Towards a Method for Unsupervised Web Information Extraction*

Hassan A. Sleiman and Rafael Corchuelo

Universidad de Sevilla, ETSI Informática,
Avda. Reina Mercedes, s/n, Sevilla E-41012
{hassansleiman,corchu}@us.es

Abstract. The literature provides a variety of techniques to build the information extractors on which some data integration systems rely. Information extraction techniques are usually based on extraction rules that require maintenance and adaptation if web sources change. We present our preliminary steps towards an unsupervised information extraction technique that searches web documents for shared patterns and fragments them until finding the relevant information that should be extracted. Experimental results on 1230 real-web documents demonstrate that our system performs fast and achieves promising results.

Keywords: Web Information Extraction, Unsupervised Technique.

1 Introduction

The Web is a huge and still growing information repository. Web information is usually embedded into HTML tags and buried in other contents that are not relevant for a particular purpose. Business processes that require structured information, need to extract and structure the information they require from HTML documents. Information extractors are usually used for this purpose and can be broadly classified into two types: Those that work on free text, including blogs and news documents [1], and those that work on semi-structured documents such as search results and web documents with detailed information about some items [2]. Our work fits within the second category.

Information extractors are usually based on rules. These rules can be handcrafted, learnt using semi-supervised techniques that require the user to provide some annotated training documents [3,4], or unsupervised techniques that learn extraction rules for all the information they consider as relevant inside some training documents [5,6]. Rule-based information extractors need to be maintained or even rewritten if the web source on which they were trained changes [7].

* This work was supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E).

This has motivated researchers to work on a new group of unsupervised information extractors that are not based on extraction rules [8,9], but on a number of hypothesis that have proven to perform well on many web sources.

In this paper, we report on our preliminary ideas on an unsupervised information extractor based on the hypothesis that web documents, generated by the same server-side template, share string patterns that are irrelevant.

2 System Overview

Our proposal takes two or more web documents, and searches for shared patterns amongst them of size $s = max$ down to $s = min$, where $max \geq min \geq 1$. When a shared pattern *sp* is found, the text of each document is partitioned to create 3 groups: prefixes, suffixes, and separators. Prefixes contain the text fragment from the beginning of each text until the start of the first occurrence of *sp* in this text; suffixes contain the text fragment from the end of the last occurrence of *sp* in each text to the end of this text, and separators include each separating text between every two consecutive occurrences of *sp* inside each text.

Now that we have created three groups of text, the algorithm tries to search for a shared pattern of the the same size s between the components of each group. If a group shares a string pattern, it is partitioned again; if not, s is

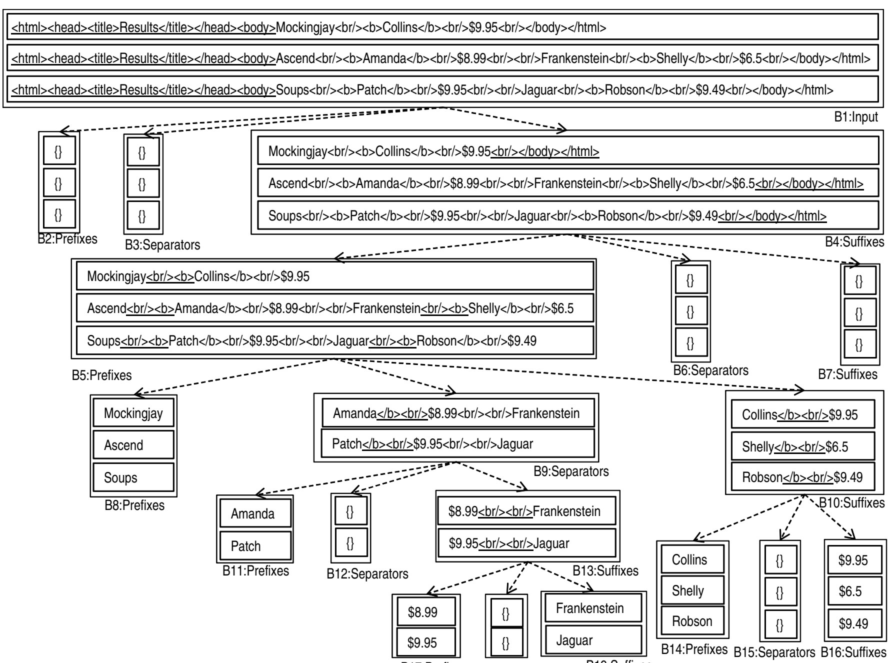

Fig. 1. An example of how our proposal works

decreased, as long as $s \geq min$, and the algorithm starts again its shared pattern search on this group. When $s = min$ and no shared patterns are found, the proposal considers that the remaining non-empty text fragments inside each group can be considered as relevant text that should be extracted. The search for shared patterns is performed using a modified version of Knuth-Morris-Pratt's algorithm [10] in which all the occurrences of a string sequence are detected without overlapping.

