Siphon++: A Hidden-Web Crawler for Keyword-Based Interfaces

Karane Vieira[1], Luciano Barbosa[2], Juliana Freire[2], Altigran Silva[1]
[1]Departamento de Ciência da Computação
Universidade Federal do Amazonas
{karane, alti}@dcc.ufam.edu.br
[2]School of Computing
University of Utah
{lbarbosa,juliana}@cs.utah.edu

ABSTRACT

The hidden Web consists of data that is generally hidden behind form interfaces, and as such, it is out of reach for traditional search engines. With the goal of leveraging the high-quality information in this largely unexplored portion of the Web, in this paper, we propose a new strategy for automatically retrieving data hidden behind keyword-based form interfaces. Unlike previous approaches to this problem, our strategy adapts the query generation and selection by detecting features of the index. We describe a preliminary experimental evaluation which shows that our strategy is able to to obtain coverages that are higher than those of previous approaches that use a fixed strategy for query generation.

Categories and Subject Descriptors: H.4 [Information Systems Applications]: Information Search and Retrieval

General Terms: Algorithms, Design.

Keywords: Online databases, Hidden-Web crawler

1. INTRODUCTION

The hidden Web has had an explosive growth as an increasing number of databases and document collections are made available online. It is estimated that there are several million hidden-Web sites. Although the hidden Web represents a substantial portion of the Web, it has been largely unexplored. The main reason being that hidden-Web data is both hard to find and access. These data are not indexed by existing search engines, which are limited to crawl pages that have a well-defined URL. Thus, to access a hidden-Web source, users (and applications) first need to know where to find it, and then fill out a form in order to access the content.

In this poster, we present a crawler for automatically retrieving Web content hidden behind keyword-based interfaces. Being able to retrieve and index this content has the potential to uncover hidden information and help users find useful information that is currently out-of-reach for search engines. Unlike multi-attribute forms, keyword-based interfaces are simple to query, since they do not require detailed knowledge of the schema or structure of the underlying data. It is thus possible to create automatic and effective solutions to crawl these interfaces [1, 2]. We propose a new crawling strategy which *adapts* to the features of the indexes underlying the search interface. We show that this adaptation leads to improved coverage: our crawler retrieves more hidden content than previous approaches.

2. THE HIDDEN-WEB CRAWLER

Figure 1 shows the architecture of Siphon++, the proposed adaptive crawler. Siphon++ is composed of an *adaptive component*, which discovers features of the index, and a *heuristic component*, which derives the queries to retrieve the hidden content.

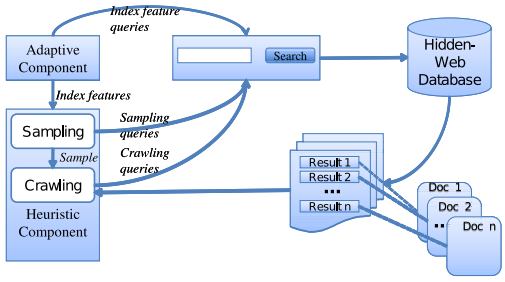

Figure 1: Architecture of the *Siphon++*.

The Adaptive Component (AC). The Adaptive Component (AC) detects the index features by issuing probe queries against the search interface. We call these queries *index feature queries* (see Figure 1).

Two techniques commonly used for compacting search engine indexes are: stopword removal (e.g., the removal of prepositions and articles); and stemming, i.e., reducing words to their root (stem) form. These techniques have an impact on the effectiveness Heuristic Component (HC). As the HC searches for high-coverage queries, if stopwords are present in the index, they should be included in the queries since they are likely to appear in many documents in the database. On the other hand, if they have been removed from the index, they should be ignored, since they will lead to queries that have no results. Stemmed words can also be exploited to help the HC. Consider, for example, the stemmed word "realiz". This word covers all documents in the database containing words that can be reduced to it, such as: realize, realizes, realized, realization, etc.

To detect the stopword feature, the AC issues probing queries with stopwords (e.g., "the", "a") and checks whether an actual result page or an error page is returned. For stemming, a similar approach is used: the interface is probed using stemmed words. The procedure to identify error pages follows the simple heuristic: issue queries using *dummy* words (e.g., `ewrwdwewdwddasd`) and record the results in an error template. As queries are issued, their results can be checked against the error template.

The Heuristic Component (HC). The Heuristic Component is responsible for defining a policy for submitting queries. The HC first builds a sample of the database by issuing a set of queries (*Sampling phase*). Next, it selects the most frequent words in the documents in this sample to crawl the database (*Crawling phase*), assuming they also have a high frequency in the actual database/index.

The goal of the Sampling phase is to assemble a representative sample of the database. The performance of Siphon++ heavily depends on the quality of the sample. The only way to acquire this information is through the keyword search interface that serves as the entry point to the database. To initiate this process, a set of

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
ACM 978-1-59593-991-3/08/10.

words from the entry page of the the Web site that contains the database is obtained and issued to the interface. Once a word from this set returns a valid result page containing links to target documents, the HC populates the sample with the top-k target documents, i.e., the k documents considered as most relevant for this word. We select the first keyword from the entry page because it is very likely that words in this page are presented in the database, assuming both (database entry page and database) are on the same topic. In the next step, the HC randomly selects a word from this sample and submits it. The top-k target documents are again added to the sample. This iterative process goes on, randomly probing selected words from the sample and adding their result pages to the sample, until the sample *converges*. These queries that contribute to build the sample are called sampling queries (see Figure 1).

