OminiSearch: A method for searching dynamic content on the Web

David Buttler, Ling Liu, Calton Pu, Henrique Paques, Wei Han, Wei Tang
Georgia Institute of Technology
Atlanta, GA
{buttler,lingliu,calton,paques,weihan,wtang}@cc.gatech.edu

1. INTRODUCTION

To address the search problem over dynamic web pages, several domain-specific information integration portal services have emerged, such as **jango.excite.com** and **cnet.com**. These integration services offer uniform access to heterogeneous collections of dynamic pages using wrapper technology. Common techniques used for constructing wrapper programs often require embedding programmers' understanding of the specific presentation layout or specific contents of the Web pages, often with the assistance of semi-automatic tools.

However, even with partially automated tools, developing and maintaining wrappers turns out to be labor intensive and error-prone, especially for Web sites that are frequently changing their information presence on the Web. As a result, most information integration services are limited to a handful of sources and have a limited capability to effectively incorporate additional or updated content providers into their existing integration access framework.

The OminiSearch system demonstrates a method to incorporate new dynamic sources into an integration system in which useful data objects are extracted from dynamic Web pages without programmer intervention, and to maintain the capability to extract relevant information in the face of evolving structure in the Web sites.

2. OMINISEARCH ARCHITECTURE

The OminiSearch architecture consists of four main components: a search execution engine, the Omini object extraction engine, an autonomous Web crawler to discover and categorize new sites, and a context catalog to store discovered Web sites, their relevant contexts, and associated information extraction rules.

Users enter a search request, and choose from a list of contexts for their search (e.g., general Web search, book search, news search, etc.). The *search execution engine* first requests a list of Web sites that match the chosen context for the search from the *context catalog*. Then, for each Web site it constructs the appropriate URL based on the user search request and the Web site search interface description. The URL's are passed to the *Omini object extraction engine*, which retrieves the Web pages and returns extracted objects to the search execution engine for formatting. Results from each Web site are grouped together and sent back to the user. If a page format or query interface change was detected and the object extraction process failed, new rules are automatically discovered and an update is sent to the context catalog.

The final component of OminiSearch system is the *autonomous Web crawler*. It constantly searches the Web, looking for new sources that have searchable interfaces and are related to the contexts stored in the catalog. In addition, users can suggest static pages to add to the catalog or even define new contexts to incorporate into the meta-search.

The core of OminiSearch is the Omini object extraction engine, which is implemented in four steps. First, the Web page is prepared for extraction by normalizing it into a well-formed document, and converting it into a tag tree. Second, the primary content region in the document is located, and the minimal subtree containing this region is identified. Third, the boundary between individual objects is discovered. Finally, the objects are extracted from the document, leaving behind the irrelevant parts.

3. DESCRIPTION OF DEMO

The focus of this demo is on automated information extraction from individual pages, with a simple supporting framework to highlight this technology. The OminiSearch system is written as a 3-tier application in Java. Users submit queries via web forms to a servlet which incorporates the search execution engine. The servlet uses JDBC to request context information from the back-end catalog. The Web crawler acts as an independent entity, contributing new sources to the catalog asynchronously. We present the effectiveness of OminiSearch, in terms of precision and recall of the object extraction component, on over 100 different Web sites. We also show the utility of an integration system that can automatically incorporate content from dynamic searchable Web sites as well as static Web pages.

ACM SIGMOD 2001 May 21-24, Santa Barbara, California, USA