WebSpy: Retrieving Web Contents for e-Business Intelligence

Simon Fong, Shirley Siu
Department of Electrical and Electronic Engineering
Universidade de Macau
Macao

Aixin Sun
School of Computer Engineering
Nanyang Technological University
Singapore

***Abstract*--WebSpy is designed to assist users to retrieve and monitor competitors' websites information. It has two parts, namely Price Watcher and Market Watcher. Price Watcher operates as an information-surveying tool, which takes competitors' URLs and "spies" on the prices of some selected products over the WWW. The prices collected from the competitors are stored in a local database for doing price comparison at the front-end of an e-Commerce website and for market research at the back-end. In particular, we have implemented two algorithms for product name matching and price extraction in order to achieve this. By selectively showing the competitors' prices along with ours on our online store, it may help encouraging consumers to commit placing on the spot. The Market Watcher is an information-collection tool, which assists the users to monitor the specified competitors' Web sites and locate their relevant information automatically. One of the two operational modes is to constantly scan the predefined competitors' URLs, checks if there are any changes in their content. The other mode is to query some popular search engines for discovering information (or news) relating to the competitors and/or their products. The motive of WebSpy is to give the user timely and quality information about his competitors in an automatic way otherwise would have to be done manually.**

***Preferred Topic Areas*—Track 3. E-Commerce, Track 14: Web Content Generation, Usage and Management**

I. Introduction

Accurate and timely information is essential for any business to remain competitive. With the Internet, we can gather a tremendous amount of business intelligence information on prospects, competitors, vendors, suppliers, customers, or other companies. Most businesses leverage their home pages as a marketing or communication tool for generating and retaining business. As such, corporate sites may provide annual reports, news articles, business ventures, prices and information about products and services. Any company who is concerned about business intelligence collection would spend a good portion of research time reviewing the information available on the Internet. They would usually employ a team of staff assigned to do the monitoring constantly over the competitors' websites for collecting intelligence information. This is a tedious manual process as it involves regularly visiting those websites, checking for updates, downloading the information, filtering and organizing the information.

In the past few years, there are many Web monitoring software programs developed to keep track mentions of companies, people, brands, topics, news, etc. However, most of these software programs find information from only certain news resources such as CNN, Dow Jones and BBC [1]. Many of them are hosted application services that means integrating their services into one's own local IT infrastructure would be somewhat challenging. Furthermore, very few of them were designed specifically to monitor competitors' websites for prices, updates on some selected group of products.

Recently we have designed and developed a generic version of information agents called WebSpy that is used to track the news, prices and mentions at the competitors' websites. This paper is organized in this way: in the next section, the two agents that are designed as a total solution for WebSpy called Price Watcher and Market Watcher respectively are described, subsequently their technical design would be given as follow, and finally a conclusion is drawn.

II. WebSpy #1 Agent: Price Watcher

Price Watcher, as its name suggests, takes the responsibility of retrieving prices of products from competitors' websites. The prices taken from there would be used for doing a price comparison at the front-end of our website. Users who are visiting on our website, while browsing our products, can click on the 'price compare' button to check how our products are better priced than others. This kind of on-the-spot price comparison provided by Price Watcher is intended to convince online shoppers that the offered price is indeed the best that can be found on the Internet, thus discouraging them from going away to the other sites. A snapshot of the application of Price Watcher from the prototype is shown in Figure 1. The agent can be configured such that only the prices higher than (or equal to) ours are displayed. The main objective is to encourage the consumer to commit a purchase on the spot at the current site.

The differences between our Price Watcher agent and most of the other Web-based price comparison software and portals [2, 3, 4] existing on the market are as follow:

1. **Designed for e-commerce shopping mall to use.** Price watcher is a price-monitoring tool used by a shopping mall to show their customers that their offered price is the best available. The usual Web-based price comparison services are made publicly available for web surfers to compare prices by submitting the product name. They work more like a search engine.
2. **No broker and nor public database is used.** For most of the price comparison services, there exist a mediator which

Simon Fong (Dr) is the correspondence author of this paper with contact details as follow: (mail): Faculty of Sci & Technology, University of Macau, Macau, (Email): simon@enetique.com.sg, (Fax): +853 838 314.

is usually the web server or service provider, and a centralized database is also used to maintain the price information available to the users. In our watcher agent strategy, a private and confidential database that holds the competitors' price information is located at the local site.