Figure 1 illustrates an example on how our proposal works. Strings are tokenised in a scheme of two types HTML tags or #PCDATA. The proposal takes the first block $B1$ that contains three sample web documents, $max = 10$, and $min = 1$ as input. It searches for a shared pattern of $size = 10$ tokens between the three documents in $B1$. Since none is found, the algorithm continues decreasing $size$ to 9, then 8, until it finds a shared pattern of $size = 7$ tokens ($< html >< head >< title > Results < /title >< /head >< body >$) between the three strings in $B1$. Then, it creates prefixes $B2$, suffixes $B4$ and separators $B3$. $B2$ and $B3$ are discarded since they are empty. The algorithm now searches for patterns of $size = 7$ inside $B4$, but since no shared pattern of the given size is found in $B4$, $size$ now changes to $6, 5, 4, 3$. It finds a pattern of $size = 3$ in $B4$ ($< br/ >< /body >< /html >$), partitions it into the prefixes $B5$, suffixes $B7$ and separators $B6$. It searches for shared patterns of the same size in the $B7$. Since the strings in $B7$ do not contain a shared pattern of $size = 3$, $size$ is decreased and the algorithm finds the shared pattern of $size = 2$ ($< br/ >< b >$) between the strings in $B7$. It partitions $B7$ and creates the prefixes $B8$, suffixes $B10$ and separators $B9$. Since strings inside $B8$ do not share a pattern of $size \in [2, min]$, then $B8$ is added to the output. It now repeats the previous steps on $B9$ and $B10$ until finding blocks whose strings do not share any pattern, which are added to the output. The output of this example is a list of blocks that contain $B8$, $B11$, $B17$, $B19$, $B14$, and $B16$. Empty blocks like $B12$ and $B15$ are discarded. According to our experience, max and min can be automatically determined by considering max as 5% the size of the smallest input document, and min as 1.

3 Experimental Results

We implemented a prototype and tested it on a collection of 41 datasets from different web sites. These web sites belong to the following categories: books, cars, conferences, doctors, jobs, movies, real estates, and sports. These categories were randomly sampled from The Open Directory sub-categories, and the web sites inside each category were randomly selected from the best ranked web sites between December 2010 and March 2011 according to Google's search engine. We annotated in each dataset the relevant information and then each string item extracted by our proposal was considered as a true positive (tp), false negative (fn), or false positive (fn). We are interested in measuring precision $P = \frac{tp}{tp+fp}$, recall $R = \frac{tp}{tp+fn}$ and the extraction time of our proposal.

Table 1. Comparison between our proposal, RoadRunner, and FiVaTech

	Precision	Recall	Time (seconds)
RoadRunner [5]	0.312	0.323	0.014
FiVaTech [6]	0.800	0.904	0.348
Our proposal	0.958	0.980	0.0310

We used our collection of datasets to compare our proposal to RoadRunner [5] and to FiVaTech [6], cf. Table 1. Note that our proposal achieves a better recall and precision than both techniques. Although the extraction time archived by our proposal is higher than that one archived by RoadRunner, they both are very close to 0 and the difference between them is insignificant.

4 Conclusions

We have presented an abstract of our preliminary steps towards a totally unsupervised web information extraction technique. It builds on a simple heuristic that has proven to work well in many real-world web documents since it can achieve high precision and recall while requiring very little time. In future, we plan on studying its complexity, comparing it to other well-known techniques in the literature, to create extraction rules that can be reused, and to label the information extracted semantically.

References

1. Turmo, J., Ageno, A., Català, N.: Adaptive information extraction. ACM Comput. Surv. 38(2) (2006)
2. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of web information extraction systems. IEEE Trans. Knowl. Data Eng. 18(10), 1411–1428 (2006)
3. Kushmerick, N., Weld, D.S., Doorenbos, R.B.: Wrapper induction for information extraction. IJCAI (1), 729–737 (1997)
4. Hsu, C.N., Dung, M.T.: Generating finite-state transducers for semi-structured data extraction from the Web. Inf. Syst. 23(8), 521–538 (1998)
5. Crescenzi, V., Mecca, G., Merialdo, P.: RoadRunner: Towards automatic data extraction from large web sites. In: VLDB, pp. 109–118 (2001)
6. Kayed, M., Chang, C.H.: FiVaTech: Page-level web data extraction from template pages. IEEE Trans. Knowl. Data Eng. 22(2), 249–263 (2010)
7. Chidlovskii, B., Roustant, B., Brette, M.: Documentum ECI self-repairing wrappers: performance analysis. In: SIGMOD Conference, pp. 708–717 (2006)
8. Álvarez, M., Pan, A., Raposo, J., Bellas, F., Cacheda, F.: Extracting lists of data records from semi-structured web pages. Data Knowl. Eng. 64(2), 491–509 (2008)
9. Elmeleegy, H., Madhavan, J., Halevy, A.Y.: Harvesting relational tables from lists on the Web. PVLDB 2(1), 1078–1089 (2009)
10. Knuth, D.E., Morris Jr., J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. 6(2), 323–350 (1977)