Content Based Cross-Site Mining Web Data Records

Jebeh Kawah, Faisal Razzaq, Enzhou Wang
Mentor: Shui-Lung Chuang
CS 511 – Spring 2005
Project #7 Data Record Extraction

1. Introduction

Current web data record extraction methods [1, 2, 3] are mostly based on HTML syntax analysis. Algorithms and heuristics are developed to identify HTML tag patterns. The boundary of a data record is determined by tag patterns. Such an approach makes minimal assumptions about the web pages to be processed and is applicable in a wide range of data extracting tasks. However, in many web data record extraction scenarios, we have more information we can utilize than just HTML syntax structure. For example, when we search a keyword at www.amazon.com, we can be sure that the returned data records will contain the keyword. When we aggregate information from multiple sites, we will search the same keyword on multiple sites. The correlation between the search results from multiple sites may be able to help identify data records. This is because the repetition of the data records among different sites, (or even the repetition of the same data records on the same site) will establish a pattern that can be used to obtain information about other similar data records. Based on these observations, we propose a new algorithm CCM (Content-Based Cross-Site Mining Web Data Records) that takes advantage of cross-site reference information to tackle data record extraction problem.

We believe combining existing techniques of extracting data records based on the structure of documents (HTML tags) with an analysis of the semantics of the content will yield better data record extraction and provide more meaningful results.

2. Overview of CCM Algorithm

When we search certain keyword at a single site, we know for sure that the result data records must contain the search keyword, but we are not so certain about whether a string that contains the search keyword is part of a data record. However, if we search the same keywords at multiple sites and a certain string containing the search keyword appears in search results of several sites, it is very likely that string is part of a data record. CCM is based on such an observation that there can be data records that are easily identifiable when doing cross-site keyword search. We will identify data region and record boundary based on those records and conduct further extraction.

Assume we search a keyword on multiple sites. A keyword can be a word, a phase, or some combination. We will get one or more result page from each site. The input of this algorithm is a list of pages grouped by sites. The algorithm we propose has four major steps:

(1) Key phrase extraction

By key phrase, we mean a content phrase in HTML that contains the keyword we search for. For example, when we search "Java" on www.amazon.com, we may get a simplified result page like:

```
<html><body><table>
<tr><td>Java 2: A Beginner's Guide</td></tr>
<tr><td>Head First Java</td></tr>
<tr><td>Core Java</td></tr>
</table></body></html>
```

The key phrases will be "Java 2: A Beginner's Guide", "Core Java", and "Head First Java." We will extract a key phrase list for each site we searched. For example, by searching "Java" on www.bn.com, we get

```
<html><body><table>
<tr><td>Java (SparkCharts)</td></tr>
<tr><td>Java in Easy Steps</td></tr>
<tr><td>Head First Java</td></tr>
</table></body></html>
```

The key phrase list is "Java (SparkCharts)", "Java in Easy Steps", and "Head First Java." From www.bookpool.com, we can get

```
<html><body><table>
<tr><td> JavaScript: The Definitive Guide</td></tr>
<tr><td>Core Java</td></tr>
<tr><td>Head First Java</td></tr>
</table></body></html>
```

The key phrase list will be "JavaScript: The Definitive Guide", "Core Java", and "Head First Java."

(2) Ranked key phase list generation

Once we have key phrase list for each site, we will combine the lists to generate a ranked key phrase list. A key phrase will be ranked by how many times it appears in individual key phrase list. In our three-site search example, the ranked key phrase list is

Key phrase	**Ranking**
Head First Java	3
Core Java	2
Java 2: A Beginner's Guide	1
Java (SparkCharts)	1

Java in Easy Steps	1
JavaScript: The Definitive Guide	1

Table 1. Ranked Key Phrase List

(3) Record boundary identification by growing "seed."

We will start processing individual pages starting from the page that has the highest ranking score: the summation of all key-phrase rankings of the page. We can randomly pick one if there is a tie. In our example, we will start from the page from www.amazon.com.

We call the highest ranked key phrase "seed." We can specify a threshold to choose multiple seeds. To simplify our discussion, we choose one seed as an example: "Head First Java." We assume that the seed we choose is part of a data record we are searching for. We will try to identify data record boundary by growing this seed. The steps to grow a seed are:

(a) Identify initial data region
Grow the seed until it reaches some well-known boundary, for example "<table>" tag. That will be our initial data region.

