Applied Soft Computing 46 (2016) 398–406

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Predicting the effectiveness of pattern-based entity extractor inference

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet*, Fabiano Tarlao

DIA, University of Trieste, Italy

ARTICLE INFO

Article history:
Received 15 October 2015
Received in revised form 24 March 2016
Accepted 14 May 2016
Available online 20 May 2016

Keywords:
String similarity metrics
Information extraction
Genetic programming
Hardness estimation

ABSTRACT

An essential component of any workflow leveraging digital data consists in the identification and extraction of relevant *patterns* from a data stream. We consider a scenario in which an *extraction inference engine* generates an *entity extractor* automatically from examples of the desired behavior, which take the form of user-provided annotations of the entities to be extracted from a dataset. We propose a methodology for *predicting* the accuracy of the extractor that may be inferred from the available examples. We propose several prediction techniques and analyze experimentally our proposals in great depth, with reference to extractors consisting of regular expressions. The results suggest that reliable predictions for tasks of practical complexity may indeed be obtained quickly and without actually generating the entity extractor.

1. Introduction

An essential component of any workflow leveraging digital data consists in the identification and extraction of relevant *patterns* from a data stream. This task occurs routinely in virtually every sector of business, government, science, technology, and so on. In this work we are concerned with extraction from an unstructured text stream of entities that adhere to a *syntactic* pattern. We consider a scenario in which an extractor is obtained by tailoring a generic tool to a specific problem instance. The extractor may consist, e.g., of a regular expression, or of an expression in a more general formalism [1], or of full programs suitable to be executed by NLP tools [2,3]. The problem instance is characterized by a dataset from which a specified entity type is to be extracted, e.g., VAT numbers, IP addresses, or more complex entities.

The difficulty of generating an extractor is clearly dependent on the specific problem. However, we are not aware of any methodology for providing a practically useful answer to questions of this sort: generating an extractor for describing IP addresses is more or less difficult than generating one for extracting email addresses? Is it possible to generate an extractor for drug dosages in medical recipes, or for ingredients in cake recipes, with a specified accuracy level? Does the difficulty of generating an extractor for a specified entity type depend on the properties of the text that is *not* to be extracted? Not only answering such questions may provide crucial insights on extractor generation techniques, it may also be of practical interest to end users. For example, a prediction of low effectiveness could be exploited by providing more examples of the desired extraction behavior; the user might even decide to adopt a manual approach, perhaps in crowdsourcing, for problems that appear to be beyond the scope of the extractor generation technique being used.

In this work we propose an approach for addressing questions of this sort systematically. We consider on a scenario of increasing interest in which the problem instance is specified by *examples* of the desired behavior and the target extractor is generated based on those examples automatically [4–12]. We propose a methodology for *predicting* the accuracy of the extractor that may be inferred by a given extraction inference engine from the available examples. Our prediction methodology does not depend on the inference engine internals and can in principle be applied to any inference engine: indeed, we validate it on two different engines which infer different forms of extractors.

The basic idea is to use string similarity metrics to characterize the examples. In this respect, an "easy" problem instance is one in which (i) strings to be extracted are "similar" to each other, (ii) strings not to be extracted are "similar" to each other, and (iii) strings to be extracted are not "similar" to strings not to be extracted. Despite its apparent simplicity, implementing this idea is highly challenging for several reasons.

To be practically useful, a prediction methodology shall satisfy these requirements: (a) the prediction must be reliable; (b) it must

* Corresponding author.
E-mail addresses: bartoli.alberto@units.it (A. Bartoli), andrea.delorenzo@units.it (A. De Lorenzo), emedvet@units.it (E. Medvet), fabiano.tarlao@phd.units.it (F. Tarlao).

http://dx.doi.org/10.1016/j.asoc.2016.05.023
1568-4946/

be computed without actually generating the extractor; (c) it must be computed very quickly w.r.t. the time taken for inferring the extractor. First and foremost, predicting the performance of a solution without actually generating the solution is clearly very difficult (see also the related work section).

Second, it is not clear to which degree a string similarity metric can capture the actual difficulty in inferring an extractor for a given problem instance. Consider, for instance, the Levenshtein distance (string edit distance) applied to a problem instance in which entities to be extracted are dates. Two dates (e.g., `2-3-1979` and `7-2-2011`, whose edit distance is 6) could be as distant as a date and a snippet not to be extracted (e.g, `2-3-1979` and `19.79$`, whose edit distance is 6 too); yet dates could be extracted by an extractor in the form of regular expression that is very compact, does not extract any of the other snippets and could be very easy to generate (`\d+-\d+-\d+`). However, many string similarity metrics exist and their effectiveness is tightly dependent on the specific application [13,14]. Indeed, one of the contributions of our proposal is precisely to investigate which metric is the most suitable for assessing the difficulty of extractor inference.

