Frequency-Based Coverage Statistics Mining for Data Integration

Zaiqing Nie & Subbarao Kambhampati

Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406
Email: {nie, rao}@asu.edu

Abstract

Recent work in data integration has shown the importance of statistical information about the coverage and overlap of data sources for efficient query processing. Gathering and storing the required statistics presents many challenges, not the least of which is controlling the amount of statistics learned. In this paper we describe a novel application of data mining technology for statistics gathering. Specifically, we introduce a statistics mining approach which efficiently discovers frequently accessed query classes, and learns and stores statistics only with respect to these classes. We describe the details of our method, and present experimental results demonstrating the efficiency and effectiveness of our approach.

1 Introduction

With the vast number of autonomous information sources available on the Internet today, users have access to a large variety of data sources. Data integration systems [LRO96, ACPS96, LKG99, PL00] are being developed to provide a uniform interface to a multitude of information sources, query the relevant sources automatically and restructure the information from different sources. In a data integration scenario, a user interacts with a mediator system via a mediated schema. A mediated schema is a set of virtual relations, which are effectively stored across multiple and potentially overlapping data sources, each of which only contain a partial extension of the relation. Query optimization in data integration [FKL97, NLF99, NK01, DH02] thus requires the ability to figure out what sources are most relevant to the given query, and in what order those sources should be accessed. For this purpose, the query optimizer needs to access statistics about the coverage of the individual sources with respect to the given query, as well as the degree to which the answers they export overlap. We illustrate the need for these statistics with an example.

Motivating Example: We have been developing *BibFinder* (Figure 1, http://rakaposhi.eas.asu.edu/bibfinder), a publicly available computer science bibliography mediator. *BibFinder* integrates several online Computer Science bibliography sources. It currently covers *CSB, DBLP, Network Bibliography, ACM Digital Library, ScienceDirect,* and *CiteSeer.* Plans are underway to add
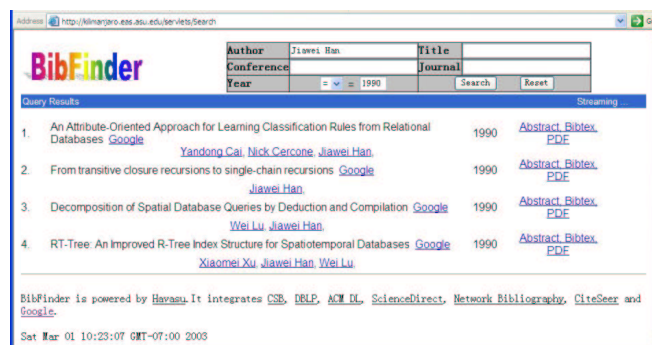

Figure 1: *The BibFinder User Interface*

several additional sources including *IEEE Xplore* and *Computational Geometry Bibliography*.

The sources integrated by *BibFinder* are autonomous and partially overlapping. By combining the sources, *BibFinder* can present a unified and more complete view to the user. However it also brings some interesting optimization challenges. Let us assume that the global schema exported by *BibFinder* includes just the relation: **paper(title, author, conference/journal, year)**. Each of the individual sources only export a subset of the global relation. For example, *Network Bibliography* only contains publications in Networks, *DBLP* gives more emphasis on Database related publications, while *ScienceDirect* has only archival journal publications etc. To efficiently answer users' queries, we need to find and access the most relevant subset of the sources for the given query. Suppose, the user asks a selection query:

Q(title,author) :− **paper**(title, author, conference/journal, year),
conference="AAAI".

To answer this query efficiently, *BibFinder* needs to know the *coverage* of each source S with respect to the query Q, i.e. $P(S|Q)$, the probability that a random answer tuple for query Q belongs to source S. Given this information, we can rank all the sources in descending order of $P(S|Q)$. The first source in the ranking is the one we want to access first while answering query Q. Since the sources may be highly correlated, after we access the source S' with the maximum coverage $P(S'|Q)$, we need to access as the second source, the source S'' that has the highest *residual coverage* (i.e., provides the maximum number of those answers that are not provided by the first source S'). Specifically we need to pick the source S'' that has next best rank in terms of coverage

but has minimal *overlap* (common tuples) with S'. If we have the coverage and overlap statistics for every possible query, we can get the complete order in which to access the sources. However it will be very costly to remember and learn statistics w.r.t. every source-query combination, and overlap information about every subset of sources with respect to every possible query!

