Automatic Web Page Concatenation

Matthew Webber
February 22, 2005
Problem Statement

Despite the many applications and resources designed to make retrieving information from the Internet easier, some tasks remain time-consuming and tedious for users. Consider the following three possible user scenarios:

Scenario 1: A user is using a web search tool to look for an image, and wants to quickly scan through as many images as possible. She wants to compare and contrast images that may have many search results between them. To do this, she may have to look at many different pages, possibly opening them in different windows. The image-heavy pages may take a long time to load, and switching between them may be difficult.

Scenario 2: A user is using a search utility to search for pages on a web site. He has two keywords, and wants to find pages that match both. The site has a very primitive search engine, which only allows the user to search for one keyword at a time. To find pages that contain both keywords, he must search for one keyword, then search within each page of search results for the other keyword.

Scenario 3: A user has access to a news article that spans three web pages. She wants to print the article or save it to her computer, but the site she is using offers no "print friendly" version of the page. To perform either task, she must open each web page and print/save it separately.

In all of these scenarios, individual computer users are experiencing the same problem: they want to perform an action (view, search, save or print) on many different web pages at once. To do so, they must navigate to, load, and interact with each page separately. This process can be very repetitive and tedious, especially when large numbers of web pages are involved.

If the user had some way of viewing all content combined into a single "master page", actions could be performed on all data at once. The problem this project aims to solve is this: How can we empower computer users to combine multi-page content from arbitrary web sites?

In this project, we will focus on a target audience of individual computer users. In general, these users can't write code, and don't have direct access to the databases that store information they care about. They use many different web interfaces to retrieve information, and the sites they use change over time. A good solution to the problem will let users view combined content pages when they want, and view other pages unchanged, without unnecessary distractions. The solution should also work for users of high- and low-bandwidth Internet connections, and not introduce any long waits for content unless necessary.

Method

This project will be broken up into four phases, which will be executed in series. The completion of each phase of the project will serve as a benchmark to measure progress. Throughout the project, the primary problem to be solved will be dealing with the complexity and diversity of web pages that users use to gather information on the
Since search tools and site layout vary from site to site, it is unlikely that a single solution exists that will properly process every web page quickly and efficiently. While improving on every web page may not be possible, there are some common features in multi-page content that can be exploited to improve most such pages.

**Phase I. Is there multi-page content?**
The first task will be to recognize if a page is part of a multi-page content sequence. To do this, we will design an algorithm to find multi-page search features, like "Previous" and "Next" buttons, and consecutive numbered links. For this purpose, we will make use of the LAPIS text-pattern search tool developed by Professor Rob Miller's research group at MIT.

**Phase II. Where are the other pages?**
Once we have detected that we are on a page that contains multi-page content, how do we locate the rest of the relevant content? Some outbound links on the page may be duplicated, and others may lead us to content that we do not care about (advertising, for example). It's important that the page selection process has a very low rate of false-positives, as erroneous results mixed in with correct results may distract and confuse the user.

**Phase III. How can we combine results?**
After an ordered list of content-containing pages has been created, these pages will be retrieved and combined into a single master page. The naïve approach to the combination is to place all content from each page after the content from the page before it. This will result in master pages with a lot of repeated content—headers, footers, sidebars, and advertisements. In this phase, we will try to identify the content that is unchanging from page to page, and reformat the master page so that such content only appears once.

**Phase IV. How do we present the results to the user?**
The final problem to address for this project is the design of a user interface. Without disrupting the browsing experience or unnecessarily distracting the user, how can we present the “See All Results” option? In this phase, we will implement a browser plug-in that examines a web page, determines if the page contains multi-page data, and inserts a "All Results" button next to search links in the page if appropriate. When the button is pressed, the plug-in will fetch and combine pages into a local "master page", which will be displayed to the user. In this phase we will make use of the work of Michael Bolin. A researcher in Professor Miller’s group at MIT, he has done research on dynamically inserting content into web pages using plug-ins for the Mozilla Firefox Internet browser.

**Progress so far**
In the fall semester, I spent time becoming familiar with the Eclipse developing environment and the LAPIS code base, both of which are critical for my project. I made progress on Phase I and II described above, by
creating LAPIS patterns to detect and locate 'previous' and 'next' links, numbered links, and the navigation control panels on web pages where these appear. I created a Java class MultipageParser which analyzes a web page and can pick out relevant sections. I also integrated this class into the LAPIS code base.

The algorithms that this class uses to locate Multipage features were not perfect when they were first written. To test them, I gathered a representative sampling of multi-page content from many different web sites, processed the sites, and analyzed the results. I have since improved my algorithms to deal with some false positives, but solutions for others remain elusive.

**Applications and Extensions**

If completed successfully, this project offers a fast and convenient solution to the problems listed at the beginning of this paper (scenarios 1-3). By reducing the overhead for multi-page content scanning, saving, printing, and searching, this project will give users greater control over the data they want from the web sites they visit.

When connections to the Internet are scarce, this project may make better use of bandwidth time. Users could use this project to quickly get data from a paid service and log out. Groups of users that share a common set of Internet connection(s) in a home or small business could also use this project to make more efficient use of their time.

Future extensions to this project could focus on retrieving and combining data from many pages that do not have the well-defined order that search results and news articles do. This project could be customized for tutorials, discussion threads, or even crawling and combining all content on a small site.