VisualSynth: Democratizing Data Science in Spreadsheets*

Clément Gautrais[1], Yann Dauxais[1], Samuel Kolb[1], Arcchit Jain[1], Mohit Kumar[1], Stefano Teso[2], Elia Van Wolputte[1], Gust Verbruggen[1], and Luc De Raedt[1]

[1] KU Leuven, Department of Computer Science, Leuven, Belgium
firstname.lastname@cs.kuleuven.be
[2] University of Trento, Italy, stefano.teso@unitn.it

Abstract. We introduce VisualSynth, a framework that wants to democratize data science by enabling naive end-users to specify the data science tasks that match their needs. In VisualSynth, the user and the spreadsheet application interact by highlighting parts of the data using *colors*. The colors define a partial specification of a data science task (such as data wrangling or clustering), which is then completed and solved automatically using artificial intelligence techniques. The user can interactively refine the specification until she is satisfied with the result.

1 Introduction

Most data science techniques are beyond the reach of typical end-users. For instance, spreadsheets – a key tool in data management – are analyzed using formulas, which most users can neither write nor understand [2]. Hence, they are often processed manually, preventing scalability and increasing the chance of mistakes. The question we ask is then: *is it possible to enable non-technical end-users to specify and solve data science tasks that match their needs*? Our key observation is that the data science task of interest can often be identified by looking at the data and with a little help from the user. This observation enables us to design an interactive framework, VisualSynth[3], in which the machine and the user collaborate towards designing and solving a data science task.VisualSynth combines two components: a minimal interaction protocol that allows non-technical users to (partially) specify a data science task using color highlighting, and a smart framework for solving the partially specified data science task automatically, based on inductive models.

The intent of this setup is to combine the respective strengths of end-users, namely their domain knowledge, and of computers, namely their ability to quickly carry out large computations. In contrast to other automation approaches, like

* This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No [694980] SYNTH: Synthesising Inductive Data Models)

[3] Demonstration video: https://youtu.be/df6JgHl28Vw

AutoML [9,3], in VisualSynth the task to be performed (e.g., classification) is not assumed to be fixed or known. While interacting with spreadsheet users has been studied on a simple machine learning task [8], VisualSynth considers a wide range of data science tasks using a unified interaction system.

2 Interacting Using Colors

Given a spreadsheet, we define a *sketch* to be a set of colors applied to one or more rows, columns, or cells in the spreadsheet (see Fig. 1, left). VisualSynth employs colors to define the type, inputs, outputs and other parameters of a data science task. Taken together, the sketch and the spreadsheet can be mapped onto a concrete *data science task* (e.g., a wrangling task), which can then be solved and whose results (e.g., a set of new tables) can be placed into the original spreadsheet, yielding an extended spreadsheet. This idea is captured in the following schema: $\left.\begin{smallmatrix}\text{spreadsheet}\\+\\\text{sketch}\end{smallmatrix}\right\} \rightarrow \begin{smallmatrix}\text{data science}\\\text{task}\end{smallmatrix} \rightarrow \left.\begin{smallmatrix}\text{spreadsheet}\\+\\\text{model}\end{smallmatrix}\right\} \rightarrow \begin{smallmatrix}\text{new}\\\text{spreadsheet}\end{smallmatrix}$ Let us illustrate this schema for wrangling, the task of transforming data in the right format for downstream data science tasks. The user indicates which cells belong to the same row by coloring them using the same color [10]. A *wrangling sketch* is therefore a set of colored cells, where each color defines a partial example of the expected wrangling result and imposes a constraint on the output, i.e., that the partial example should be mapped onto a single row into the target spreadsheet. Figure 1 left presents an example of a wrangling sketch. A commonly used paradigm for data wrangling is programming by example (PBE) [5], in which a language of transformations $\mathcal{L}$ is defined and the wrangler searches for a program $P \in \mathcal{L}$ that maps the input examples to the corresponding outputs. In the context of VisualSynth, the task is: **Given** a wrangling sketch and a spreadsheet, **Find** a program that transforms the spreadsheet so that cells with the same color end up in the same row, and no row can contain cells with multiple colors. In Figure 1 left, the task is: **Given** the green and orange colorings, a spreadsheet, a language $\mathcal{L}$, **Find** a wrangling program $P \in \mathcal{L}$ such that green cells end up in a single row, and orange cells in another single row. Figure 1 middle shows the output of the wrangling program P. We refer the reader to [4] for a more in-depth presentation of sketches and their uses for different data science tasks.

