Exploration of Deep Web Repositories

Nan Zhang
George Washington University
nzhang10@gwu.edu

Gautam Das
University of Texas at Arlington
gdas@uta.edu

ABSTRACT

With the proliferation of online repositories (e.g., databases or document corpora) hidden behind proprietary web interfaces, e.g., keyword-/form-based search and hierarchical/graph-based browsing interfaces, efficient ways of exploring contents in such hidden repositories are of increasing importance.

There are two key challenges: one on the proper understanding of interfaces, and the other on the efficient exploration, e.g., crawling, sampling and analytical processing, of very large repositories. In this tutorial, we focus on the fundamental developments in the field, including web interface understanding, crawling, sampling, and data analytics over web repositories with various types of interfaces and containing structured or unstructured data. Our goal is to encourage audience to initiate their own research in these exciting areas.

OUTLINE OF TUTORIAL

The following is an outline of topics that shall be covered.

1. INTRODUCTION

The tutorial shall begin with a series of real-world examples of deep web repositories hidden behind web interfaces (see Figure 1 for a typical architecture). Specifically, a repository with structured data is Yahoo! Autos (*http://autos. yahoo.com*), while an unstructured one is the document corpus of Wikipedia.

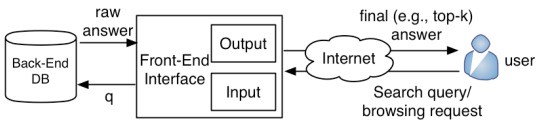

Figure 1. System Architecture.

We shall then use these examples to motivate the importance of efficient exploration over hidden web repositories. In particular, we shall show that many repositories only support a very restrictive set of search queries. To provide full (SQL) search support, one may need to *crawl* all elements from a repository and then execute the search locally. We shall also discuss the need of mining over hidden repositories. To support mining without incurring as many web accesses as crawling, one needs the ability to efficiently perform *sampling* and *analytical processing* over a hidden repository. We shall note that this tutorial focuses on deep web repositories with given URLs. Resource discovery - i.e., how to find URLs of deep web repositories (e.g., for a given topic) - is an orthogonal problem.

Taxonomy of Web Interfaces: We shall describe four types of interfaces commonly present for web repositories: *keyword search* (e.g., Google), *form-like search* (e.g., Yahoo! Autos), *hierarchical browsing* (e.g., Amazon's drop-down menu for product browsing), and *graph-based browsing* (e.g., Wikipedia).

Exploration Tasks: We shall describe three important tasks commonly desired for the deep web: crawling, sampling, and data analytics (e.g., the efficient processing of aggregate queries). We shall argue that while samples may also support aggregate (e.g., AVG) estimation, performing data analytics directly may be more efficient as its design can be made aligned with the specific aggregates to be estimated. On the other hand, sampling is more "versatile", as a collected sample may later support analytical tasks not yet known at the time of sampling.

2. CHALLENGES

Our tutorial shall next discuss why the three tasks outlined above are difficult to accomplish over deep web repositories. We summarize two key challenges, one on understanding the interface - e.g., how to model web query interfaces and perform schema matching - and the other on the efficient exploration of data - e.g., how to determine which queries/browsing requests to issue, especially given the extremely restrictive input and output interfaces of a hidden web repository. We devote the rest of our tutorial to addressing the second (i.e., exploration) challenge. For the first one, we shall briefly review it and point audience to recent tutorials covering the topic.

3. CRAWLING

In this part of the tutorial, our focus is on discussing the crawling of a deep web repository *after* its interface is properly understood. We shall start with illustrating the motivations for crawling, and then discuss existing crawling techniques for repositories with search and browsing interfaces, respectively.

Search Interfaces: We shall identify two main prerequisites for efficient crawling over search interfaces: One is how to generate "legitimate" values for populating into input fields (e.g., query phrases as keywords). The other is how to input values such that each combination returns a large number of distinct elements. Since solutions to both depend upon the repository's content, most existing techniques feature a bootstrapping process which starts with a small number of probing search queries, then uses the returned results to refine the selection of input keywords or attribute value combinations to quickly achieve high coverage.

Proceedings of the VLDB Endowment, Vol. 4, No. 12

Browsing Interfaces: The problem of crawling here is often reduced to the traversal of vertices in a tree (for hierarchical browsing) or a graph (for graph-based browsing). The common technique is breadth-first search (aka *snowball* method). While the technique itself is relatively straightforward, we shall point out to the audience that the main challenge is the *comprehensiveness* of crawling, as the graph is not necessarily connected. We shall discuss techniques used by existing crawlers to address this issue.

We shall conclude this part of the tutorial with discussions of the system-related issues (e.g., using a cluster of machines for crawling) that apply to both types of interfaces.

