Automatic Filling of Web Forms

Gustavo Zanini Kantorski and Carlos Alberto Heuser

Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
{gzkantorski,heuser}@inf.ufrgs.br

Abstract. Since the only way to gain access to Hidden Web data is through form submission, one of the challenges is how to fill Web forms automatically. In this work, we describe an efficient method to select good values for fields and propose a new approach to minimize the number of queries that must be generated for the automatic filling of Web forms.

Keywords: Hidden Web, Deep Web, Filling Forms, Crawling

1 Introduction

The surface Web is the portion of the World Wide Web that can be reached by direct link navigation. However, a vast portion of the information on the Web is available in online databases and can only be reached through Web form filling and submission. This portion of the Web is known as *Hidden Web* [11] or *Deep Web* [3] and it is not indexed by conventional search engines. In this context, one of the challenges is how to automatically fill forms in order to gain access to the data. This task is not trivial, since forms were designed to be used by human beings. The simplest solution would be submitting the combination of all possible field values in a cartesian product. However, this solution is not feasible when the number of fields and possible values are large.

The state of art deep web crawling approaches has several solutions for automatic form filling. In Liddle *et al.* [6], automatic form filling is carried out by assigning default values to form fields. Text fields are ignored and, if they are mandatory, user intervention is needed. Barbosa *et al.* [4] present an approach for filling forms based on keywords. The discovery of words is based on the data coming from the database itself, instead of random word generation. On the other hand, they do not handle Web forms that do not contain keyword fields. Wu *et al* [9] present a form filling technique based on a feedback process of the values filled in the form. The issue with this method is that, in the query, just one form field can be used at a time. It is not possible to combine form fields for several queries. Jian *et al.* [1] present a formal framework based on reinforcement learning. Toda *et al* [8] describe a solution for form filling based on value extraction from a text document. The approach relies on the knowledge obtained from the values of previous submissions for each field. The work by Madhavan *et al.* [2] describes a system for surfacing the content of the Hidden Web. The goal is to index the HTML pages resulting from form submissions. The authors introduce the concept of template for Web forms. A template designates

a subset of the inputs, called *binding inputs*, and the remainder are considered *free inputs*. Multiple form submissions can be generated by assigning different values to the binding inputs. The number of fields (*binding inputs*) that make up a template will be referred to as the *dimension of the template*.They improved the keyword selection algorithm by ranking keywords by their TFIDF (Term Frequency Inverse Document Frequency).

In this work, we propose an automatic method for filling forms. Our method explores two strategies. The first, called FTF (*Filling Text Fields*), is how to fill the fields efficiently, specially the text fields, which do not have a set of predetermined values. The second strategy, called ITP (*Instance Template Pruning*), is how to select *queries* to submit to a particular form in order to retrieve more data with fewer submissions. The strategy to minimize the set of queries, *i.e.,* the number of form submissions, involves pruning the set of all possible queries. As each query is submitted, data extracted from the resulting page is used to identify wasteful queries and prune them.

Most of the existing work on form filling [1,4,6–8,10] overlooks the problem of finding good values for these fields. Existing solutions for dealing with text fields usually rely on a list of values previously built by a specialist [7], on a sample of known values [2], or they entail the extraction and understanding of the fields [1, 5, 8] (which depends on the language and on the domain of the forms). Our proposed solution is automatic, requires no training, and relies on feedback from previous submissions. Furthermore, it can work with forms from any domain (*i.e.,* books, jobs, hotels, airfares, etc.). Domain often influences value selection for the fields. Although our method does not use domain knowledge explicitly, our experiments show that the values generated for the fields are domain-related. We carried out experiments using real Hidden Web data. The results show that in most cases, our approach achieves superior coverage compared to a baseline method.

2 Solution Overview and On-going Work

The problem we address in this work is how to choose values for filling form fields in a way to maximize the number of distinct database rows retrieved while minimizing the number of form submissions. Our solution is divided into a series of steps organized in an architecture. Similar to the work by Madhavan *et al.* [2], we rely on the concept of templates. Figure 1 shows the proposed architecture in which our main contributions are in the highlighted modules *Value Selection* and *Instance Template Generation and Submission*. The *Candidate template generation* module represents the HTML form processing to generate all possible templates. The *Value Selection* module finds the values to fill each field in the form. Here the main problem is how to choose values for fields with infinite domains. FTF is based on a feedback loop, in which each element has an effect on the next one, until the last element produces feedback on the first element. The selection of values for text boxes is done by assigning a score r which aims to select meaningful tokens from previous submissions. The *instance*