Fortunately, the users' interests may only focus on a small part of the space of all the possible queries. For example, in our *BibFinder* scenario, the users are much more interested in asking queries to find papers in some well known conferences such as Sigmod and VLDB than some other lesser known conferences, even through the lesser known conferences may have a larger number papers stored in some sources. Even if a mediator cannot provide accurate statistics for every possible query, it can still achieve a reasonable average accuracy by keeping more accurate coverage and overlap statistics for queries that are asked more frequently, and less accurate statistics for infrequent queries. □

In this paper, we introduce *StatMiner*, a statistics mining module for web based data integration. *StatMiner* comprises of a set of connected techniques that estimate the coverage and overlap statistics while keeping the amount of needed statistics tightly under control. Since the number of potential user queries can be quite high, *StatMiner* aims to learn the required statistics for *query classes* i.e. groups of queries. Specifically, *StatMiner* will group queries into classes and efficiently discover frequent query classes. For each discovered frequent class, our approach probes sources using the most frequently accessed queries within the class and learns class-source association rules w.r.t. to the probing results. An example class-source association rule could be: $SIGMOD \rightarrow DBLP$ with confidence 100%, which means information source $DBLP$ covers all the paper information for $SIGMOD$ related queries. The statistics for infrequently accessed query classes will not be learned. Instead we use the statistics of a more general abstract query class which has the total frequency more than the threshold to estimate the coverage and overlap of the queries in the infrequent classes. In this way, we will provide more accurate statistics for frequently accessed queries. Our objective is to keep the number of statistics low enough while still providing high average accuracy for users' future queries. Here we assume that queries asked by the users in future will have the same distribution as the past queries. If the assumption holds, then the average accuracy of the mediator's coverage and overlap estimation will be higher if we provide more accurate statistics for more frequently asked queries.

The rest of the paper is organized as follows. In the next section, we give an overview of our approach and define the needed terminology. This is followed by a detailed description of our experimental setup and the results we obtained demonstrating the efficiency of our learning algorithms and the effectiveness of the learned statistics. Next, we discuss the related work and several important potential extensions to the basic framework. Finally we conclude with a summary of our contributions.

2 Overview

In order to better illustrate the novel aspects of our association rule mining approach, we purposely limit the queries to just projection and selection queries.
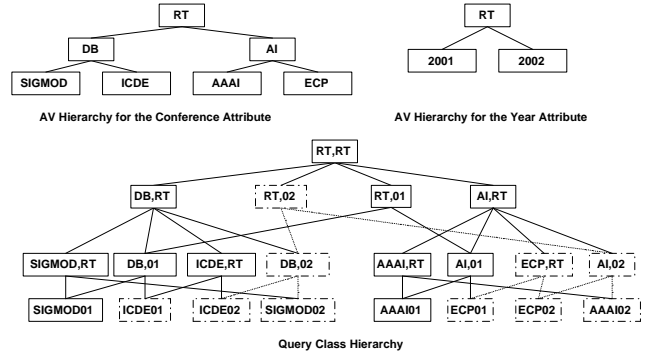

Figure 2: *AV Hierarchies and the Corresponding Query Class Hierarhcy*

2.1 Grouping Queries into Classes

AV Hierarchy: Since we are considering selection queries, we can classify the queries in terms of the selected attributes and their values. To abstract the classes further we assume that the mediator has access to the so-called "attribute value hierarchies" for a subset of the attributes of each mediated relation. An *AV hierarchy* (or attribute value hierarchy) over an attribute A is a hierarchical classification of the values of the attribute A. The leaf nodes of the hierarchy correspond to specific concrete values of A, while the non-leaf nodes are abstract values that correspond to the union of values below them. Figure 2 shows two very simple AV hierarchies for the "conference" and "year" attributes of the "paper" relation. Note that hierarchies do not have to exist for every attribute, but rather only for those attributes over which queries are classified. We call these attributes the **classificatory attributes**. We can choose as the classificatory attributes the best k attributes whose values differentiate the sources the most, where the number k is decided based on a tradeoff between prediction performance versus computational complexity of learning the statistics by using these k attributes. The selection of the classificatory attributes may either be done by the mediator designer or using automated techniques. The AV hierarchies themselves can either be hand-coded by the designer, or can be learned automatically.