3. **No participation of retailing shops is required.** The way that most price comparison services work is they let the participating stores to submit their latest prices to the mediator. Our approach is different because there is no need to get the competitors involved. Instead, the Price Watcher stealthily downloads the price information from the competitors' web sites.
4. **Forms part of the Competitor Intelligence strategy.** The price watcher is to be implemented as a part of the competitor intelligence strategy that includes information retrieval, filtering, analysis, and presentation [5].

Figure 1. Snapshot of the application of Price Watcher in the prototype.

III. WebSpy #2 Agent: Market Watcher

Market Watcher is designed as a multi-featured monitoring tool for retrieving relevant information from the competitors' websites. While it is true that there are many similar commercial products on the markets that help retrieving competitors' information online. Market Watcher differs in a way that it is specifically built for monitoring sections of the competitors' websites online, and these sections are configurable. Instead of being a general website monitoring tool that claims to monitor anything from financial news to latest movie hits, Market Watcher is specialized to retrieve information that are relevant to our sales team. The information include competitors' product prices, company news, product news, their mentions and their search result rankings from the most popular search engines. With Market Watcher, sales team will get the very latest and timely information about the competitors' movement on their websites. It is like employing an automatic agent who keeps an eye on a long list of online competitors round the clock. Since the software for Market Watcher is built and configured in-house, what and whom have to be monitored, and how the information is to be integrated with the user's existing business intelligence infrastructure can be tailor-made.

IV. System Architecture of WebSpy

The architecture of the WebSpy is shown in Figure 2. It is composed of two major parts. One part is the price watcher that "spies" on the competitors' prices over the WWW. The prices collected from the competitors are then stored in the local database. The other part is the Market Watcher. The Market Watcher helps the administrator of the shopping site stores the latest information about his competitors' web sites [6]. Besides retrieving information directly from the competitors' websites, it is able to collect news from some news sites based on a list of relevant keywords and a schedule pre-defined by the user (normally on a daily basis) [7]. It also contains a meta-search engine to help the user locate the information on the web easily. Search engines that currently supported are Yahoo, AltaVista, Lycos, Infoseek, Excite and Webcrawler, which are the most popular search engines available on the web. Both the search engines and the webpages downloading mechanisms are implemented at the Information Retrieval Layer. The local databases store the product and price information of the online shops at the storage layer. There are two modes of operation in spying competitors' price information: (i) it constantly scans the predefined URLs, (ii) it queries the search engines for any relevant information regarding the competitors. In the first mode, the user defines who the competitors are and also their corresponding URLs. The second mode helps users explore the Web for discovering information about the competitors.

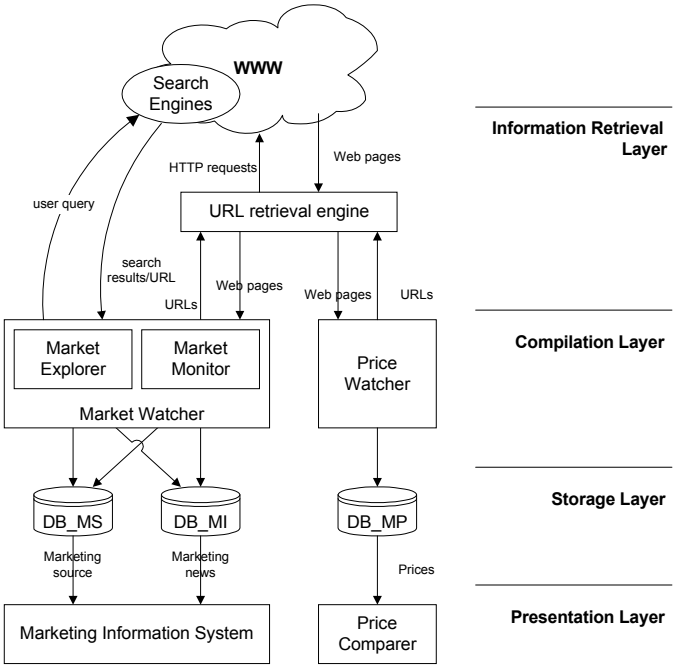

Figure 2. System Architecture of WebSpy.