4. SAMPLING

In this portion, we discuss sampling techniques which aim to draw representative elements (e.g., documents, tuples) from an online repository while minimizing the number of web accesses. We shall start the discussion with motivating applications for sampling, and then review existing techniques for keyword search, form-like/hierarchical browsing, and graph browsing interfaces, respectively

Keyword-Search Interface: We begin by showing that a key problem facing the "sampling" process in many existing techniques is that the returned elements have an unknown but often significant *skew*, i.e., certain elements are sampled with much higher probability than others. We shall then discuss a skew-correction technique through *rejection sampling*.

Form-like Search or Hierarchical Browsing Interface: Skew reduction remains a challenge here. In particular, the main source of skew is the *scoring function* used by the interface to determine which top-k elements to return. We shall discuss two ideas of skew removal: One is to avoid the influence of scoring function by finding queries that return $<k$ elements. The other idea assigns a one-to-many mapping from queries to elements in the repository, such that even if a highly scored tuple is returned by more queries, it can only be sampled from one.

Graph Browsing Interface: We shall describe two types of existing techniques for sampling over a graph browsing interface: (1) the early work which uses BFS/snowball sampling to produce sample elements with an unknown skew; and (2) the random walk based techniques which has roots in the theory of finite Markov chains to produce known (and thus removable) skew over connected graphs.

5. DATA ANALYTICS

We shall now discuss analytics techniques for online repositories. We shall first argue that the key enabler for data analytics is the ability to approximately answer aggregate queries over an online repository, and then describe a few motivating examples of aggregate queries. After that, we shall discuss *bias* and *variance*, two complementary measures for the accuracy of aggregate estimations, and then review the existing techniques for the three types of interfaces, respectively.

Keyword-Search Interfaces: We shall focus on two types of data analytics techniques over keyword search interfaces. One is a two-step process which first calls upon the above-discussed sampling techniques to produce sample elements, and then use the sample to extract aggregate information for analytics. The other type of technique directly estimates aggregates without the middle step of sample generation. A key advantage here is that unlike in the sampling case where many retrieved elements may have to be rejected for skew removal, all retrieved elements may be used, albeit in a weighted fashion, for aggregate estimations.

Form-like Search or Hierarchical Browsing Interfaces: We shall first demonstrate that a direct estimation of aggregates over form-like or hierarchical browsing interfaces avoids the costly process of rejecting elements for eliminating sample skew. Then, we shall explain why SUM and COUNT queries can be easily estimated without bias, while doing so for AVG queries is extremely difficult if not impossible. After that, we focus on *variance-reduction* techniques for improving estimation accuracy. Before concluding this part, we shall briefly discuss a few recent works which have the exact opposite objective – i.e., to *prevent* aggregate queries from being estimated (accurately) through a form-like interface, in order to protect the privacy of aggregate information for repository owners.

Graph Browsing Interfaces: We shall start by arguing that data analytics over graph browsing interfaces is closely related to the problem of graph testing, as the latter assumes an access cost to learning whether an edge exists in the graph, resembling the web access cost for a graph browsing interface, and aims to learn certain (aggregate) information of the graph while minimizing the access cost. Nonetheless, we shall argue that the cost models of real-world interfaces are much more diverse than what have been studied in graph testing, leading to vastly different solutions and calling for further research on the cost models. We shall then discuss the existing work for aggregate estimation using random walks, random BFS, etc.

6. CONCLUSIONS

We shall summarize how the challenging problems of crawling, sampling and analytics over hidden web repositories require expertise in traditional query processing, IR, social networks, data mining as well as algorithms. We shall conclude by identifying open challenges.

TARGET AUDIENCE

The anticipated audience will consist of database, WWW, IR, data mining, and algorithms researchers, web developers, as well as information systems designers. No specific prerequisite other than general knowledge of databases is required.

BIOGRAPHICAL SKETCH

Nan Zhang is an Assistant Professor of Computer Science at the George Washington University, Washington, DC. His research on databases, data mining and information privacy is supported by NSF, including an NSF CAREER award in 2008.

Gautam Das is a Professor and Head of the Database Exploration Laboratory (DBXLAB) at the CSE department of the University of Texas at Arlington. He graduated with a PhD in computer science from the University of Wisconsin, Madison. Dr. Das's research interests span data mining, information retrieval, databases, algorithms and computational geometry.

ACKNOWLEDGEMENT

Nan Zhang is partially supported by NSF grants 0852674, 0915834 and a GWU Research Enhancement Fund. Gautam Das is partially supported by NSF grants 0812601, 0915834 and grants from Microsoft Research and Nokia Research.