Answering Aggregation Queries on Hierarchical Web Sites Using Adaptive Sampling*

Foto N. Afrati
Computer Science Division
NTUA, Athens, Greece
afrati@softlab.ece.ntua.gr

Paraskevas V. Lekeas
Computer Science Division
NTUA, Athens, Greece
plekeas@mail.ntua.gr

Chen Li
Dept. of Computer Science
UC Irvine, CA 92697, USA
chenli@ics.uci.edu

ABSTRACT

We study how to answer aggregation queries over hierarchical Web sites using adaptive sampling.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Retrieval models.

General Terms: Algorithms.

Keywords: Hierarchical Web Sites, Aggregation Queries, Adaptive Sampling.

1. MOTIVATION

Many e-commerce sites and portals on the Web publish their data in HTML pages organized as hierarchies. For instance, Amazon.com has many pages about its books. It classifies these books as different hierarchical classes. Figure 1 shows part of the hierarchy. Each internal node in the hierarchy represents a class of books, such as `Accessories` or `History`. This node corresponds to a Web page that displays the categories (Web links) of its child classes. A leaf node contains a collection of books belonging to the corresponding category, such as `Warehousing` or `Encryption`. These books are published as different pages. The main reason of having such a hierarchy is to allow easy browsing by Web users, who can browse the pages starting from the root to locate the books in a class.

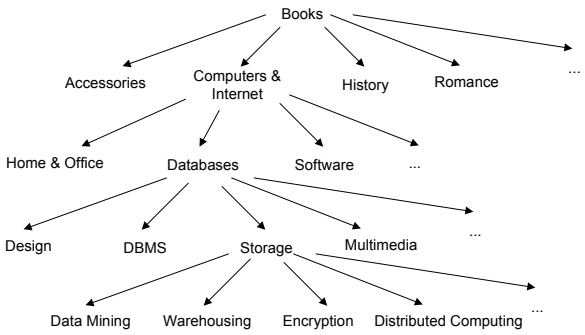

Figure 1: Book hierarchy at Amazon.com.

*Foto N. Afrati and Paraskevas V. Lekeas were supported by the project PYTHAGORAS I, co-funded by the European Social Fund (75%) and National Resources (25%).

Copyright is held by the author/owner.
CIKM'05, October 31–November 5, 2005, Bremen, Germany.
ACM 1-59593-140-6/05/0010.

Many applications built on these Web sites need to ask queries on the data. Consider, for example, an online book store that competes with Amazon.com. In order to make marketing strategies about its own books, it wants to collect the information about the prices of different classes of books at Amazon.com. The following are a few example queries the company would issue about Amazon.com books.

- Query Q_1: Find the average price of books on `Warehousing`. These books correspond to a class of a leaf node.
- Query Q_2: Find the maximal price of books on `Databases`. These books correspond to a class of an internal node in the hierarchy.

We could answer a query by accessing all the book pages relevant to the query. When there are many such books, this method requires to access a large number of pages, causing a significant workload on the Web site. The problem becomes even worse when we need to ask such queries frequently, or when the data at the Web site is very dynamic.

2. METHODOLOGY

In this paper we study how to answer aggregation queries on Web hierarchies by sampling. We focus on the following three challenges related to how to allocate Web-access resources to the pages of different classes, illustrated by the book example. (1) The books from different classes have different price distributions. For instance, the number of books and average price of the `Warehousing` class could be very different from those of the `Encryption` and `Distributed Computing` classes. As of October 2004, the numbers of books in these three categories were 122, 132, and 202, respectively. Their average prices were $336, $78, and $91, respectively. A naive, random-sampling approach is not ideal. (2) The distributions of the objects in different categories are not known a-priori. (3) We need to estimate the quality of the answers computed using sampling.

2.1 Formulation

Hierarchical Web-Site Structure and Access Model: Consider a Web site that publishes its information about *objects* such as books, electronics, and computers. The information about an object is published in a single page, and one Web access is needed to retrieve this page. The site organizes these pages as a hierarchical tree [2, 3], where each leaf node corresponds to a class of objects. For example, in Figure 1, the leaf node `Data Mining` has a list of links to the pages of data mining books. (This list could be published in multiple pages for browsing purposes.) The size of a leaf node is the number of all its objects. An internal node in the

hierarchy corresponds to a list of links pointing to its children (internal nodes or leaf nodes). The size of an internal node is the sum of the sizes of all its descendant leaf nodes. We are given the number of objects in each leaf node.

Queries: We consider aggregation queries using functions AVG, MAX, and MIN. An aggregation query on a node concerns the objects represented by this node and the aggregation is applied on these objects. We call this node the *query node* for the given query. A query node can be a leaf node or an internal node. For example, in Figure 1, if we ask for the average price of all the `Encryption` books, then the query node is a leaf node. If we ask for the maximum price of the `Databases` books, then the query node is an internal node, and the actual data is stored in its descendant leaf nodes such as `DBMS`, `Data Mining`, `Encryption`, etc. Formally, an aggregation query is represented as $\gamma(v)$, in which γ is an aggregation function (AVG, MAX, or MIN) and v is a node in the hierarchy. For instance, the query $MAX($`Databases`$)$ asks for the maximum price of all the books in the `Databases` class. Let $ans(\gamma(v))$ denote the real answer to this query.

