General-Purpose Inductive Programming for Data Wrangling Automation

Lidia Contreras-Ochando, Fernando Martínez-Plumed, Cèsar Ferri,
José Hernández-Orallo, María José Ramírez-Quintana
Universitat Politècnica de València,
Camí de Vera s/n, 46022 València, Spain.
{liconoc,fmartinez,cferri,jorallo,mramirez}@dsic.upv.es

Abstract

Data acquisition, integration, transformation, cleansing and other highly tedious tasks take a large proportion of data science projects. These routine tasks are tedious basically because they are repetitive and, hence, automatable. As a consequence, progress in the automation of this process can lead to a dramatic reduction of the cost and duration of data science projects. Recently, Inductive Programming (IP) has shown a large potential as a paradigm for addressing this automation. This short paper elaborates on the recent success of induction using domain-specific languages (DSLs) for the automation of data wrangling process and advocating for the use of inductive programming over general-purpose declarative languages (GPDLs) using domain-specific background knowledge (DSBKs).

Extended abstract

Raw data in data science applications is usually messy since it usually comes in a mix of structured and unstructured formats from different sources. *Data wrangling* refers to the process of acquiring, integrating, manipulating, cleansing, enriching and transforming data from its raw format to a more structured and valuable form for easy access and analysis.

It is estimated that data wrangling, which is still carried on by means of ETL tools and scripting languages, spends a great proportion (50%-80%) of the time devoted to data science projects [Steinberg, 2013]. Progress in the automation of this process is crucial to the efficiency of data-oriented projects. In fact, there are some tools that try to automate the knowledge discovery process such as AutoWeka [Thornton et al., 2013], AutoML [Aut, 2016] or DataRobot [Dat, 2016]. However, these initiatives are focused primarily on the modeling phase which, compared to the data wrangling one, has relatively low cost and duration. Other companies, such as *Trifacta*, *Tamr* and *Paxata*, focus on data wrangling but most of their automation efforts are aiming at the analysis of semi-structured and structured input data by means of innovative visual analysis and machine learning techniques for merging and extracting relationships across data sets. In contrast, unstructured or free-form data is less likely to be automatically processed due to the fact that it may require human intervention to process and shape the data into a machine-readable form.

Recently, *inductive programming* [Kitzelmann and Schmid, 2006] has also shown a large potential for data wrangling automation. In contrast to standard machine learning approaches, inductive programming addresses the problem of learning small (but complex) programs from a few representative input/output examples, generated as the user transforms one (or very few) particular instances or fields of the data. The release of killer applications for data wrangling such as *FlashFill* [Gulwani, 2011] (a tool for automating repetitive string transformations), *FlashExtract* [Le and Gulwani, 2014] (a tool for extracting structured data from semi-structured text/log files and webpages) and *FlashRelate* [Barowy et al., 2015] (a tool for extracting tabular/relational data out of semi-structured spreadsheets) is a

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

demonstration that inductive programming research has matured in such a way that these applications are becoming feasible [Gulwani et al., 2015].

It should be noted that all of these recent demonstrations are based on domain-specific languages (DSLs), i.e., languages that are defined for a particular kind of processing (e.g., string processing, number processing, etc.), instead of using general-purpose declarative (programming) languages (GPDL) jointly with the inductive programming tools created for them during the last two decades, such as Progol [Muggleton, 1995], MagicHaskeller [Katayama, 2012], FLIP [Ferri et al., 2001], Metagol [Muggleton et al., 2015], gErl [Martínez-Plumed et al., 2013] and many others. Inductive programming using GPDLs is usually inefficient and/or incomplete because of the large search space, as they are not specialised for a particular domain. However, with the correct definition of a reduced library of functions (or predicates) in a domain-specific background knowledge (DSBK), the search space for these generic inductive programming tools can be bounded by the size of the solution and the number of functions in this DSBK. In other words, in theory, the use of inductive programming using GPDL + DSBK can be as powerful as the use of inductive programming tools that are specifically designe for a particular DSL.

For instance, let us consider the problem of the automatic conversion between strings that contain dates [Singh and Gulwani, 2016]. Here, the goal is to automatically transform strings that express dates (in different formats) into the standard format “$dd/mm/yyyy$”. We consider the functional language *Haskell* as the general-purpose declarative language (GPDL) where we can induce the transformation functions, using an inductive programming tool such as MagicHaskeller. First we need to define the functions that can capture the background knowledge that humans have about dates. This DSBK must include functions that perform type conversions and numeric validations, specialised to the problem domain. In Haskell, a sample of such set of functions follows:

```haskell
-- Background Knowledge Functions
import Data.Char
import Data.List.Split
-- read :: String -> Int
-- show :: Int -> String
-- splitOn :: Eq a => [a] -> [a] -> [[a]]
-- (++) :: [a] -> [a] -> [a]
completeyear a = show (2000 + read (a))
ismonth numb = if (read(numb)>0 && read (numb)<12) then True else False
isday numb = if (read (numb)>0 && read (numb)<32) then True else False
getsep [x] = ’?’
getsep (x:xs)= if (isDigit x) then getsep xs else x
splitOnNonNumeric cad= splitOn [getsep(cad)] cad
```

