Optimizing Web Extraction Queries For Robustness

Mantas Kanaporis

Kongens Lyngby 2013
IMM-M.Sc.-2013-????

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-M.Sc.-2013-????

Summary

The goal of the thesis is to ...

Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Informatics.

The thesis deals with ...

The thesis consists of ...

Lyngby, 27-January-2013

Mantas Kanaporis

Contents

Chapter 1

Introduction

1.1 Background

The world wide web contains a vast amount of information, counting in billions of web pages [WorldWideWebSize.com]. Mostly it is unstructured or semi-structured documents, making it difficult to process the data directly by machines. As a result, sharing and reusing knowledge across applications, enterprises, and communities is hardly possible without human interaction.

Addressing the need to automatically extract and process large volumes of accessible information, machine needs to extract loosely-structured data from web sources and populate databases with well-structured data for further handling. The problem can be seen as information extraction problem for web pages.

There are many use cases for sharing data between web-based applications with no dedicated integration capabilities. Some examples include extracting product price information from competitor on-line shops, or integrating with an online scheduling application which has no application programming interface (API). In both cases data is presented to the user via the web browser and no special care was taken to make it easy for automatic extraction.

1.2 Problem

Most of the pages on the web are HTML documents. Many websites use view templates to generate individual pages on the server side. Thus, the pages generated from the same template have similar stucture. HTML has a tree structure and can be viewed as an labeled ordered rooted tree. To extract information from the webpage, one needs to query a node in a tree. The program that performs the actual information extraction (IE) task from webpages is called a web wrapper. The term originates from information system integration domain [CKGS06], where a proxy interface abstracts away the complexity of accessing a data source. The web wrapper can be expressed in a number of standartized notations, e.g., XPath, CSS selectors, etc.

For a template based webpage, writing a query that extracts certain pieces of information is a straight forward task. The query can be written in many ways and still target the same node in DOM tree. The problem gets really interesting when you take into consideration that websites are maintained and evolve over time. A slight user interface update might break the wrapper.

Building the most robust wrapper from all possible wrappers is the subject of this thesis. Defining wrapper robustness is a problem of its own. Some format definitions will be shown later. Informally, it's the wrapper that has the lowest odds of breaking after any changes to the web page structure.

The problem of building a robust wrapper is relevant in many areas, inluding web application integration, web user interface test automation, and web scrapping.

1.3 Goal

The goal of this thesis is to design an algorithm for building the most robust wrapper from a single HTML snapshot. Futher more, the algorithm is constrained with the following limitations and assumptions:

1. There is a single base version of the web page, i.e. no training set.
2. The wrapper must be robust and return the distinguished node(s) on the future web page versions.
3. The wrapper must be reasonably fast (average execution time < 0.5 sec).

Inputs:

1. Base version of HTML document ω_{base}
2. Location of the distinguished node in the base document $d(\omega_{base})$
3. New version of HTML document ω_{new}

Outputs:

1. New location of the node $d(\omega_{new})$
2. Confidence measure c

1.4 Contribution

What I managed to do, though, is to narrow down my focus to the tool-for-the-job – a tree edit-distance application. There are three recent papers that build on this idea [DBS09], [PDGMR11], [LWYL12].

While the idea of robust wrappering is not new, the core difference from earlier systems is the unique combination of ideas and tools that are optimized for performance. Here the idea of computing edit-distance between two trees (or DOMs) is introduced, robustness is defined formally, and later dynamic algorithm for finding the optimal wrapper is designed. My contribution was to implement the algorithm, optimize the performance using the latest research, and improve the algorithm to support multiple result sets (as opposed to single node from the query).

1.5 Organization

The concepts are defined in chapter 2. Next current state of the art is discussed and our focus is narrowed. Next we elaborate on the design and implementation of our method. Afterwards we present the empirical experiment results. The thesis is conluded with a finding discussion and future research direction.

Chapter 2

Main concepts

2.1 Labeled ordered rooted tree

2.2 Wrapper

2.3 Wrapper robustness

2.4 Edit tree distance

Chapter 3

Related research

3.1 Web data extraction methods

3.1.1 Broad view

[LRNdST02] Recently, many tools have been proposed to better addresss the issue of generating wrappers for Web data extraction. Such tools are based on several distinct techniques such as decrlarative languages, HTML structure analysis, natural language processing, machine learning, data modeling, and ontologies. The paper introduces a taxonomy.

The problem of data extraction from web pages has been addressed in a number of papers. [LRNdST02] reviews multiple techniques for generating wrappers with various techniques, inluding natural language processing, languages and grammar, machine learning, information retrieval, databases, and ontologies. A common goal of all wrapper generation tools is to build a wrapper that is accurate and robust, but is built with the least possible human interaction. The article provides a taxonomy for wrapper inducityon techniques.

In a recent overview of web data extraction techniques [CKGS06] the author argues, that due to template generated content, web IE problems can take advan-

tage machine learning and pattern matching methods. Compare it to traditional IE approaches that are mostly based on natural language processing (NLP). - Unsupervised approaches can only support template pages. The extension of such systems to non-template page extraction tasks is very limited.

