An Intelligent Agent for Retrieval and Monitoring of Tasks and Assignments

Carl Magnuson
University of Minnesota Computer Science Undergraduate

December 10, 2006

Abstract

The possible uses for domain-specific searches guided by methods of artificial intelligence are vast. This research project attempted to create an intelligent software system capable of retrieving and displaying a student's pertinent coursework (homework assignments, laboratories, quizzes and tests) from a given web address.

This project can be used as a model for the application of domain-specific search to other bases of information. Such specialized searches have many possible powerful applications, returning very specific information compared to what is retrieved by general purpose search engines.

The project was successful in many respects and also had its shortcomings. These shortcomings made the point clear, that a great deal of intelligence is required to retrieve information from a specific domain, and that the difference in required intelligence between obtaining mostly correct results, and nearly perfect results is a large one. However the successes showed that a fairly simple search can be quite effective when used on a limited domain, and that future work could drastically improve such a search to work over broader domains.

Introduction

This project attempts to apply the power of an intelligent software agent to a student's task of retrieving homework assignment, laboratory, and exam information and dates from web-based sources. The project attempts to identify and implement successful heuristics for extracting this information from provided sources, and to then output it in a concise yet legible form.

Related Work

There is a large volume of research material regarding the search of the internet or databanks as a whole, but not as much on the topic of searches directed towards smaller domains (such as a course website). Most of the available research also relates to finding entire web pages, instead of specific sections of data within web pages.

Research done by Jian-Shuang Deng[1] explored such a topic, finding that the use of spatial data, keywords, and structure extraction is an effective method. With proper rules in place, Shuang was able to use this method with a high degree of success, on both test pages and new web sites.

There are many similarities between the objective of this project, and that of data mining. Data mining is the search for information from large databases[2]. There is a large base of research to be found on the topic of data mining. Applying the tools of data mining to web based searches could yield positive results[3]. However, because of the nature of web based information sources, it would be undesirable to rely only on the tools of data mining, and ignore the contextual references inherent in HyperText Markup Language (HTML) files[4].

Using contextual and page structure information to guide a search is a powerful method[3]. Extracting data from link structure and textual proximity and display style is a successful search method employed by the search engine Google[5]. There is also research indicating that the processing and search of HTML tables will yield positive results[6, 1, 7].

Problem Description

This project works towards the goal of an intelligent agent that is able to accurately (within a margin of error) retrieve and display student assignment, lab-work and exam dates and information for provided class websites. In order to move towards this goal, a number of incremental steps had to be first completed.

First a software framework was designed to retrieve, traverse, and store the data in the provided web address. The content was descretized into separate entities for each line in the original document. The HTML and other extraneous material was then be stripped out, and references left to HTML code that could guide a heuristic search such as font-size, boldness, and other formatting cues.

At this point, the search process for candidate sections of text begins. Heuristics were envisioned to evaluate the text based on its content, relevant keywords, proximity to other relevant content, and formatting cues[2]. There is reason to believe that relevant keywords and local content will be

strong guides for heuristic methods, *"Using positive examples alone can be especially useful in Web document classification"*[3]. *"Based on the common structure of Web page layout, the most important section of a web page can be detected"*[8].

After these methods were developed, a search algorithm was implemented to search the solution space. At this point the agent can be run on a variety of course web-sites, and the accuracy and usefulness of the intelligent agent can be analyzed. From this stage, predicted improvements can be tested and implemented, and conclusions can be drawn about the utility of an intelligent agent for performing such a search.

Methods and Discussion

Retrieval and storage of data

A software framework was designed to retrieve, and store the web page data. A seed is given as input to begin the search[9]. From this seed, finding all of the linked pages was accomplished by modifying example link-checking code found in Ian Darwin's *Java Cookbook*[10]. This code was modified, so that it would keep a list of all local HTML pages found, as well as search pages generated by PHP (which many of the test web sites contained) in a breadth first search [9]. Additionally, a limit was put on the number of links to traverse (depth of the search), as suggested by literature[11] as well as the result of testing the program without a limiting factor.

With the list of valid web pages found, another procedure downloaded all of the web pages found to be local to the input site, and created a list of page objects to correspond with the downloaded web pages. These pages were populated with the data from the web page they were associated with - line objects, metadata describing the page, and a weight value to be determined in the search. Each line object contains the original HTML, the line number, the text stripped of tags (by regular expression matching), and a weight value .

Search of data

The search process scans lines to see if they contain keywords relevant to the search, and assigns them a weight based on the keywords they contain. Pages are assigned a weight as well if they contain keywords in their url or title. An aggregate weight for each page is calculated based on the page's

weight, and the weight of the lines it contains. This initial search then returns the page with the highest aggregate weight, and any other pages with a weight close to that of the maximum weighted page.