V. PRICE WATCHER DESIGN

The four steps involves in the Price Watcher are dollar sign detection, HTML Semi-Data Tree construction, product name matching, and price information extraction.

Dollar sign detection is to determine whether the given web page contains any price information. The dollar signs are read from the system setting file named *DollarSign.list*, which include "S$", "$", "SGD". If none of these signs is found in one page, it is clear that the page does not contain any price information. The dollar sign detection process can be performed with any of the exact pattern matching algorithms. The input to the process is not the HTML code, but the pure contents of the page. The main reason is that some HTML code are embedded with JavaScript code where "$" can be contained in identifiers. Another advantage is to reduce the size of the input data and makes processing faster.

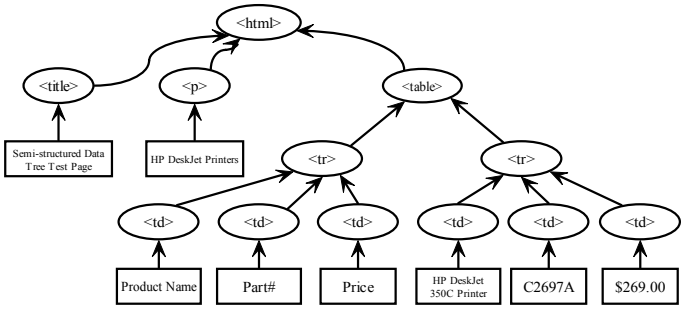

Figure 3. Constructed HTML SDT Tree from the Sample Page

The Semi-structured Data Tree (SDT) is constructed from the HTML tags. A driver-based HTML code to SDT converter is designed in this project. The driver indicates the HTML tags and the rules to be followed during the converting process. Not all the HTML tags are included in the driver. For example, <b>, <font>, <form>, <img> will be filtered out and ignored by the converter. This is to reduce the complexity of the converting process since these tags are not important in price extraction algorithm implementation.

The next step is to search for product names using the Product Name Matching Algorithm. The essential part, model number, is searched first. Once the model number is located, the brand will be searched prior to the position of the model number. The series part is either between these two parts or just after the model number. The Approximate Word Matching algorithm is applied to the series part matching because it is found to be most efficient as discussed in [8]. The final similarity obtained from the three parts is compared to the predefined threshold.

Once a product name is matched, the price is then located with the price extraction algorithm that has also been discussed in [8]. With the help of SDT model, the extraction process is carried out in the following order shown in Table 1. The software is implemented in Java and the main classes are shown in Table 2

The prices appearing in the Web page are normally in this form of *D*123,456.78, where *D* denotes a dollar sign. A state machine is designed and implemented to extract such price strings from the given text strings. In Figure 4, the "$" character indicates any matched dollar sign.

Searching Nodes	Possible HTML tags
The current data node	<p>, <title>, <body>, <h1>-<h6>, <td>, <th>,
The current data node and all children nodes	<p>, <body>, , <td>
The adjacent data node in same level	<p>, , <td>, e.g. in the same row of one column-wise table.
The sub-tree headed from the parent of the current data node.	<tr>,<td> e.g. in the same column of one row-wise table.

Table 1. Search Nodes and Possible HTML Tags

Class Name	Class Objective
DatabaseAgent	Contains all the database interfaces of the Price Watcher.
PriceURLDownload	To download the Web sites where the Price Watcher needs to be applied to.
DollarDetector	To detect dollar sign and extract the price strings from the text string.
HtmlTree	To construct HTML SDT from any given HTML page.
StringSimilarity	Implementation of the approximate word matching algorithm.
TreePriceAgent	Implementation of the price extraction algorithm, e.g. to extract the price information from the HTML SDT.
PriceWatcherThread	To control the scheduling of the Price Watcher.

Table 2. Price Watcher Main Classes Description.