3.1.2 Extracting wrapper from a single page

One the the first papers to discuss XPath wrappers was [MJ02]. Author defines two metrics for measuring robustness. (1) a number of times that expression failed to extract correct results. (2) expression complexity in terms of depth. The paper defines a notion of anchors (DOM tree nodes) and hops (relative XPath expressions) over a normalized HTML document. The wrapper is a set of XSLT extraction rules, i.e. hops that are based on structure, attributes, or content. Yet the wrappers were created all by hand.

[KOKA06] empirically confirmed that there is a notable difference between various versions of wrappers in terms of robustness – relative XPath expressions outperform absolute ones significantly. This proves the idea that some wrappers are more robust.

[GMM+11] introduces apriori style algorithm for learning xpath-based extraction rules that are robust to variations in site structure. Algorithm learns rules from human annotated pages based on structural features. Based on domain knowledge, these features are classified into strong (e.g. HTML attributes `class` or `id`, tags, textual fragments) and weak (e.g. `font`, `width` attributes). This method tries to build a path from strong features only by generating and combining candidates. If Apriori fails to output precise XPath, Naive-Bayes based classifier is used to evaluate the best guess of newly generated nodes with weak features. Essentially this method is based on enumeration, which makes it slow for large websites. This has been shown by our preliminary empiric implementations. Additional limitation is that it requires a set of annotated pages for certain heursitics (e.g. support metrics) to properly work.

[TEBS12]

- http://www.cs.uic.edu/ liub/WebMiningBook.html

3.2 Minimum Cost Edit Model

elaborate on the work so far:
- Zhang, Shasha
- A Survey on Tree Edit Distance and Related Problems
- RTED - A Robust Algorithm for the Tree Edit Distance
- Simple fast algorithms for the editing distance between trees and related problems

3.3 Defining robustness

- Optimal Schemes for Robust Web Extraction
- Robust Web Content Extraction
- Robust Web Extraction - An Approach Based on a Probabilistic Tree-Edit Model
- Robust Web Extraction Based on Minimum Cost Script Edit Model
- Robust Web Data Extraction - A Novel Approach Based on Minimum Cost Script Edit Model

Chapter 4

Robust web extraction framework

HTML has a tree structure and can be viewed as an ordered labled tree.

Define the content more carefully: all sections and a brief description what you will write in each of them. Define the main concepts you will need and fix the notations. Then you can write the chapters in any order you want. Make also a work plan: what you will do and when.

Specify your topic carefully. Don't take too large topic! Invent a preliminary title for your thesis and define the content in a coarse level (main chapters). Ask your supervisor's approval! Decide with your supervisor what material you should read or what experiments to make.

You can write the thesis after you have read all material or made all experiments. However, you can begin to write some parts already when you are working. Often you have to change your design plan, but it is just life! Ask feedback from your supervisor, when your work proceeds.

Chapter 5

Generating minimal candidate wrappers

Chapter 6

Learning edit costs

Chapter 7

Building robust wrappers

Chapter 8

Optimizing for performance with parallelization

Chapter 9

Experiment evaluation

This section should write itself. Since you have a goal, the results section must document that you have reached this goal (or verified your hypothesis).

Chapter 10

Conclusion

You can conclude that you have reached your goal or discuss why not if that is the case. Also discuss hindsights: What things were even better or a little worse than expected regarding the methods you used to solve your problems. How could your project be improved by further work.

10.1 Summary

10.2 Conclusion

10.3 Future research

Appendix A

Stuff

This appendix is full of stuff ...

Bibliography

[CKGS06] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled F. Shaalan. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 18(10):1411–1428, October 2006.

[DBS09] Nilesh N. Dalvi, Philip Bohannon, and Fei Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *SIGMOD Conference*, pages 335–348, 2009.

[GMM+11] Pankaj Gulhane, Amit Madaan, Rupesh R. Mehta, Jeyashankher Ramamirtham, Rajeev Rastogi, Sandeepkumar Satpal, Srinivasan H. Sengamedu, Ashwin Tengli, and Charu Tiwari. Web-scale information extraction with vertex. In *ICDE*, pages 1209–1220, 2011.

[KOKA06] Marek Kowalkiewicz, Maria E. Orlowska, Tomasz Kaczmarek, and Witold Abramowicz. Robust web content extraction. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 887–888, New York, NY, USA, 2006. ACM.

[LRNdST02] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Rec.*, 31(2):84–93, June 2002.

[LWYL12] Donglan Liu, Xinjun Wang, Zhongmin Yan, and Qiuyan Li. Robust web data extraction: A novel approach based on minimum cost script edit model. In *WISM*, pages 497–509, 2012.

[MJ02] Jussi Myllymaki and Jared Jackson. Robust web data extraction with xml path expressions. *IBM Research Report*, 2002.

[PDGMR11] Aditya G. Parameswaran, Nilesh N. Dalvi, Hector Garcia-Molina, and Rajeev Rastogi. Optimal schemes for robust web extraction. *PVLDB*, 4(11):980–991, 2011.

[TEBS12] Jakob G. Thomsen, Erik Ernst, Claus Brabrand, and Michael Schwartzbach. Webself: a web scraping framework. In *Proceedings of the 12th international conference on Web Engineering*, ICWE'12, pages 347–361, Berlin, Heidelberg, 2012. Springer-Verlag.