The pages(s) returned are then processed, so that highly weighted lines and those in their proximity remain in place, and lower weighted lines are removed. The percentage of a page's lines that are returned is inversely proportional to the total number of pages that were returned from the first stage of the search. That is if only one page is returned from the first portion of the search, its entire text will be returned as the result of the entire search. If numerous pages are returned from the first stage of the search, only the sections deemed most relevant from those pages are returned.

Output of results

The page(s) found by the search are output to an HTML file. The title of each page is printed, as well as each of its lines with enough weight to pass the search algorithm. A horizontal bar serves to separate each separate page output to the one result page.

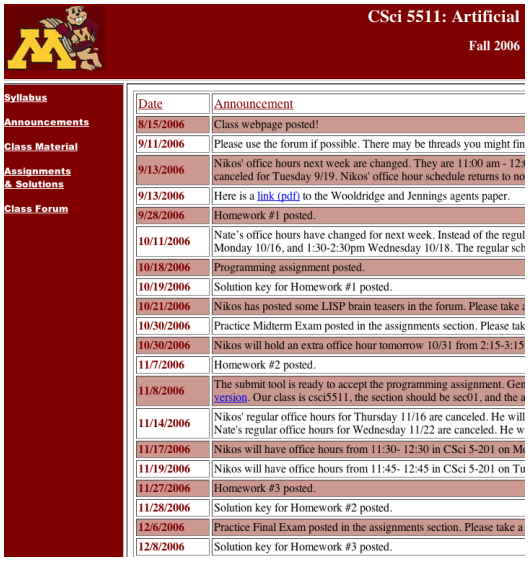

Figure 1. Example course website

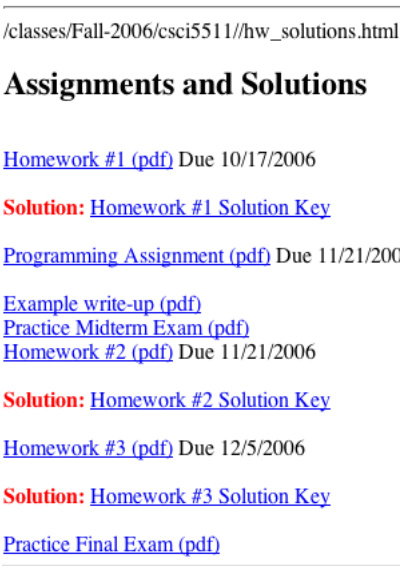

Figure 2. Example output file

This is a very simplistic, yet effective method of displaying the search results. In most cases this yields an easy-to-read web page with all of the relevant coursework information, preserving links to homework assignments

from the original HTML. This method does have some difficulties however, in that sometimes HTML markup may span multiple lines of text, resulting in improperly formatted output.

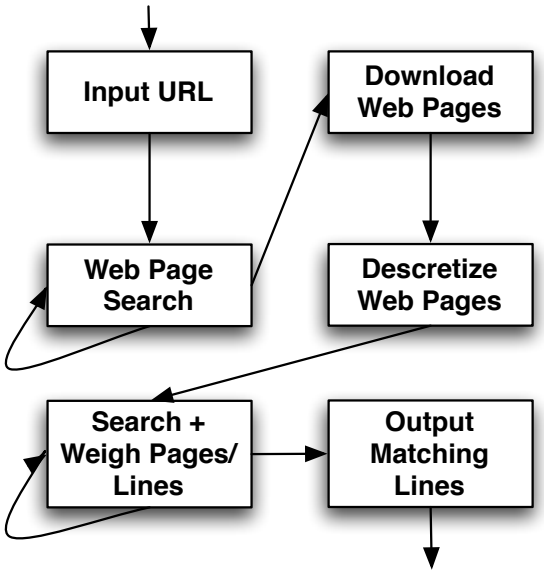

Figure 3. Flow of agent's search.

Future Work

Searching for dates

One of the proposed heuristic methods for this project is whether or not a line of text contains a date, and if so, how close that date is to the current day. It is expected that this will be a strong heuristic method. Unfortunately it has proved exceedingly hard to implement an efficient search for a date within a string. There are various ways to represent a day in English text, such as a coma, period, or slash separated list of numbers, days, months, years, or their abbreviations.

Such a heuristic method would likely be a good indicator of homework assignments, or other relevant course material, as such information is almost always displayed in conjunction with a due date, or an assigned date.

HTML output

An HTML output file was chosen, so that relevant links would still be

useable. One complication however is that on some pages where HTML tags span lines of text, the HTML output file can be corrupted. For example if a <bold> tag starts on a line of text being outputted, but is closed on a subsequent line not sent to the output file, the output may be bolded from that line down. There are further complications from links, or color changes which span lines of text.

This seems to be an inherent difficulty as a result of descretizing the pages into lines, which may contain partial HTML tags, or be surrounded by a multi-line tag. One possible solution to this would be to build a regular expression matcher to find lines of text in the output file that are not valid HTML, and remove the offending tags.

