Exhibit D

Improving Recall of Regular Expressions for Information Extraction

Karin Murthy, Deepak P., and Prasad M. Deshpande

IBM Research - India, Bangalore, India
{karin.murthy,deepak.s.p,prasdesh}@in.ibm.com

Abstract. Learning or writing regular expressions to identify instances of a specific concept within text documents with a high precision and recall is challenging. It is relatively easy to improve the precision of an initial regular expression by identifying false positives covered and tweaking the expression to avoid the false positives. However, modifying the expression to improve recall is difficult since false negatives can only be identified by manually analyzing all documents, in the absence of any tools to identify the missing instances. We focus on partially automating the discovery of missing instances by soliciting minimal user feedback. We present a technique to identify good generalizations of a regular expression that have improved recall while retaining high precision. We empirically demonstrate the effectiveness of the proposed technique as compared to existing methods and show results for a variety of tasks such as identification of dates, phone numbers, product names, and course numbers on real world datasets.

1 Introduction

Unstructured data such as emails, blogs, web pages, and chat conversations increasingly form an integral source of information for data analytics. An important step to make unstructured data available for analytics is Information Extraction (IE), which extracts structured information (that is, concepts such as phone numbers, email addresses, person names, and SSNs) from text.

Learning-based IE (for example, [15] and [13]) relies on humans to mark enough examples of the concept that has to be extracted; the labelings are then used to build a mathematical model for IE. Rule-base IE (for example, CPSL [1] and SystemT [4]) in contrast relies on humans to construct appropriate machine-understandable rules to extract specific concepts from a text. Regular expressions (regexes) are the dominant building block for a large class of extraction tasks in rule-based IE. To build effective information extraction engines, regexes that cover almost all instances of a specific concept (high recall) and have few false positives (high precision) are required.

Data-driven fine-tuning of regexes to render them more accurate for information extraction tasks is a highly challenging task. An intuitive method of modifying a regex is to specialize or generalize it. Specializing a regex involves restricting its scope so that the modified regex matches only a subset of earlier matches; an example is to modify $\backslash w\{3\}$, the regex that matches any sequence of three alpha-numeric characters to $\backslash d\{3\}$, a regex that matches any sequence of three numeric characters. An example of generalization is the reverse modification.

X.S. Wang et al. (Eds.): WISE 2012, LNCS 7651, pp. 455–467, 2012.
© Springer-Verlag Berlin Heidelberg 2012

If all the matches of a regex are labeled as true matches (those that indeed denote the concept of interest) or false matches, the regex itself can be specialized [11] to cover as many of the true matches as possible, while excluding as many of the false matches as possible. However, the analogous task of generalizing a regex to render it more accurate for IE, has not been well addressed in literature. Generalizing a regex is much harder since the scope of labeling is not limited to instances covered by the existing regex. Instead, to identify false negatives additional labeling of the text corpus is necessary.

Consider the regex, *1-\d{3}-\d{3}-\d{4}*, intended to extract phone numbers. It is possible that the intent was to also extract phone numbers of the form *1-800-COMPANY* (sometimes used for advertisements). However, no labeled instances of this kind were given to the regex learner or the regex author was unaware of such phone numbers. In such a case, it would be beneficial to learn that a slightly generalized expression such as *1-\d{3}-\w{3}-{0,1}\w{4}* covers such phone numbers. In the new regex the last two groups of digits (*\d*) were generalized to also allow letters (*\w*) and the second '-' was made optional.

We focus on regex generalization as a way to improve the recall of a regex and exploit the fact that in general humans can easily decide whether newly covered instances are true or false. In particular, we present a technique to incrementally navigate the exponential space of possible generalizations of an initial regex and identify good quality generalizations while soliciting minimal user feedback on correctness of newly discovered matches.

2 Related Work

Learning a regular expression from scratch using a list of true matches has been explored well [5–7]. It has also been found to have applications in learning other patterns like DTDs from XML data [3, 8].

Data-driven modification of a seed pattern to arrive at more accurate patterns has also been subject of some research. Outside the realm of regexes, an early work [12] proposes techniques to generalize a pattern in a data-driven manner; this, however, assumes the availability of a comprehensive set of true and false matches which is harnessed by the search procedure to fine tune the generalization. In particular, the information whether any matching instance is a true match or not, is deemed to be available beforehand. [2] proposes a clustering-based method for a similar generalization task for regexes under such an all-labels-available-beforehand assumption. Our problem addresses the more realistic scenario where only a few labels, those of instances that match the seed regex, are available. Adapting the techniques in [2, 12] to solicit labels on the fly would demand significant manual labeling effort, whereas the manual labeling effort is typically the point of optimization. We show in Section 5.5 that the technique we propose incurs up to orders of magnitude less labeling effort than methods that rely on all-labels-available-beforehand assumptions

If the seed regex is already generic enough, the technique proposed in [11] may be employed to find more accurate specializations of the regex. It works by using a labeling of all instances covered by the seed regex, to arrive at specializations that cover as many of the true matches and as few of the false matches as possible. Our work is complementary in that our goal is to discover generalizations.