Query Classes Since we focus on selection queries, a typical query will have values of some set of attributes bound. We group such queries into query classes using the AV hierarchies of the classificatory attributes. A query **feature** is defined as the assignment of a classificatory attribute to a specific value from its AV hierarchy. A feature is "abstract" if the attribute is assigned an abstract (non-leaf) value from its AV hierarchy. Sets of features are used to define query classes. Specifically, query class is a set of (selection) queries that all share a particular set of features. The space of query classes is just the cartesian product of the AV hierarchies of all the classificatory attributes. Specifically, let H_i be the set of features derived from the AV hierarchy of the ith classificatory attribute. Then the set of all query classes (called $classSet$) is simply $H_1 \times H_2 \times ... \times Hn$. The AV hierarchies induce subsumption relations among the query classes. A class C_i is subsumed by class C_j if every feature in C_i is equal to, or a specialization of, the same dimension feature in C_j. A query Q belongs to a class C if the values of the classificatory attributes in Q are equal to or are specializations of the features defining

Conference	Year	Frequency
SIGMOD	2001	150
ICDE	2001	50
ICDE	NULL	10
AAAI	2001	90

Table 1: *Tuples in the table QList*

C. Figure 2 shows an example class hierarchy for a very simple mediator with the two example AV hierarchies. The query classes are shown at the bottom, along with the subsumption relations between the classes.

2.2 Class Access Probability

As we discussed earlier, it may be prohibitively expensive to learn and keep in memory the coverage and overlap statistics for every possible query class. In order to keep the number of association rules low, we would like to prune query classes which are rarely used. We use a threshold on the support of a class (i.e., percentage of total frequency of queries that use that class's statistics), called *minfreq*, to identify frequent access query classes. Coverage and overlap statistics are learned only with respect to these frequent classes.

We assume the mediator maintains a query list *QList*, which keeps track of the user queries and their access frequency. In table 1, we show an example query list. We use FR_Q to denote the access frequency of a query Q, and FR to denote the total frequency of the all the queries in QList. The *query probability* of a query Q, denoted by $P(Q)$, is the probability that a random query posed to the mediator is the query Q. It can be computed using the formula: $P(Q) = \frac{FR_Q}{FR}$. The *class probability* of a class C, denoted by $P(C)$, is the probability that a random query posed to the mediator is subsumed by the class C. It can be computed using the following formula:

$$P(C) = \sum_{Q \in C} P(Q)$$

The class probability of a class is solely dependent on the total frequency of the all the queries belonging to the class. We use the term *candidate frequent class* to denote any class with class probability more than the minimum frequency threshold *minfreq*. The example classes shown in Figure 2 with solid frame lines are candidate frequent classes. As we can see some queries may have multiple lowest level ancestor classes which are candidate frequent classes and not subsumed by each other. For example, the query (or class) (ICDE,01) has both the class (DB,01) and class (ICDE,RT) as it's parent class. For a query with multiple ancestor classes, we need to map[1] the query into a single ancestor class with lowest class probability, whose statistics will be more relevant to the query.

The *class access probability* of a class C, denoted by $P_{map}(C)$, is the probability that a random query posed to the

---

[1]Mapping the queries into query classes is straightforward for queries binding classificatory attributes. In general (e.g. for queries that bind non-classificatory attributes), mapping becomes an instance of classification learning problem [HK00].

mediator is actually mapped to the class C. It can be computed using the following formula:

$$P_{map}(C) = \sum_{Q \text{ is mapped to } C} P(Q)$$

Since the classes may overlap in terms of queries they subsume, the the summation over $P(C)$ for all class C will add to greater than 1. However since each query is mapped into only one class, the summation over all $P_{map}(C)$ will add to 1.