Approximate Answers with Quality Guarantees: Given a query $\gamma(v)$, we want to compute an approximate answer to the query using samples from the objects in the classes relevant to the query, i.e., those objects in the leaf nodes of the node v. The query also comes with a confidence δ, which is a real between 0 and 1. Our goal is to find an estimation e of the real answer to the query, such that with probability at least $1 - \delta$ the "distance" between the estimated answer and the real answer is as small as possible. The "distance" measurement depends on the aggregation function γ. For instance, in the case of AVG, the distance is $|e - ans(\gamma(v))|$. For the case of MAX and MIN, we use quantiles to measure such a distance.

2.2 Adaptive Sampling

We view the data in the class of each leaf node as a random variable that takes values uniformly at random from the objects in this class. We assume that Web-access resources are sparse and expensive. We focus on how to make the best use of the available resources to improve the quality of the approximate answer. Specifically, one problem we are investigating is: Given a number of resources and a confidence, what is the best way to sample the objects to answer the query so that we obtain a good approximation of the real answer with the given confidence? Our solutions can also be used to solve other related problems, such as deciding how many resources are needed when the user also specifies a threshold on the distance between the estimate answer and the real answer.

As we do not know the distributions of the objects in different leaf nodes a priori, we use *adaptive sampling*. The main idea is to allocate resources to these leaves proportionally to a measure that represents the importance of the particular leaf in finding a good answer to the query. The measure also represents a parameter of the distribution of the data in the particular leaf. For instance, if the standard deviation of the objects in a particular leaf is small, then we do not need to allocate a large number of resources to this node, because we can obtain a good estimation with a few resources. On the other hand, if the standard deviation is large, we need to allocate more resources to this leaf node in order to achieve a good approximation.

In [1] we propose a family of adaptive sampling algorithms for different aggregation functions. Each algorithm allocates the available resources iteratively. In each stage, it assigns resources to those leaves that introduce more error in the estimated answer. The algorithm adaptively decides how many resources need to be allocated to a leaf node in the next iteration, until we use all the resources, or, alternatively, when the user is satisfied with the quality of the approximate answer. Specifically, (1) we propose adaptive-sampling algorithms for answering a single AVG/MAX/MIN query, and (2) we propose an algorithm for answering a set of aggregation queries.

3. EXPERIMENTS

We have conducted experiments to evaluate the proposed algorithms for both a real data set and a synthetic data set. We collected the real data about book prices from Amazon.com from March 2004 to July 2004. We implemented a crawler to download all the "Buy New" price tags of different book categories. For the synthetic data set, we used Mathematica to produce leaf nodes with the same population and standard deviation as the leaf nodes in the real data set, but with a normal distribution.

For instance, we evaluated our adaptive-sampling algorithm on an internal node with two leaf nodes of the Amazon data set. The first node had 3396 books, with an average price 48, and a standard deviation 58.38. The second node had 2157 books, with an average 21, and a standard deviation 27.92. The average of the internal node was 38. The total number of resources was 1000, and in each iteration we assigned $n = 200$ resources. We also implemented a naive algorithm that allocates the same number of resources to each leaf. Table 1 shows the results of the two approaches in different runs. The adaptive approach gave more accurate estimations of the average. Even though the naive approach gave estimations with a small confidence interval, their estimations were far from the real average value.

Run	Estimated (Adaptive)	Estimated (Naive)
1	35 ± 4	31 ± 2
2	37 ± 4	30 ± 2
3	36 ± 5	29 ± 2
4	37 ± 4	30 ± 2
5	36 ± 6	31 ± 3

Table 1: Answering an AVG query on an internal node (real data). The real average is 38.

In [1] we present more experimental results, including different algorithms on answering AVG/MAX/MIN queries on leaf nodes and internal nodes, and an algorithm for answering multiple queries. Our results show the advantages of our adaptive sampling approach compared to the naive, random-sampling approach.

4. REFERENCES

[1] F. Afrati, P. Lekeas, and C. Li. Answering aggregation queries on hierarchical web sites using adaptive sampling (full version). Technical report, Department of Computer Science, UC Irvine, http://www.ics.uci.edu/∼chenli/pubs.html, 2002.

[2] J. Czyzowicz et al. Evaluation of hotlink assignment heuristics for improving web access. *Computing (IC' 01)*, 2:793–799, June 2001.

[3] B. Heeringa and M. Adler. Optimal website design with the constrained subtree selection problem. In *ICALP*, pages 436–447, 2004.