2015	3
5	Jane Doe	2013	3
6	Jane Doe	2014	2
7	Jane Doe	2015	3

	A	B	C	D
1	Name	2013	2014	2015
2	John Smith	2	1	3
3				
4				
5				
6				
7				

(b) Step 2

	A	B	C
1	Name	Year	Papers
2	John Smith	2013	2
3	John Smith	2014	1
4	John Smith	2015	3
5	Jane Doe	2013	3
6	Jane Doe	2014	2
7	Jane Doe	2015	3

	A	B	C	D
1	Name	2013	2014	2015
2	John Smith	2	1	3
3				
4				
5				
6				
7				

(c) Step 3

	A	B	C
1	Name	Year	Papers
2	John Smith	2013	2
3	John Smith	2014	1
4	John Smith	2015	3
5	Jane Doe	2013	3
6	Jane Doe	2014	2
7	Jane Doe	2015	3

	A	B	C	D
1	Name	2013	2014	2015
2	John Smith	2	1	3
3	Jane Doe	3	2	3
4	Tom Schmidt	3	1	4
5				
6				
7				

(d) Step 4

Fig. 1: 'Unfolding' a spreadsheet so that each 'Year' value becomes a separate column.

Fig. 1a shows the spreadsheet after the user has defined two mappings: the red noodle maps the first column of the input spreadsheet to the first column of the output, while the green noodle maps values from the second input column to column labels in the output. To finish, the user must specify how data will be arranged into those columns. Fig. 1b shows a noodle (in blue) with the left end attached to values in the input sheet. The right end is 'active', offering alternative mappings to cells in the output spreadsheet. Hovering over cells on the right gives suggestions (in light grey) showing possible ways that these values might be modified or filtered by 'passing through' the noodle transformation channel. By clicking, the user accepts the displayed suggestion and places the right end of the noodle (Fig. 1c). The example is now complete, and the user can request the rest of the output spreadsheet to be filled

978-1-5090-0252-8/16/$31.00 ©2016 IEEE

in automatically (Fig. 1d). If the synthesised transformation is different from the one the user had in mind, parts of the output can be 'corrected' to refine the example and re-synthesise.

B. Transformation language and program synthesis

Our transformation language is inspired by the Potter's Wheel [2] and Wrangler [3] languages. It currently implements reshaping transformations such as Fold and Unfold, and we plan to add support for filtering, splitting and merging. In addition, we currently support an interpolation operation, which fills in missing values of a column by inferring a simple linear or nearest neighbour substitution from example text entered in that cell. The transformation program is synthesised using the PROSE SDK [4], a .NET framework for program synthesis of domain-specific languages. In applying PROSE to spreadsheets, we define the syntax and semantics of our transformation language, a set of 'witness functions' that give domain-specific insight to guide the synthesis process, and a set of 'scoring functions' used to rank candidate programs.

III. Previous work

Data wrangling tasks have previously been identified as a candidate for end-user programming methods. Tools such as PADS [5] and FlashExtract [6] allow users to extract structured data from semi-structured data, by either specifying the required format with a description language (PADS) or demonstrating it with an example (FlashExtract). Previous spreadsheet transformation systems use menus, demonstration or examples to specify a transformation program. Menu-driven tools such as Potter's Wheel [2] allow the user to create programs by specifying a sequence of operations. Wrangler [3] extends this approach with an interface that allows the user to demonstrate the scope of a operation by selecting columns or rows, after which the system suggests transformations. FlashRelate [7] allows users to give examples of the spreadsheet transformations they want to perform. The transformations are then automatically synthesised. Our aim is to unify these last two approaches, by providing an interface that prompts the user to specify an example of the effect of the desired transformation, automatically synthesises and executes a program that matches it, and allows the user to understand and amend this program when needed.

IV. Discussion

This approach draws on a long tradition of programming-by-example (PBE) where users demonstrate example behaviour using concrete data values, after which the system generalises for arbitrary input. In our spreadsheet context, we specify the example using a concrete subset of the data, which is then generalised to the entire spreadsheet. To minimise the burden of learning new representations, we make the PBE interface look like a spreadsheet. We indicate the transformation process with a familiar before-and-after metaphor from left to right, emphasised by the fact that the 'after' side is initially empty, and that the user must fill it in.

Drag and drop of cells from the input spreadsheet offers clear benefits: it is faster and less error-prone than copying text, and provides an unambiguous input cell reference. However, as the number of input-output examples grows, and the synthesised transformation becomes more complex, these transient operations are hard to remember and impossible to modify. By instead drawing lines that express the transformation, we express the dependencies between input and output, preserve and visualise the transient actions of the user, and offer a visual transformation language.

Rather than straight lines, we created a stylised rendering with physics-based animation effects, to give the impression of a noodle dangling between the input and output cells. Our goal here was to facilitate ludic engagement – an important new consideration for data wrangling tasks as they extend beyond professional data scientists to end-users [1]. Although program synthesis by example is technically complex, and presents new usability challenges, previous research has not given explicit consideration to hedonic pleasure in the design of data management tools. The playful appearance (and indeed, the name) of Data Noodles is intended to draw in and engage users who might otherwise find data manipulation a dry and unengaging task.

Although motivated by our research agenda that combines artistic live coding with scientific visualisation, we recognise the inevitable tradeoffs in the playful approach of Data Noodles. The curved trajectory might increase perceptual load, or impair scalability. Although complex transformations can be specified with only 5-10 noodles, how many can we include before creating an impossible-to-detangle noodle soup? The physical metaphor of catenary geometry with dynamic wobbling movement helps to provide perceptual cues, but we have made a deliberate trade-off of perceptual efficiency in favour of pleasurable engagement. Further study is required to investigate the value that this provides in our intended application domain.

Acknowledgment

The project is funded by the Health Foundation.

References

[1] M. I. Gorinova, K. Prince, S. Meakins, A. Vuylsteke, M. Jones, and A. F. Blackwell, "The end-user programming challenge of data wrangling." PPIG, 2016.

[2] V. Raman and J. M. Hellerstein, "Potter's wheel: An interactive data cleaning system," in *VLDB*, vol. 1, 2001, pp. 381–390.

[3] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, "Wrangler: Interactive visual specification of data transformation scripts," in *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. New York, New York, USA: ACM Press, 2011, p. 3363.

[4] O. Polozov and S. Gulwani, "Flashmeta: A framework for inductive program synthesis," in *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2015, pp. 107–126.

[5] K. Fisher and D. Walker, "The PADS project: An overview," in *Proceedings of the 14th International Conference on Database Theory - ICDT '11*. New York, New York, USA: ACM Press, 2011, p. 11.

[6] V. Le and S. Gulwani, "FlashExtract: A framework for data extraction by examples," *ACM SIGPLAN Notices*, vol. 49, no. 6, pp. 542–553, 2014.

[7] D. W. Barowy, S. Gulwani, T. Hart, and B. Zorn, "FlashRelate: extracting relational data from semi-structured spreadsheets using examples," *ACM SIGPLAN Notices*, vol. 50, no. 6, pp. 218–228, 2015.