(b) Identify data record boundary
In this step, we can choose any existing syntax based algorithm and heuristics, or their combination aided by the extra information about the seed. In this project, we plan to experiment an approach that uses the ranked key phrase information. We will choose the higher ranked neighbor of the seed and the separators between them will be the record boundary. For example, in the page from amazon.com, "Core Java" is the neighbor we will choose and the separator <tr><td> will be identified as record boundary.

(4) Data extraction for all pages of the site with known record boundary
Once we identify the record boundary, we will apply the data region and record boundary information to all the result pages of this site. For each page, we will keep track of how many key phrases for that page have been identified. If there is a key phrase that has a ranking value above a threshold that is not included in any data record, it means we may have missed an additional region. We will use that key phrase as seed and repeat step 3 to help extract records from those missed data regions. At the end, we will output any un-extracted key phrase into a log file so that we can improve the algorithm accordingly.

(3) and (4) will be repeated for all sites. We assume each site has different presentation style and the data region and record boundary is not used across sites.

3. Functional specification

(1) Input: a list of web pages from different sites based on the same search keyword.

This algorithm can be a post-processing step for syntax based extractions or combined with other extraction approaches. We will experiment this algorithm directly on web pages so that we can get a reliable recall/precision statistics, but it can easily be combined with other algorithms.

We will use locally cached pages to concentrate on developing the algorithm. For initial test data, we will try keyword search on book sites: www.amazon.com; www.bn.com; www.bookpool.com; www.half.com; http://www.powells.com to retrieve test pages.

(2) Output: a list of data records for each site. A log file about the missed key phrases for each page.

We will "clean up" HTML pages when doing preprocessing and also we will try to refine data records to be simple text, and will make it humanly readable without reformatting them into HTML. This will facilitate algorithm evaluation and make it easier to integrate with data field extraction work.

Technical specification

- Programming Language to use: PERL
  PERL is convenient for text processing and regular expression matching. It is also suitable for fast prototyping in an area like data record extraction. If the algorithm is proved to be effective, we can use other languages to build the system to integrate with other processing systems.

- Implementation

1. Preprocessing
   General clean up: remove the obviously unnecessary parts and attributes.
2. Get a list of key phrase for each site
   We will simply get any string between ">", "<" that contains the search keyword.
3. Compile the ranked key phrase list
4. Sort the sites into a queue based on each site's ranking score.
5. process a site:

(1) Identify seed;
(2) Read the whole preprocessed page as a string and identify the boundary separator based on the seed;
(3) Process page based on data region and record boundary to extract data records;
(4) If any key phrase having ranking score higher than *r* is left out, go to (1) and repeat.

(5) Output the key phrase that is left out into log file.

6. Repeat step 5 until all sites are processed.
7. Output all data records as text file for each site.

- Schedule

Milestone	Date
Proposal Due	3/18/2005
Project Page Construction	Ongoing
First Meeting with Mentor	3/15/2005
System Construction	3/19/2005 - 5/5/2005
Initial prototype, proof of concept	3/30/2005
Second Meeting with Mentor	4/25/2005
Project Completion and Demo	5/6/2005

Test / Evaluation / Experiments

Test will be conducted on the locally cached data and the most important criteria of evaluation are recall and precision. We will compare our results with the corresponding results by using MDR [3]. MDR is a web mining system that identifies and extracts regularly structured data records (e.g., products and data tables) from Web pages.

We will measure system performance by elapsed time and we will conduct asymptotic analysis, that is, review the limiting factors on our algorithm.

Reference:

1. D. Buttler, L. Liu and C. Pu. *A fully automated extraction system for World Wide Web*. In Proceedings of the 2001 International Conference on Distrubuted Computing Systems, 2001.
2. D. W. Embley and Y. Jiang and Y.-K. Ng. *Record-boundary discovery in Web documents*. In Proceedings of SIGMOD, 1999.
3. B. Liu, R. Grossman, and Y. Zhai. *Mining data records in Web pages*. In Proc. of the ACM SIGKDD, 2003.