2.3 Coverage and Overlap w.r.t Query Classes

The *coverage* of a data source S with respect to a query Q, denoted by $P(S|Q)$, is the probability that a random answer tuple of query Q is present in source S. The *overlap* among a set $\widehat{S}$ of sources with respect to a query Q, denoted by $P(\widehat{S}|Q)$, is the probability that a random answer tuple of the query Q is present in each source $S \in \widehat{S}$. The overlap (or coverage when $\widehat{S}$ is a singleton) statistics w.r.t. a query Q be computed using the following formula

$$P(\widehat{S}|Q) = \frac{N_Q(\widehat{S})}{N_Q}$$

Here $N_Q(\widehat{S})$ is the number of common answer tuples for Q that are from $\widehat{S}$, N_Q is the total number of answer tuples for Q. We assume that the union of the contents of the available sources within the system covers 100% of the answers of the query. In other words, coverage and overlap is measured relative to the available sources.

The *coverage* of a source S w.r.t. a class C, denoted by $P(S|C)$, is the probability that a random answer tuple of a random query belonging to the class C is present in source S. The *overlap* among a set $\widehat{S}$ of sources with respect to a class C, denoted by $P(\widehat{S}|C)$, is the probability that a random answer tuple of a random query belonging to the class C is present in each source $S \in \widehat{S}$. The overlap (or coverage when $\widehat{S}$ is a singleton) statistics w.r.t. a query class C can be computed using the following formula:

$$P(\widehat{S}|C) = \frac{P(C \cap \widehat{S})}{P(C)} = \frac{\sum_{Q \in C} P(\widehat{S}|Q)P(Q)}{P(C)}$$

The coverage and overlap statistics w.r.t. a class C is used to estimate the source coverage and overlap for all the queries that are mapped into C. These coverage and overlap statistics can be conveniently computed using an association rule mining approach.

2.4 Mining Class-Source Association Rules

In order to define the term class-source association rule, we first define the term *source set*. Let $\tau_s = \{S_i, S_2, ..., S_m\}$ be a set of all the sources available to a mediator. A subset of τ_s is referred to as a source set.

A *class-source association rule* represents strong associations between a query class and a source set. Specifically, we are interested in the association rules of the form $C \rightarrow \widehat{S}$, where C is a query class, and $\widehat{S}$ is a source set (possibly singleton). The *support* of the class C (denoted by $P(C)$) refers to the class probability of the class C, and the overlap (or coverage when $\widehat{S}$ is

a singleton) statistic $P(\widehat{S}|C)$ is simply the *confidence* of such an association rule(denoted by $P(\widehat{S}|C) = \frac{P(C \cap \widehat{S})}{P(C)}$). Examples of such association rules include: $AAAI \rightarrow S_1$, $AI \rightarrow S_1$, $AI\&2001 \rightarrow S_1$ and $2001 \rightarrow S_1 \wedge S_2$.

2.5 The StatMiner Architecture

In *StatMiner*, the frequent query classes are discovered by using the DFC, an algorithm we developed to efficiently identify the query classes with sufficiently large support (for more information see [NK02]), and learn the coverage and overlap statistics using a variant of the Apriori algorithm [AS94]. The resolution of the learned statistics is controlled in an adaptive manner with the help of three thresholds. A threshold *minfreq* is used to decide whether a query class has large enough support to be remembered. When a particular query class doesn't satisfy the minimum support threshold, *StatMiner*, in effect, stores statistics only with respect to some abstraction (generalization) of that class. A threshold *minoverlap* is used to decide whether or not the overlap statistics between a set of sources and a remembered query class should be stored. Another threshold *minprobe* is used to control the minimum percentage of the total query frequency of all the queries in a class covered by the chosen probing queries to learn coverage and overlap statistics for the class.

Using DFC we then classify user queries based on AV Hierarchies and dynamically identify frequent classes for which the query access frequency is above the specified threshold *minfreq*. We learn and store statistics only with respect to these identified frequent classes (see [NK02] for more information). When the mediator, in our case *BibFinder*, encounters a new user query, it maps the query to one of the query classes for which statistics are available. Since we use thresholds to control the set of query classes for which statistics are maintained, it is possible that there is no query class that exactly matches the user query. In this case, we map the query to the nearest abstract query class that has available statistics. The loss of accuracy in statistics entailed by this step should be seen as the cost we pay for keeping the amount of stored statistics low. Once the query class corresponding to the user query is determined, the mediator uses the learned coverage and overlap statistics to rank the data sources that are most relevant to answering the query.