Now consider that we need to automatically turn dates into the standard format “$dd/mm/yyyy$” from two different scenarios a) strings formatted as “*dd-mm-yy*” (e.g. “31-02-11”), and b) strings formatted as “$mm/dd/yy$” (e.g. “05/31/11”). Using the DSBK defined previously, and some examples of the conversions, state-of-art inductive functional programming (e.g., [Ferri et al., 2001, Katayama, 2012, Kitzelmann and Schmid, 2006]) with a search space constrained by the DSBK could be used to attempt to induce two functional programs in *Haskell* that are able to perform the transformations, as follows.

```haskell
-- Induced Programs
-- "31-02-11" -> "31/02/2011"
convert1 (md) =let (d:m:y:[])= splitOnNonNumeric md) in
                 if (ismonth m && isday d)
                   then d ++ "/" ++ m ++ "/" ++ completeyear y else "NA"
-- "05/31/11" -> "31/05/2011"
convert2 (md) =let (d:m:y:[])= splitOnNonNumeric md) in
                 if (ismonth m && isday d)
                   then d ++ "/" ++ m ++ "/" ++ completeyear y else "NA"
```

The advantages of using this approach are manifold. First, the same data wrangling tool with inductive programming can be used for diversity of problems and domains, without specialised tools for every domain. Second, a library of DSBKs could be provided with the data wrangling tools using inductive programming. The user just needs to suggest which one to use for a particular problem: dates, times, emails, names, cities, addresses, etc. Since the languages for creating the DSBK are general-purpose and well known (Haskell, Prolog, etc.), users can create their own DSBKs and share them with other users to help them automate their data wrangling transformations.

Acknowledgements

This work has been partially supported by the EU (FEDER) and Spanish MINECO grant TIN2015-69175-C4-1-R, and the REFRAME project, granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences Technologies ERA-Net (CHIST-ERA), and funded by MINECO in Spain (PCIN-2013-037) and by Generalitat Valenciana PROMETEOII/2015/013.

References

AutoML: taking the human expert out of the loop, 2016. URL `http://www.automl.org/`.

DataRobot: Machine learning software, 2016. URL `https://www.datarobot.com/`.

D. W. Barowy, S. Gulwani, T. Hart, and B. G. Zorn. Flashrelate: extracting relational data from semi-structured spreadsheets using examples. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, June 15-17, 2015*, pages 218–228, 2015.

C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Incremental learning of functional logic programs. In H. Kuchen and K. Ueda, editors, *Functional and Logic Programming*, volume 2024 of *Lecture Notes in Computer Science*, pages 233–247. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41739-2. doi: 10.1007/3-540-44716-4_15.

S. Gulwani. Automating string processing in spreadsheets using input-output examples. In *Proc. 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '11, pages 317–330, New York, NY, USA, 2011. ACM.

S. Gulwani, J. Hernandez-Orallo, E. Kitzelmann, S. H. Muggleton, U. Schmid, and B. Zorn. Inductive programming meets the real world. *Communications of the ACM*, 58(11):90–99, 2015.

S. Katayama. An analytical inductive functional programming system that avoids unintended programs. In *Proceedings of the ACM SIGPLAN 2012 workshop on Partial evaluation and program manipulation*, pages 43–52. ACM, 2012.

E. Kitzelmann and U. Schmid. Inductive synthesis of functional programs: An explanation based generalization approach. *Journal of Machine Learning Research*, 7(Feb):429–454, 2006.

V. Le and S. Gulwani. Flashextract: A framework for data extraction by examples. In *ACM SIGPLAN Notices*, volume 49, pages 542–553. ACM, 2014.

F. Martínez-Plumed, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Learning with configurable operators and rl-based heuristics. In A. Appice, editor, *New Frontiers in Mining Complex Patterns*, volume 7765 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2013.

S. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13(3-4):245–286, 1995. ISSN 0288-3635. doi: 10.1007/BF03037227.

S. H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.

R. Singh and S. Gulwani. Transforming spreadsheet data types using examples. *ACM SIGPLAN Notices*, 51(1):343–356, 2016.

D. Steinberg. How much time needs to be spent preparing data for analysis? *http://info.salford-systems.com/blog/bid/299181/How-Much-Time-Needs-to-be-Spent-Preparing-Data-for-Analysis*, 2013.

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In *KDD*, 2013.