Third, the number of snippets in an input text grows quadratically with the text size and becomes huge very quickly—e.g., a text composed of just 10^5 characters includes $\approx 10^{10}$ snippets. It follows that computing forms of similarity between all pairs of snippets may be feasible for snippets that are to be extracted but is not practically feasible for snippets that are *not* to be extracted.

We propose several prediction techniques and analyze experimentally our proposals in great depth, with reference to a number of different similarity metrics and of challenging problem instances. We validate our techniques with respect to a state-of-the-art extractor generator[1] approach that we have recently proposed [9,5,6]; we further validate our predictor on a worse-performing alternative extractor generator [15] which works internally in a different way. The results are highly encouraging suggesting that reliable predictions for tasks of practical complexity may indeed be obtained quickly.

2. Related work

Although we are not aware of any work specifically devoted to predicting the effectiveness of a pattern-based entity extractor inference method, there are several research fields that addressed similar issues. The underlying common motivation is twofold: inferring a solution to a given problem instance may be a lengthy procedure; and, the inference procedure is based on heuristics that cannot provide any optimality guarantees. Consequently, lightweight methods for estimating the quality of a solution before actually generating that solution are highly desirable.

In *combinatorial optimization* a wealth of research efforts have been devoted to the problem of estimating the difficulty of a given problem instance [16]. Such efforts may be broadly categorized in two classes: identifying features of a problem instance which may impact difficulty in terms of quality of a solution; and, identifying problem instance-independent features that may help in characterizing the difficulty of a task in general.

The work in [17] considers a specific class of combinatorial optimization tasks (TSP: traveling salesman problem) and follows a different line of research aimed at identifying features of a problem instance that may be helpful in choosing from a portfolio of algorithms the best one for that instance. The cited work actually considers only two such algorithms and assesses the ability of several classifiers, trained on a number of problem instances, to predict the relative performance of these two algorithms.

A recent proposal in this area followed a common approach consisting in the generation of a number of problem instances for a specific problem class (TSP) by means of a parametrized generation method [18]. A regressor for the solutions was then generated by using features of each problem instance that included values for the generation parameter. Our approach is similar except that we address a radically different task, thereby calling for radically different features.

An indirect but strong indication that the problem that we are addressing is amenable only to heuristic solutions without any formal guarantee is provided in [19], which considers optimization problems and proves that predictive measures that run in polynomial time do not exist.

Problem difficulty prediction is a very important research topic in *evolutionary computation*: an excellent survey can be found in [20]. The cited work presents a general method for estimating performance of evolutionary program induction algorithms with an experimental evaluation on two important classes of tasks, i.e., symbolic regression and Boolean function induction. The method is based on regressors trained on features extracted from problem instances. Features are defined over forms of distances computed over input–output pairs of the problem instance. We are not aware of any instantiation of this method for application domains involving string similarity computations, where there are many metrics that can be used and their effectiveness is tightly dependent on the specific task (e.g., [21,22]).

A systematic analysis of a number of measures aimed at characterizing the difficulty of a *classification* problem is presented in [23]. In principle, this analysis could be applied also to text extraction problems, because such problems require classifying each individual character in a stream depending on whether the character is to be extracted. On the other hand, the cited work focuses on the geometrical properties of classification, considering measures that may highlight the separation between classes in the measurement space. Text extraction problems are generally not suitable to interpretations of this kind.

Performance prediction is an important research topic in *information retrieval*, aimed at assessing effectiveness of a query before or during early stages of retrieval [24–27]. Methods in this area generally require an indexing phase of the document corpus and then emit a prediction for a query based on a quick comparison between query terms and various indexed structures [28] (a corpus-independent approach is proposed in [29]).

As mentioned above, the effectiveness of a given string similarity metrics is usually highly dependent on the specific class of task. For this reason, we apply our proposal on a number of different metrics following an approach taken in other application domains. Several preprocessing strategies in combination with a variety of similarity metrics were assessed with reference to *ontology alignment* task [13]. The focus was finding the combination which exhibits best performance on a wide selection of problem instances representative of the ontology alignment task. A number of similarity metrics proposed by various research communities were applied to the task of matching entity names to database records [14]. The focus was finding the metric most suitable to the specific task. The key difference from our approach is that we investigate different string metrics as a tool for predicting the quality of a solution. The solution itself, i.e., the extractor tailored to a specific task instance, is built with a method which does not use string metrics in any way.

The availability of an estimate of costly data elaborations may be desirable also when dealing with data quality. For instance, the authors of [30] propose a method for estimating the number of duplicates in a dataset, before actually applying more

[1] A web based version is available on http://regex.inginf.units.it/; the source code is published on https://github.com/MaLeLabTs/RegexGenerator.