Although our current experiments with *StatMiner* use hand-coded hierarchies, we are extending *StatMiner* to automatically build AV hierarchies using an agglomerative hierarchical clustering algorithm (see [NK02] for more information) from the query list maintained by the mediator. The basic idea of generating an AV hierarchy is to cluster similar attribute values into classes in terms of the coverage and overlap statistics of their corresponding selection queries binding these values. Then the problem of finding similar attribute values becomes the problem of finding similar selection queries. In order to find similar queries, we define a distance function to measure the distance between a pair of selection queries $(Q1, Q2)$.

$$d(Q1, Q2) = \sqrt{\sum_i [P(\hat{S}_i|Q1) - P(\hat{S}_i|Q2)]^2}$$

Where $\hat{S}_i$ denotes the ith source set. The interpretation of the distance function is that we consider two queries similar if their source coverage and overlap statistics are similar. See [NK02] for more information.

3 Preliminary Results

We designed a simple mediator which only exports data for the paper relation (see the motivating example in Section 1). In Figure 4, we show the two AV hierarchies we use. We setup 20 data sources each of which contains data for the global relation $paper$. The data of the sources are the papers in DBLP published by computer science researchers in Database and Artificial Intelligence. The sources have different concentration of the data. For example, one source may contain only papers published in SIGMOD after 1996. Some of the sources are highly correlated for some queries. The queries can be selection query with conference and/or year attribute bound. The query list we used to discover frequent query classes was generated manually. The frequency of the queries are assigned proportionally to the statistics provided by NEC Research Index. We combine the statistics about the impact of the conferences and the most frequently accessed papers in Research Index to simulate the query frequency of our experimental system. We setup a one second delay for answering each query sent to a source to simulate the probing cost.

In order to evaluate the effectiveness of our learned statistics, we implemented the **Simple Greedy** and **Greedy Select** algorithms described in [FKL97] to generate query plans using the learned source coverage and overlap statistics. *Simple greedy* generates plans by greedily selecting top k sources ranked according to their coverages, while *Greedy select* selects sources with high residual coverages calculated using both the coverage and overlap statistics. A simple **Random Select** algorithm is also used to randomly choose k sources as the top k sources.

After we learn the statistics, we randomly issue 100 queries according to the frequency distribution of the query list to test the accuracy of these statistics. We generate plans using the learned statistics and the above algorithms. The effectiveness of the statistics is estimated according to how good the plans are. The goodness of the plan is evaluated by calling the sources in the plan and all the other sources available to the mediator. We define the precision of a plan to be the fraction of sources in the estimated plan, which turn out to be the real top k sources after we execute the query. The average precision and number of answers returned by executing the plan are used to estimate the accuracy of the learned statistics.

In Figure 3 (a), we observe the number of candidate frequent query classes and the number of frequent query classes. As we can see from the figure, as we increase the threshold *minfreq*, the number of candidate frequent classes and frequent classes will both decrease, and there is a sharp drop for the small thresholds. We also see, for almost all the minfreq thresholds, we always prune more than a half of the candidate frequent class discovered from DFC with low class access probability.