3 Use Case: Ice Cream Sales Auto-completion

Let us illustrate VisualSynth using an auto-completion task. Imagine that you are a sales manager at an ice cream factory. You have data about past sales and about your shops, as shown in Figure 1. A first difficulty is that the sales is not nicely formatted. A first task is therefore to wrangle it into a format that is more amenable to data analysis. A sketch for wrangling consist in coloring cells that should belong to a single row with a single color. As soon as the user starts putting colors on cells, VisualSynth automatically suggest a wrangling task

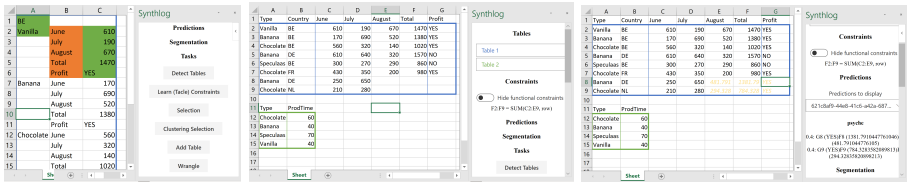

Fig. 1: Left: view of the VisualSynth add-in before performing the wrangling. User hints are represented as colors, that VisualSynth uses to infer a wrangling task (Section 2). The right panel displays a button for each suggested task. Middle: VisualSynth add-in after the wrangling execution. Constraints have also been learned by clicking on the associated button. The right panel displays the learned constraints. Right: view of the VisualSynth add-in after autocompletion has been performed. The add-in displays learned constraints, as well as the predictions and their confidence.

to the user in the add-in panel. The user can then perform the suggested task by clicking on the associated button in our add-in. Through this interaction, the data wrangling component can produce the table presented in Figure 1.

As this table contains segments of consecutive numerical values, VisualSynth suggests to learn Excel constraints. By clicking on the associated button, Excel constraints will be detected using TaCLe [6], a spreadsheet constraint learner. The learned constraints are then displayed to the user who can inspect them. In this case the system learned that *Total=June + July + August*.

However, some past sales data are still missing. To determine which shops made a profit you need to obtain an estimate of the missing values. As the table contains missing values, VisualSynth suggests the Auto-completion task. A key feature of Auto-completion is that missing values are inferred using one or more predictive models (often classifiers or regressors [1]) while ensuring that predictions are compatible with the formulas and the constraints detected in the spreadsheet [7]. For example, in Figure 1 right, the values inferred respect the constraint *Total=June+July+August*. If you click on the associated button, the machine starts suggesting values, which you can then either accept as is or correct them. Corrections trigger a new auto-completion loop, with additional constraints expressing that the user corrected some values.

This use case featured only a few of VisualSynth capabilities but it can handle a number of key data science tasks, namely data wrangling, concept learning, data selection, prediction, clustering, constraint learning and auto-completion. This use case also shows how VisualSynth uses sketches to perform non trivial data science tasks in Excel, a software that is familiar to spreadsheet users.

4 Architecture

VisualSynth is implemented as a Microsoft Excel© add-in (see Figure 2). The add-in is implemented with React (https://reactjs.org), the server with Node.js© (https://nodejs.org), and the Synth library with Python. Tasks can be implemented in any language, as long as their output correspond to state components.

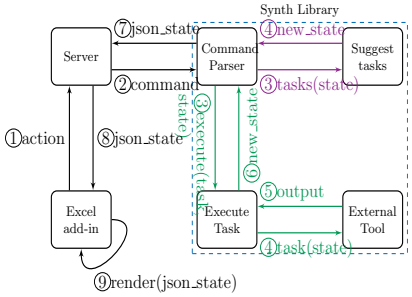

Fig. 2: Architecture of VisualSynth. A state contains the spreadsheet data, the learned models and the interactions with the user: state = data + models + interactions. When an action is performed in Excel, VisualSynth calls the Synth library through a server to perform a function. There are 2 main types of functions in the Synth library: 1) Getting task recommendations given the current state (in purple), and 2) Executing a task (in green). In both cases, the Synth library returns a new state, which is passed to the Excel add-in and rendered by mapping state components (tables, models, predictions...) to a visual element (border, font or fill formatting of a cell). For example, in Figure 1, a table is mapped to a range border in Excel, the confidence of a cell value prediction is mapped to the cell font color. Task are recommended using the state [4].

References

1. Bishop, C.M.: Pattern recognition and machine learning. springer (2006)
2. Chambers, C., Scaffidi, C.: Struggling to excel: A field study of challenges faced by spreadsheet users. In: 2010 IEEE Symposium on Visual Languages and Human-Centric Computing. pp. 187–194. IEEE (2010)
3. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in neural information processing systems. pp. 2962–2970 (2015)
4. Gautrais, C., Dauxais, Y., Teso, S., Kolb, S., Verbruggen, G., De Raedt, L.: Human-machine collaboration for democratizing data science. arXiv preprint arXiv:2004.11113 (2020)
5. Halbert, D.C.: Programming by example. Ph.D. thesis, University of California, Berkeley (1984)
6. Kolb, S., Paramonov, S., Guns, T., De Raedt, L.: Learning constraints in spreadsheets and tabular data. Machine Learning (Jun 2017)
7. Kolb, S., Teso, S., Dries, A., De Raedt, L.: Predictive spreadsheet autocompletion with constraints. Machine Learning pp. 1–19 (2019)
8. Sarkar, A., Jamnik, M., Blackwell, A.F., Spott, M.: Interactive visual machine learning in spreadsheets. In: 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 159–163. IEEE (2015)
9. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD. pp. 847–855. ACM (2013)
10. Verbruggen, G., De Raedt, L.: Automatically wrangling spreadsheets into machine learning data formats. In: International Symposium on Intelligent Data Analysis. pp. 367–379. Springer (2018)