Data Quality Challenge: Toward a tool for string processing by examples

ALBERTO BARTOLI, DIA, University of Trieste, Italy
ANDREA DE LORENZO, DIA, University of Trieste, Italy
ERIC MEDVET, DIA, University of Trieste, Italy
FABIANO TARLAO, DIA, University of Trieste, Italy

1. INTRODUCTION

Many data-related activities at organizations of all sizes are concerned with low-level string processing, such as format transformation and validation, data cleaning, substring extraction and classification, and so on. Problems of this sort occur routinely in a *one-off* fashion as part of specific processes or activities that cannot be integrated in long-lived workflows, such as analysis of data gathered from the web or from other enterprise sources. The input stream may range from structured or semistructured data, such as database tables or spreadsheets, to unstructured data, such as text in natural language, as well as data whose structure is not available, such as a web page or a pdf invoice. These tasks are a vital ingredient of virtually every organization but are difficult to address efficiently: they are usually too simple to justify the cost and latency of a full-blown IT project, yet they are not simple enough to be solved by non-IT specialists.

Motivated by these considerations, several proposals have emerged for enabling forms of string transformation and extraction by providing only *examples* of the desired behavior, thereby eliminating the need of developing a dedicated program or script. Researchers from SAP AG have developed an approach that infers regular expressions for extracting business-relevant data automatically, based solely on a set of examples derived from enterprise data, such as, e.g., a product catalog or historical invoices [Brauer et al. 2011]. A group of researchers from IBM Almaden and the University of Michigan had proposed a similar approach earlier, in which a starting regular expression was automatically improved based on the provided examples of the desired behavior [Li et al. 2008]. Microsoft Research has recently developed sophisticated and powerful algorithms for string transformation [Gulwani 2011] and information extraction [Le and Gulwani 2014] by examples, delivered in Excel 2013 (FlashFill) and in the latest Windows PowerShell preview (ConvertFrom-String), respectively. Another group in Microsoft Research has investigated the use of crowdsourcing for solving string classification tasks specified by examples [Cochran et al. 2015]. Google recently released an add-on for Google Sheets capable of completing partially filled columns based on data on other columns [Davydov and Rostamizadeh 2014]. Other powerful tools for interactive execution of data transformations based on examples include [Raman and Hellerstein 2001; Miller and Myers 2002; Guo et al. 2011; Wu et al. 2014]. We also contributed in this area with evolutionary algorithms having very good generalization capabilities and robustness against noisy data [Bartoli et al. 2014; De Lorenzo et al. 2013].

Proposals of this kind are extremely promising for the one-off scenarios mentioned above and embed a great potential for business developers, i.e., for people whose primary job function is not application development but routinely have very specific business needs that they have to satisfy with applications developed by themselves [Gualtieri 2011]. Indeed, string transformation and extraction by examples may constitute a significant step forward for many daily IT-related activities, but actually exploiting this potential in a practical tool is very challenging. Each of the above mentioned proposals is tailored to a specific kind of processing; no library embeds a broad range of methods addressing different needs; quality of the results depends strongly on the input data and may often be too low to be acceptable in practice. Moreover, understanding what can and what cannot be realistically obtained with each of those proposals on a given dataset is quite difficult.

2. CHALLENGES AND RESEARCH DIRECTIONS

There have been many *programming by examples* proposals in the past [Smith et al. 2000] but it is fair to claim that none of them has had any widespread impact in practice. We believe a key reason for those failed attempts is that they aimed at supporting a general programming framework, rather than at fitting specific domains. As such, those failures should not discourage further research but should instead drive researchers to the development of highly *specialized* tools. To this end, one should identify a delicate balance between capability of the processing engine, practical usefulness of those capabilities and power of the learning machinery. Figuring out suitable trade-offs between these axes of the design space may be the key toward a practical solution.

There are also fundamental challenges related to the role of the examples of the desired behavior that the users provide to the system. In particular, the fundamental tension between *generalization* and *overfitting* is likely to be crucial in this context. Suppose, for example, that a user wants to anonymize the email addresses in a large collection of documents and provide examples in which all the email addresses happen to belong to a `.com` domain, or happen to be all 20 characters long, or happen to have a dot character in the portion preceding the `@`. Enabling the learning machinery to infer the general pattern that the user had in mind is clearly very difficult. Indeed, it may even be the case that the user actually requires an overfitting of the provided examples, i.e., only the very same input-output pairs provided in the examples have to be processed and any other possible input is to be left unchanged. There are many questions to be investigated in this respect: can a single tool accommodate such different needs effectively? should we expect the user to provide further pieces of information beyond examples of the desired input-output behavior? should that information be provided in advance or interactively? how should the user and the tool interact? We speculate that for many of those questions there is no single universally valid answer.