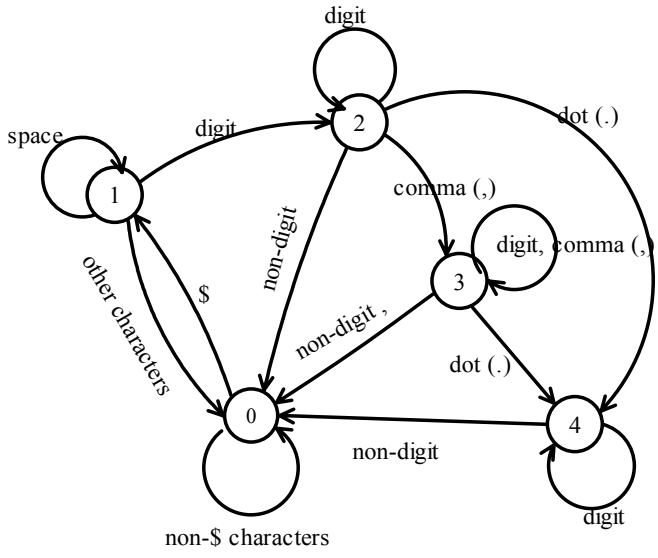

Figure 4. Price String Extraction State Machine

From all the states (i.e. state 0 to state 4), there are paths going back to state 0, if no specified inputs are matched. The price string is extracted just before the state moving from state 2, 3 and 4 to state 0. The possible matches are:

State Number	Possible Matches	Valid Price
1	S$ (any defined dollar sign)	No
2	S$123	Yes
3	S$123,456	Yes
4	S$123,456.78	Yes

Table 3. Price Extraction States and Possible Matches

Some sample user-interfaces of the price watcher agent are shown in the appendix.

VI. Market Watcher Design

The two main challenges in Market Watcher development are user profile construction and document ranking. The user profile is a set of keywords obtained directly from the users. The document ranking is based on the popular vector model.

The input for the Market Watcher is the category profile and a set of Web pages. The output is the information indicating which pages are relevant to the user's requirement. The information is stored in the local database and news report can therefore be generated. The information includes the Web page title, updated time, URL, similarity level, and summary. The summary will be the first one or two sentences of the page as we assume that most of the copy-writers give a summary in the beginning of the news article.

Class Name	Class Objective
DocDatabaseAgent	Contains all the database interfaces of the Market Watcher.
MarketURLDownload	To retrieve the monitoring Web sites.
TextAnalysis	Apply operations on text, e.g. extract the index terms, and calculate the document similarity level.
WordList	Perform operations on index terms, e.g. sorting, term weighting.
PorterStemmer	Implementation of the Porter's algorithm for word stemming.
MarketWatcherThread	To control the Market Watcher scheduling.

Table 4. Market Watcher Main Classes Description.

A. Instant News Watcher

The Instant News Watcher is developed to retrieve the most updated news from homepages of some Web sites where there are instant event sections. The inputs are category profiles and homepages, and the output will be the updated news extracted.

The Instant News Watcher is a special case of the Market Watcher where more frequent monitoring is required. However, the output is the news instead of the summary of the entire page. The entire structure and most of the contents of the homepage of one Web site will not be updated on daily bases. The frequently updated part is the instant news section or instant event section. Hence, the document-ranking algorithm to calculate the similarity level of the profile and entire Web page cannot be applied to the Instant News Watcher.

In the Instant News Watcher design, the content of the Web page is divided into small sections based on its internal structure with help of the Semi-Data Tree Model. Each time, one small section, e.g. one paragraph, one table cell, or one list item, is compared to the category profile. Since the input data is not so much as the entire Web page, the similarity level given by the document-ranking algorithm will be relative low. For this reason, it's hard to set any threshold. Therefore, the exact pattern matching is good enough in this case. If one section matches any of the keywords or key terms from the category profile, the section is considered to be relevant and will then be saved to database. The two main classes involved in the Instant News Watcher implementation will be **HtmlTree** and **StringList**. The duplicated sections will be detected before storing to the database.