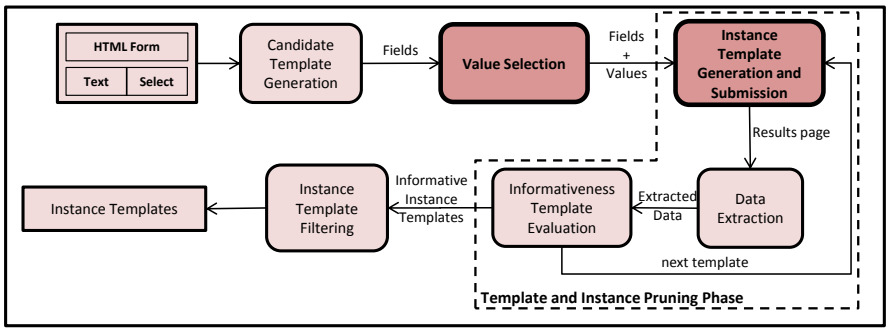

Fig. 1. Proposed Architecture

template generation and submission module attaches input values to each field of each template, creating its set of instance templates. The templates are processed in order of dimension. First all 1-dimension instance templates are generated and submitted, then, the results of these submissions are used to generate 2-dimension instance templates and so on. We present the *Instance Template Pruning* (ITP) method, used to prune the wasteful instances of a template. The *Data extraction* module extracts the information from the pages resulting from form submissions. This extraction is needed to find where the data region is located in the result page. Information about ads and presentation from the result page are discarded. The extracted data is used to evaluate each template. This data is also used to populate a database, which is used to generate values for fields with infinite domains (*Value Selection* module). The intuition is that higher coverage may be achieved if instead of using randomly select data, values resulting from previous submissions are used. The *informativeness evaluation* module checks if the template is informative after the submission of the selected instance templates. A template is *informative* if its instance templates retrieve sufficiently distinct data. If a template t is considered non-informative, higher order templates including t will be discarded. At the end of the process, after processing all templates, the *Filtering Instance Templates* module determines the minimum number of instance templates needed to retrieve all distinct rows extracted from the pages resulting from form submissions.

For our purposes, we divide the Web forms in two types: those that contain only text boxes and those that contain, besides text boxes, finite domain fields, such as selection lists. The division is necessary, because depending on the type of form, the generation of initial values for the fields is different. The *select* fields are filled by the values extracted from the code of the form, in the *option* tag from HTML form. Queries are generated by values extracted from option tag and then submitted. The query results are stored in a database. The order of submission of queries always starts by finite domain fields followed by infinite domain fields. The database containing query results will be used later to generate values that are used to fill infinite domain fields. The information inside the HTML page

which contains the form is extracted only when the form has just infinite domain. These information are *seed* values for text fields.

Several experiments were performed in order to evaluate the proposed strategies for input value selection and instance template pruning. All experiments were carried out using real Web forms. The proposed strategies and the architecture are domain independent. Forms from several domains (such as, Jobs, Books, Movies and Food), and sizes were used in experiments. Table 1 shows details about the forms used in experiments. The number of web forms used in our experiments is similar to what is used in related work ([4,6–8]). Our baseline is an implementation based on the method proposed by Madhavan *et al.* [2]

Three metrics were used in the evaluation. The coverage [4] C_f, of a form f is the number of distinct records extracted during the whole process, i.e., the total information obtained by the informative templates. The execution efficiency [10], EE_f, for a form f evaluates the coverage C_f compared to the cost of the method, that is, the ratio between coverage and the number of URLs submitted ($Total_{submitted}$) in the process. The indexing efficiency, IE_f, for a form f evaluates the relationship between coverage C_f and the cost of indexing of each method, i.e., the average number of records obtained for each distinct URL indexed ($Total_{indexed}$).

In all forms tested, the coverage for the ITP method had the best performance, because it makes further exploration of the submission possibilities, since it uses all templates. For the FTF method, the number of URLs indexed by our method was higher in all forms. This is true because the generated values are of better quality, that is, they retrieve more rows. Thus, the likelihood of a template be informative is higher for our method than for the baseline. Another observation is that, for forms that contain only text fields, the templates with dimension larger than one have a small probability of being informative. Our experiments showed that all dimension-1 templates are informative, increasing the chance of templates with higher dimension being also informative. FTF method reached a higher coverage, compared with the baseline. The overall average coverage from baseline was 47,8%, while average by FTF was 81,8%. This shows the efficiency of our method. Our method, FTF, was always better for forms that contain only text box fields.