Table extraction

Retaining the HTML table information, and using it to guide in a search could prove to be a very powerful heuristic method. The amount of contextual information stored in tables alone is huge[1]. The flexibility of HTML tables allow them to be used in many situations, to convey various data in a useful way[6].

Parsing of HTML pages

A possibly superior method to retrieving the information from the discovered HTML pages would be to build a parser and lexer which would read in HTML files, and output objects containing the original page data. Such an option would make it possible to make more use of the context clues given in the HTML code, and would make table extraction a much easier task. Such a method of parsing web pages is currently used by Google as part of their web indexing strategy[5].

Probabilistic methods of text extraction and output

Another possible alternative to extracting useful information from course websites would be through the use of Hidden Markov Models. Hidden Markov Models have been successfully trained to extract specific data from HTML pages with a high probability of success[12].

While not assured to be completely accurate, such a method would be acceptable assuming it has a high success rate, as its benefits would outweigh its risks in such a situation. If not used to perform the entire search, another

application of Hidden Markov Models could be to solve the task of extracting dates embedded in the document text.

Handing of PHP and other non-standard HTML pages

One difficulty faced in this project was dealing with dynamically generated PHP pages. If the web search algorithm were altered so it could more easily record accessed urls, as well as their associated get and post values, less redundant information would be retrieved. Implementing such improvements would not likely affect the accuracy of the search, it would increase the performance, by minimizing the number of HTML pages that must be visited by the search agent.

Conclusions

Even with a simple keyword search, promising results have been obtained. The useful assignment information was retrieved most of the time, stripped of less important material. The addition of simple proximity methods, and url and page title keyword matching methods have served to increase the accuracy of the search. On average, the search was able to find most or all of the assigned material from a course website, while returning a minimum of extraneous data.

In some instances, the search was misled by the number of keywords present in course syllabi, however the addition of the url and title keyword methods, which had a higher weight then a simple textual keyword, helped to reduce the number of these instances. The addition of a negative weight factor for pages which appear to be syllabi, and taking into account the number of lines in a page - so that lengthy pages don't dominate more concise ones also helped to solve this problem.

While in many cases correct in the data that is output, the search often returned data with the wrong typeface, coloring, incorrect links attached to it, or improper line spacing. This has no relation to the success of the search in finding the correct data, but in its useful application to finding and intelligently displaying the found data. Were the data read in in a more careful manner, or a parser used to retrieve the data, this would appear to be a relatively simple problem to overcome.

It is hypothesized that if additional heuristic methods are implemented (such as date information, typeface, tabular information, and web page creation dates), and a search based off of all these factors used, that correct results will still be obtained on a broader sample space, and these results will have less additional material than what is generated by the current search. It does not appear to be possible to build a simple agent that is able to retrieve the necessary information from a general course website with high precision. However it is possible to create an agent that is able to retrieve such information from a similar set of course websites, with a high degree of accuracy.

References

[1] J.-S. Deng, Q.-L. Zheng, and H. Peng, "Information retrieval from large number of web sites," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 4, Aug. 2005, pp. 2172–2177.

[2] B. Thuraisingham, "A primer for understanding and applying data mining," *IT Professional*, vol. 2, no. 1, pp. 28–31, 2000.

[3] J. Han and K. Chang, "Data mining for web intelligence," *IEEE Computer*, vol. 35, no. 11, pp. 64–70, 2002.

[4] T. Berners-Lee and D. Connolly, "Hypertext Markup Language - 2.0," RFC 1866, Nov. 1995.

[5] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998. [Online]. Available: citeseer.ist.psu.edu/brin98anatomy.html

[6] H. Chen, S. Tsai, and J. Tsai, "Mining tables from large scale html texts," 2000. [Online]. Available: citeseer.ist.psu.edu/chen00mining.html

[7] S. Li, Z. Peng, and M. Liu, "Extraction and integration information in HTML tables," in *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*, Sept. 2004, pp. 315–320.

[8] B.-H. K. Shakirah Modh Taib, Soon-Ja Yeom, "Elimination of redundant information for web data mining," in *Proc. IEEE ITCC'05*, vol. 1, apr 2005, pp. 200–205.

[9] V. Gudivada, V. Raghavan, W. Grosky, and R. Kasanagottu, "Information retrieval on the world wide web," *IEEE Internet Computing*, vol. 1, no. 5, pp. 58–68, 1997.

[10] I. F. Darwin, *Java Cookbook*, 2nd ed. O'Reilly Media, Incorporated, 0596001703, 2004.

[11] M. P. Robert Popp, Bohdan Maksymiuk, "Efficient information retrieval on the world wide web using adaptable and mobile java agents," in *Systems, Man, and Cybernetics, IEEE International Conference on, '98*, vol. 3, oct 1998, pp. 11–14.

[12] V. R. Borkar, K. Deshmukh, and S. Sarawagi, "Automatic segmentation of text into structured records," in *SIGMOD Conference*, 2001. [Online]. Available: citeseer.ist.psu.edu/borkar01automatic.html