A Web data extraction approach to harvesting data from online sources

Richi NAYAK and Magnus HAUGAASEN
School of Information Systems, Queensland University of Technology
Brisbane, Australia  
{r.nayak@qut.edu.au, magnus.haugaasen@gmail.com}

Abstract
With the Web becoming a main source of data representation, businesses have opportunities to gather data from various independent web sources and condense these data into specialized services. However, there is no unified structure of web pages and therefore extracting data from sources can be a complex task. We present a solution to locate and extract data from a large group of online bookmaker pages to provide a real-time service to deliver price on sporting events.

1. Introduction

Recent years has seen a growing trend of enterprises specializing in collecting very specific data from various Web sources and collaborate these data to form a business concept independent from that of the source of the data. The idea of this paper is to propose a real-time price service delivering odds on sporting events from a large group of online bookmakers. On major sports, bookmakers offer the same events but at different prices. Differences can often be up to 30% for a single outcome, either due to bookmaker preferences (favourite – underdog), or due to volume of bets. The prices or odds for events fluctuate on a frequent basis according to the movement in the market. Since the number of bookmakers and customers is growing, it is becoming difficult to track the movements of this huge market without the use of automated tools. There are available services which can provide services for collection and comparison of data but often the intervals between data collection is too large to pick up market changes [3].

How to obtain information in html is researched much in traditional web information retrieval. There are spider, crawler and robot [1,3,4] for general information retrieval, and wrapper for each special web source [5]. Traditional spider is improved in this paper to fetch the accurate and dynamic data of interest. This paper presents a software solution which is able to scan and harvest data from bookmakers web pages for prices on some sporting and horse racing events, and process the collected data by incorporating the concept of web content mining.

2. The process to harvest data from online sources

The objective is to create a system which on an uninterrupted basis is able to lookup target Web pages, extract required information and process this information by matching and comparing data from various sources. The system provides users with
information on risk free bets at real-time from sport events or from horseracing. It gives the opportunity to the user to calculate stakes for arbitrage occurrences and possible profit, and to select a particular event accordingly.

2.1 General Features

Since we are focused towards collecting the dynamic data (subjected to rapid price changes, e.g., betting data during a horse race), the data should be collected in short intervals - almost real-time - to ensure the freshest possible data is included. In doing so, the extraction method should create as low as possible load on target web servers, and must run continuously with minimal need for user interaction into the process. Another requirement is the quality of the data collected. When the objective is to deal with products and prices, and subsequently provide the recommendation to users according to the comparison and calculations based on these numbers, it is of importance that these numbers are most accurate. The data generated by web content extraction must be of a universal format such as XML that can be read by any application on any system.

2.2 System architecture and the process

The system (figure 1) includes processes for (1) the data discovery to identify target Web pages, (2) the extraction of relevant data from Web pages and (3) pre-processing of data to generate useful information for end-users.

2.2.1 Data discovery and selection

The first step is to identify and select the possible data sources that form the base for data extraction. This involves the fetching of web pages with information relevant to one or more queries without any specific user request in mind. The application developed in this project is very specific in its nature - the data is only related to the bookmaker prices for certain sports. Since the links to most pages in this application remain static and only the content within is dynamic, the page links were manually localised through static assignment of URL’s for the web crawler scripts to access. In
this way, we reduce the number of pages visited, and restrict the data extraction from
d Pages of interest; thereby eliminating the need to deal with useless data.

Another method tried was keyword search through two approaches: (1) enter a
key word in the Google search engine to fetch pages, and (2) perform a keyword search
in a web spider engine to retrieve pages. The Google search engine can point to the
current page through a regular link, however, it has no knowledge of sporting events
added after their web crawlers snapshot of the site. Also, search engine robots tend to
skip links that are dynamic containing elements such as ‘?’ , ‘&’. Web crawlers also
face problems to localize the data of interest with high accuracy. An inaccurate data,
e.g., wrong price for an event, leads to a conflict.