Id	Web Form	# fields text	# fields select	Id	Web Form	#Fields text	#Fields select
Forms for FTF Method				Forms for ITP/ITTP Methods			
1	http://www.beerintheevening.com/pubs/	3	1	1	http://www.foodandwine.com/search/	3	1
2	http://www.foodandwine.com/search/	3	1	2	http://www.global-standard.org/	1	3
3	http://www.mymusic.com/advancedsearch.asp	6	2	3	http://onlineraceresults.com/search/index.php	2	0
4	http://www.posteritati.com/advanced_search.php	1	1	4	http://www.phillyfunguide.com/	1	2
5	http://www.e4s.co.uk/	1	2	5	http://www.whoprofits.org/	2	1
6	http://www.whoprofits.org/	2	1	6	http://www.hcareers.com/seeker/search/	1	5
7	http://www.mldb.org/search-bf	4	0	7	http://www.rtbookreviews.com/rt-search/books	1	2
8	http://usajobs.opm.gov/	2	0	8	http://formovies.com/search/combined.html	3	0
9	http://www.movlic.com/k12/search.asp	2	0	9	http://www.careerbuilder.com/	2	1
10	http://jobs.careerbuilder.com/	2	0	10	http://www.policechiefmagazine.org/magazine/	1	2

Table 1. Form properties

3 Conclusions and Future Work

In this work, we describe two methods for improving the search on the Hidden Web: *i.* a method to minimize the number of queries submitted on the form and *ii.* an automatic method for selecting values for text fields. The ITP strategy reduces the number of submissions that not return new distinct data. The FTF method is totally automatic and can be used in any Web form that has text fields. Although FTF method is domain-independent, the selected values for text box fields adapt for domain. Our experiments demonstrate that our approaches are able to properly deal with text fields and, query selection and have verified to be useful as an alternative to automatically filling Web forms. For future work, we will carry on with the research into two phases. In the first phase, we will use a statistical model to combine field values in templates with order higher than one. The second phase will entail the definition of a new method to determine values for text fields considering the number of distinct rows and the order of instance template submissions to assess the quality of the selected values.

Acknowledgments. This research was partially supported by the National Counsel of Technological and Scientific Development, CNPq, project number 480283/2010-9.

References

1. Jiang, L. and Wu, Z. and Zheng, Q. and Liu, J.: Learning Deep Web Crawling with Diverse Features. WI/IAT, (2009) 572–575
2. Madhavan, J. and Ko, D. and Kot, Ł. and Ganapathy, V. and Rasmussen, A. and Halevy, A.: Google's Deep Web Crawl. Proc. of the VLDB Endowment. VLDB Endowment,(2008), Vol. 1,Number 2,1241–1252.
3. Bergman, M.K.: The Deep Web: Surfacing hidden value. Journal of Electronic Publishing. (2001), Vol. 7, Number 1, 07–01.
4. Barbosa, L. and Freire, J.: Siphoning Hidden-Web Data through keyword-based interfaces. SBBD, 309–321, (2004).
5. Khare, R. and An, Y. and Song, I.Y.: Understanding deep web search interfaces: A survey. SIGMOD Rec., ACM, 39(1): 33–40. (2010).
6. Liddle, S. and Embley, D. and Scott, D. and Yau, S.: Extracting data behind web forms. Advanced Conceptual Modeling Techniques. Springer, 402–413, (2003).
7. Raghavan, S. and Garcia-Molina, H.: Crawling the Hidden Web. VLDB, 129–138, (2001).
8. Toda, G.A. and Cortez, E. and da Silva, A.S. and de Moura, E.: A probabilistic approach for automatically filling form-based web interfaces. Proc. of the VLDB Endowment, 4(3): 151–160. (2010).
9. Wu, P. and Wen, J.R. and Liu, H. and Ma, W.Y.: Query selection techniques for efficient crawling of structured web sources. ICDE, IEEE, 47–47. (2006)
10. Ntoulas, A. and Zerfos, P. and Cho, J.: Downloading textual hidden web content through keyword queries. JCDL, 100-109. (2005).
11. Florescu, Daniela and Levy, Alon and Mendelzon, Alberto.: Database techniques for the World-Wide Web: a survey. SIGMOD Rec. ACM. 27(3): 59-74. Sep. (1998).