There are also fundamental challenges related to the *difficulty* of a given problem instance. For example, is inferring a pattern for extracting IP addresses more or less difficult than inferring one for extracting email addresses? Is inferring a pattern for extracting stock analyses more or less difficult than inferring one for extracting basketball statistics? How does the difficulty of inferring a given pattern depend on the provided examples? How does the difficulty of identifying a given pattern depend on the properties of the surrounding data that is *not* to be extracted or processed? Are the data to be processed *noisy* with respect to the pattern one is interested in? There is currently no principled way for providing useful answers to questions of this sort—the only approach being “try and see what happens”. Indications of this sort would certainly be extremely useful for gaining insights into the current state of the art and for driving further research. They may also be highly useful to end users, as an indication of what may or may not be “practically feasible” with a given dataset, or for figuring out whether the user-provided examples are “adequate” for the task.

Finally, establishing a common benchmark suite representative of practically relevant applications and drawn from a broad range of different problem classes may certainly be of great help to move this field forward. Such a suite should be challenging enough to prevent rewarding trivial solutions, while at the same time not being exceedingly ambitious.

REFERENCES

Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Eric Medvet, and Enrico Sorio. 2014. Automatic Synthesis of Regular Expressions from Examples. *Computer* 47, 12 (Dec 2014), 72–80. DOI:http://dx.doi.org/10.1109/MC.2014.344

Falk Brauer, Robert Rieger, Adrian Mocan, and Wojciech M. Barczynski. 2011. Enabling Information Extraction by Inference of Regular Expressions from Sample Entities. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*. ACM, New York, NY, USA, 1285–1294. DOI:http://dx.doi.org/10.1145/2063576.2063763

Robert A. Cochran, Loris D'Antoni, Benjamin Livshits, David Molnar, and Margus Veanes. 2015. Program Boosting: Program Synthesis via Crowd-Sourcing. In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '15)*. ACM, New York, NY, USA, 677–688. DOI:http://dx.doi.org/10.1145/2676726.2676973

Konstantin Davydov and Afshin Rostamizadeh. 2014. Smart Autofill - Harnessing the Predictive Power of Machine Learning in Google Sheets. http://googleresearch.blogspot.it/2014/10/smart-autofill-harnessing-predictive.html. (Oct. 2014).

Andrea De Lorenzo, Eric Medvet, and Alberto Bartoli. 2013. Automatic String Replace by Examples. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*. ACM, New York, NY, USA, 1253–1260. DOI:http://dx.doi.org/10.1145/2463372.2463532

Mike Gualtieri. 2011. *Empowering The "Business Developer"*. Technical Report. Forrester Research.

Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Notices*, Vol. 46. ACM, 317–330.

Philip J Guo, Sean Kandel, Joseph M Hellerstein, and Jeffrey Heer. 2011. Proactive wrangling: Mixed-initiative end-user programming of data transformation scripts. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 65–74.

Vu Le and Sumit Gulwani. 2014. FlashExtract: A Framework for Data Extraction by Examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. ACM, New York, NY, USA, 542–553. DOI:http://dx.doi.org/10.1145/2594291.2594333

Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and H. V. Jagadish. 2008. Regular Expression Learning for Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 21–30. http://dl.acm.org/citation.cfm?id=1613715.1613719

Robert C Miller and Brad A Myers. 2002. Multiple selections in smart text editing. In *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM, 103–110.

Vijayshankar Raman and Joseph M Hellerstein. 2001. Potter's wheel: An interactive data cleaning system. In *Proceedings of the 27-th VLDB Conference*, Vol. 1. 381–390.

David Canfield Smith, Allen Cypher, and Larry Tesler. 2000. Programming by Example: Novice Programming Comes of Age. *Commun. ACM* 43, 3 (March 2000), 75–81. DOI:http://dx.doi.org/10.1145/330534.330544

Bo Wu, Pedro Szekely, and Craig A Knoblock. 2014. Minimizing user effort in transforming data by example. In *Proceedings of the 19th international conference on Intelligent User Interfaces*. ACM, 317–322.