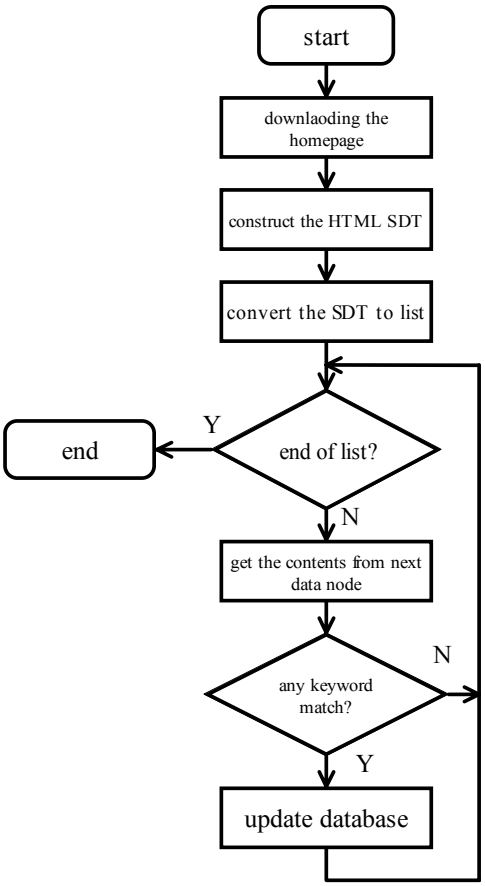

Figure 7. Instance Watcher Process Flow Chart

B. Market Explorer

Market Explorer is a part of the Market Watcher. The objective of the Market Explorer is to assist users to locate the required information with a number of popular search engines available. The currently supported search engines are AltaVista, Lycos, Excite, Yahoo, Catcha and InforSeek.

The inputs of the Market Explorer are the users queries and selection of the search engines (i.e. users may select any

combination from the six supported search engines). The output will be the top matches (i.e. matches appearing on the first page) of each search engine. The information about each match includes the page title, URL, summary, the source search engine and finally the matching order in that page, for example, the 5th match from Yahoo. The user query and matches will be saved to the database to form the user search history as well.

A target that we want to achieve in the searching process design is the time efficiency. If the unit search time is defined to be the time interval from when the query is submitted to one search engine to the time the matches are retrieved, it is possible that the matches from six search engines can be retrieved in one unit search time, by using multi-thread technology. That means the six search engines are activated simultaneously

VII. Conclusion

In this paper, we have reported an autonomous software program called WebSpy that collects competitors' product prices, news and monitor their updates on the web. It consists of two parts: Price Watcher and Market Watcher. Price Watcher is designed for the use of an e-commerce shopping site at the front-end, and Market Watcher is built as a market research tool for the users at the back-end. They both run autonomously as to relieve monitoring tasks otherwise done by human. The collected intelligence information by WebSpy is supplied to sales managers for business decision making. A proactive use of the price information is to enhance shoppers' confidence by doing an online price comparison. The application of PriceWatcher technology is believed to be relatively new and would create an impact on the way that retail shops market their goods online. The first online shop that applies this technology would benefit most, because it helps to place their business in a market position one step ahead of their competitors. So far this project has implemented the scanning functions. It is envisaged that WebSpy will be extended to include data-mining functions on competitors' information and automatic reporting as well, in the near future. We are in the progress of integrating WebSpy into a full infrastructure of business intelligence.

References

[1] Dow Jones Interactive, "http://www.bestofboth.com".

[2] B. Krulwich, "The Bargain Finder Agent: Comparison price shopping on the Internet", Bots and other Internet Beasties, SAMS.NET, http://bf.cstar.ac.com/bf.

[3] B. Doorenbos, D. Etzioni, S. Weld, "A Scalable Comparison - Shopping Agent for WWW", Technical report 96-01-03, Univ. of Washington, Department of Computer Science and Engineering.

[4] Pricewatch for Computer Products, "http://www.pricewatch.com".

[5] Andrew Pollard, "Competitor Intelligence: Strategy, tools and techniques for competitive advantage", Financial Times Management, 1999.

[6] M. Pazzani, D. Billsus, Adaptive Web Site Agents, ACM Autonomous Agents '99, pp.394-395, 1999.

[7] J. Allan, R. Papka and V. Larrenko, On-line New Event Detection and Tracking, ACM SIGIR '98, pp.37-45, 1998.