2.2.2 Data extraction

There are many ways for discovering and extracting data from selected target web
pages such as the use of screen scraping, Web crawling with text mining, Web queries
and regular expressions for direct fetch. The screen scraping methods extract data
from the screen buffer of computer terminals which we request (in this case a web
page), and outputs the whole page regardless of web page structure. Other tools and
methods such as Web queries and Perl scripts might not be able to grab all frames and
iframes from a Web page. However, to use the screen scraper solution, one either has
to visit pages manually or have to write scripts to interact with web page. The Web
crawlers with text mining provide full functionality of gathering of Web pages and
extracting keywords from gathered documents or point to Web pages which have the
best match for a given keyword [1]. However, Web spiders do tend to collect the
inaccurate data as well as they can not access pages with dynamic links.

An efficient way to extract data from target pages in a generic manner is
combining the Web crawling approach to fetch pages with a “do-it-yourself”
information extraction using regular expressions. The regular expressions allow
creating specific patterns for extracting specific data. These patterns can also be reused
indefinitely reflecting changes in dynamic content. We use PERL with additional
modules Crypt::SSLeay which ensures connectivity to secure Web pages (HTTPS) and
WWW::Mechanize which extends Perl original Web page access functionality
provided in the module LWP::Simple to give programmers the Web crawler
functionality. These modules provide functionalities such as crawling to specific depths
of the Website, manipulating forms such as tick boxes and radio buttons, pushing
buttons to name some, manipulate the user-agent field etc. To mimic a real-life internet
user we can manipulate the user agent field with the WWW::Mechanize so that it
reports itself to be a well known internet browser: $mech->agent_alias ( 'Windows IE
6' ); $mech represents the WWW::Mechanize object we are dealing with followed by
the method which sets the user agent field: agent_alias (BROWSER_NAME) . As a result, a
target website will identify us as a user which browses the internet with any browser.

While web crawlers may perform useful tasks it can affect web sites it is
applied to. It is important to define a polite access interval and how often it is
necessary to refresh the data we collect [2, 6]. Our objective is (1) to keep the average
freshness of pages in its collection as high as possible (having the out-dated pages as
few as possible), and (2) to keep the average age of pages as low as possible (having
the local copies of pages as new as possible). We follow the proportional policy
defined in [2, 6] by re-visiting more often the pages that change more frequently. The
visiting frequency is directly proportional to the (estimated) change frequency. For
regular sporting events such as tennis and baseball we would visit pages on a slightly higher interval than horse racing in which the odds fluctuate much more comparatively.

The simple formulas for intervals are: \((I + T)\) for regular sports; and \((T)\) for horse racing, where \(I\) is the set interval time that long the application will sleep before initiating a new command for data to be extracted. \(T\) is the time the application takes to process collected data before going in to the sleep state. The sleep interval for regular sports is set to 30 seconds while horse racing data extraction has no interval. A simple test was made with two scripts A and B that read the same page, but at different intervals 30 and 5 minutes respectively. To get a good variance in prices this was tested prior to the start of the game when price changes are highly anticipated. Script A can only detect two changes during this half an hour while script B detects any changes every 5 minutes. The gap in script A is a blind spot since we cannot with 100% accuracy say that this data represents the current real-time data.

2.2.3 Data pre-processing

*Event prices:* The first step is to determine the price format used by the bookmaker in which the data originated. There exist three different odds formats; Decimal, Fractional and American. Any data extracted where the odds is represented in either a fractional or an American format is transformed into the decimal format.

*Contestant/team names:* In order to match data from different sources it is important that data such as team or player names are similar. Different bookmakers might present a team with different name based on their own discretion. The text string with the team name is transformed in a way so that an occurrence at one bookmaker with a similar occurrence at another bookmaker is compared.

*Outputting data:* XML (well-formed) is used as a standard for any dataset resulting from data extraction to allow the portability of respective data sets.

3. Conclusion

Extracting data from web pages have vast different opportunities. Choosing the right method to extract this data is crucial, especially with regards to speed and precision. This paper discusses a method of retrieving pages with a web crawler and extracting data with custom written regular expression patterns for a particular application. We successfully built an intelligent software solution with the use of advanced Web techniques, which is able to harvest and process the data from bookmakers web pages for prices on some sporting and horse racing events.

**References**