The Crawling phase is responsible for (1) issuing queries to the database; (2) retrieving the result pages and extracting from them the links to the target documents; and (3) downloading the documents from the database.

The first step in this phase is to select the most promising terms from the sample to submit, i.e., the ones that are likely to result in high coverage. We call them *crawling terms* (Figure 1) and we store them in a *crawling term list*. This is an ordered list of words, such that first words will be submitted first. In the final step of the crawling phase, the crawler downloads the target documents, whose links were obtained from the result pages, and stores them as illustrated in Figure 1.

3. EVALUATION AND DISCUSSION

In this section, we report the results of an experimental evaluation of *Siphon++*. Our goal is to verify the effectiveness (coverage) of our method.

Dataset. We evaluated our crawler over a simulated search engine. To populate the simulated search engine, we used the TREC 2001 Web Track data (WT10g data set). We created an index over 306,210 pages using Lucene.[1] In the remainder of this section, we refer to this collection as WT10g server.

Index Configurations. To verify how our crawling strategy is affected by different indexing features commonly adopted in hidden-Web sources, we built two distinct indexes over the WT10g server varying the indexing features:

*Stemming and Stopword Removal (*StemStopRemoval*)*: the words are stemmed and stopwords removed. This is the scenario whereby the index is more compact;

*No Stemming and Stopword Removal (*NoStemStop*):* the words are not stemmed and stopwords are not removed. This represents a scenario where the index is not compacted.

Crawling Strategies. We evaluate following crawling strategies:

Baseline: This corresponds to the strategy proposed by Ntoulas et al. [2]. Their crawler selects the next query to issue based on the previously crawled pages. In contrast to our approach, it neither builds a database sample nor takes advantage of query interfaces that support disjunctive queries;

Siphoning: This configuration corresponds to our previously proposed crawling strategy, the Hidden-Web Siphon [1]. It detects stopwords but it does not detect stemming in the index;

Siphon++: This configuration represents our proposed method.

Figure 2 shows the coverage results for the different crawling strategies for a compacted index (i.e., *StemStopRemoval*). *Siphon++* obtains a higher coverage value for the early iterations. For example, at 20 terms, *Siphon++* reaches over 90% coverage, whereas Baseline is at around 70%. This indicates that adapting to the index features leads to a more efficient crawl. As depicted in Figure 3,

[1] http://lucene.apache.org/java/docs/index.html

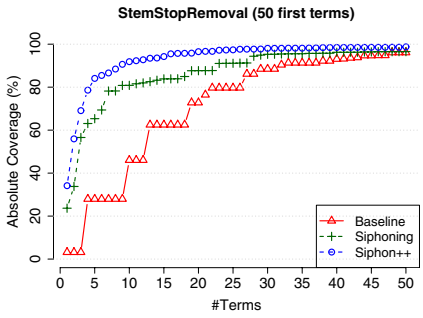

Figure 2: Coverage of the values obtained by the crawling strategies according in the StemStopRemoval scenario.

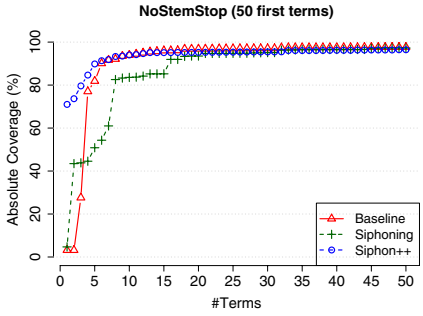

Figure 3: Coverage of the values obtained by the crawling strategies according in the NoStemStop scenario.

in the scenario where the index feature detection it is not needed (i.e., *NoStemStop*), as expected, *Siphon++* obtained a performance comparable to the Baseline. The better performance obtained by *Siphon++* for the firt queries is due to the fact that *Siphon++* builds a sample of the index before the crawl starts, whereas Baseline that builds its sample as pages are crawled.

These preliminary results are promising and show that adapting the crawling strategy to index features is beneficial. In future work, besides carrying out additional experiments using real Web sites, we plan to investigate the feasibility and effectiveness of adapting the crawler to the capabilities of the search interfaces.

Acknowledgments. This work is partially supported by the National Science Foundation (grants IIS-0713637, CNS-0751152, IIS-0746500, IIS-0513692, CNS-0514485, IIS-0534628, CNS- 0528201). This work was done while Karane Vieira was at the University of Utah.

4. REFERENCES

[1] Luciano Barbosa and Juliana Freire. Siphoning Hidden-Web Data through Keyword-Based Interfaces. In *SBBD*, pages 309–321, 2004.

[2] Alexandros Ntoulas, Petros Zerfos, and Junghoo Cho. Downloading textual hidden web content through keyword queries. In *JCDL*, pages 100–109, 2005.