[8] S. Fong, A. Sun, K. Wong, Price Watcher Agent for E-Commerce, Intelligent Agent Technology, pp.294-299, 2001.

APPENDIX

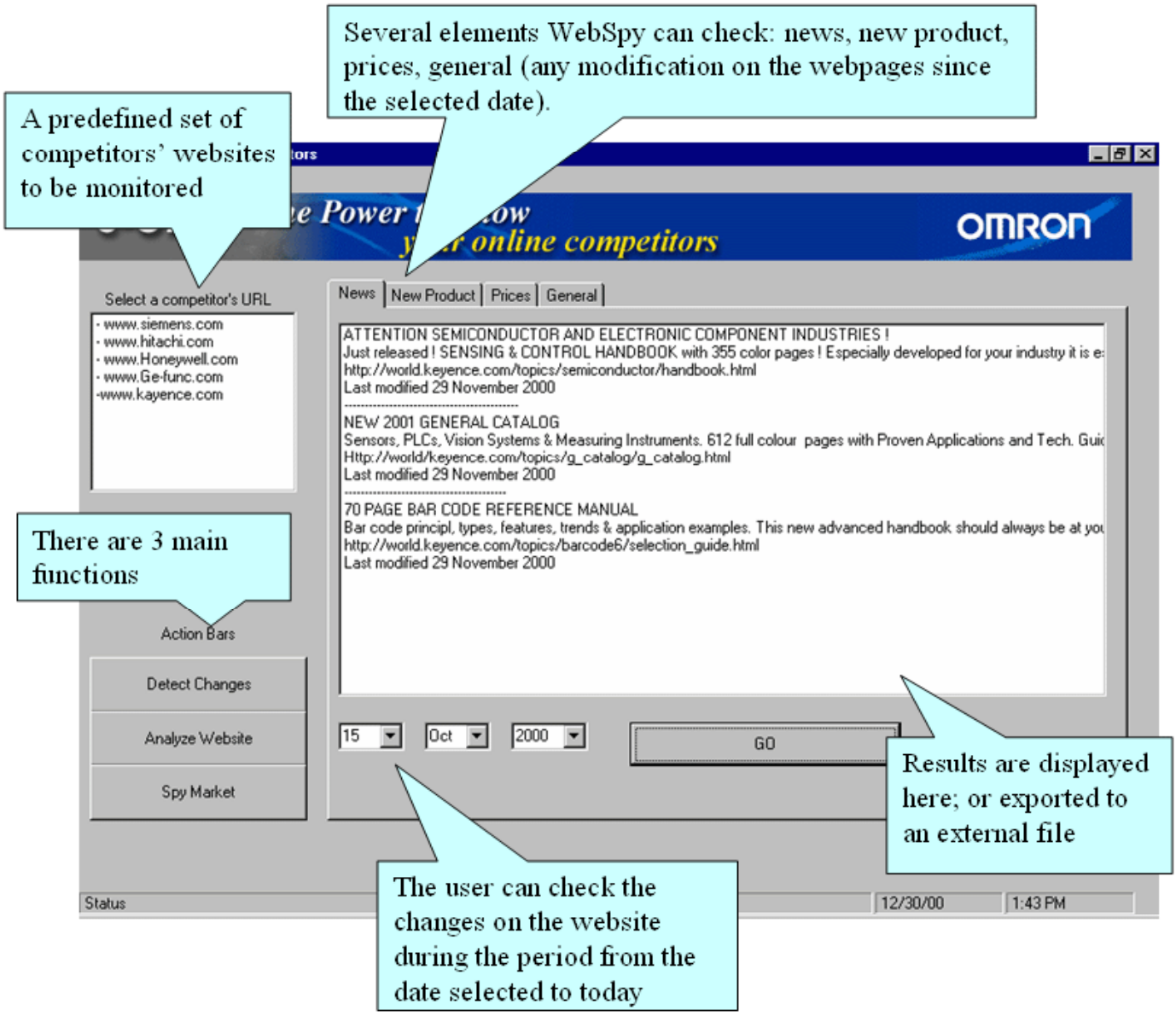

Figure 8. A snapshot of the WebSpy user interface