In Figure 3 (b), we observe the statistics learning time which includes the time for discovering frequent query classes, probing the sources and computing the coverage and overlap statistics. As you can see as we increase the minfreq, the total learning learning time decreases. In the experiment, we just probe the sources with
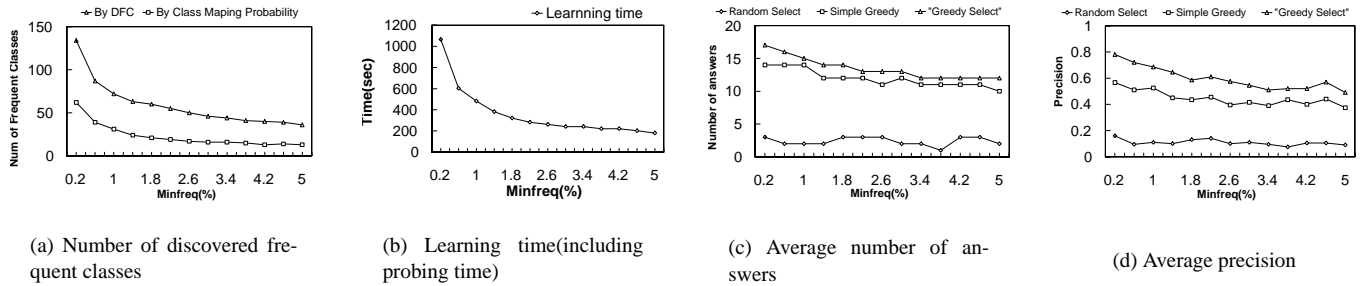
(a) Number of discovered frequent classes

(b) Learning time(including probing time)

(c) Average number of answers

(d) Average precision

Figure 3: Experiment results

AV hierarchy for the conference attribute
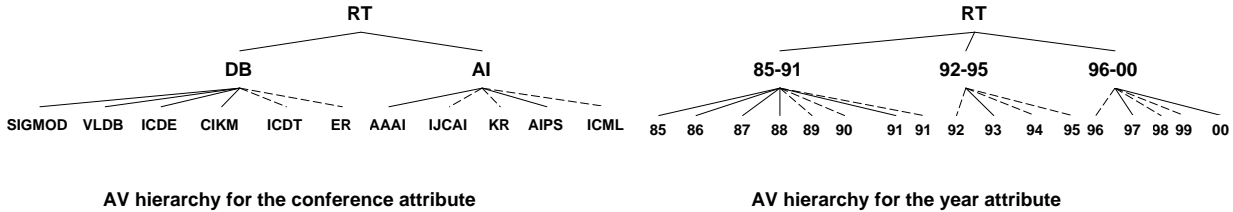
AV hierarchy for the year attribute

Figure 4: *AV Hierarchies*

a very small number of queries whose total frequency covers *minprobe*=20% of the total frequency of all the queries in the class. The threshold *minoverlap* is set to 0.5%.

In Figure 3 (c), we observe the average number of answers by executing the plans generated by the three algorithms for the 100 randomly generated queries. In Figure 3 (d), we observe the average precision of the plans. As we can see the plans generated using our learned statistics are much better both in terms of the number of answers we get and in terms of the precision of the plans for these queries than the ones generated without using any statistics.

Altogether the experiments show that our association rule mining approach can effectively control the number of statistics required by a mediator to deal with the tradeoff between the accuracy of the statistics and the cost of leaning and remembering these statistics. As we can see, the number of statistics and the learning time drop dramatically as we increase the threshold *minfreq*, while the average accuracy of the learned statistics drops smoothly.

We are currently evaluating the effectiveness of our approach by applying it to *BibFinder* , where we automatically learn AV hierarchies from it's recorded query list. The learned AV hierarchies are used by the DFC algorithm to discover frequent query classes and to map a user's query into a query class.

4   Discussion and Related Work

In order to better illustrate the novel aspects of applying association rule mining techniques to source coverage statistics gathering, we have purposely simplified some aspects of our framework. To complete the discussion, we now describe several important potential extensions to the basic framework.

In our discussion we assume that queries asked by the users in future will have the same distribution as the past queries. Since the users' interests may change over different time period, an important extension is to incrementally update the learned statistics w.r.t. the users' most recent interests. We are currently considering an incremental statistics updating approach to incrementally modify the existing class hierarchy by splitting, merging and deleting existing classes (and their respective statistics) in the class hierarchy.

In this paper, we only discussed how to learn coverage and overlap statistics of select and project queries. The techniques described in this paper can however be extended to join queries. Specifically, we consider the join queries with the same subgoal relations together. For the join queries with the same subgoal relations, we can classify them based on their bound values and use similar techniques for selection queries to learn statistics for frequent join query classes.

In this paper, we assume the mediators will maintain a query list $QList$. However the $QList$ may not be available for mediators at their beginning stages, the paper [NNVK02] introduces a size-based approach to learning statistics in such beginning scenarios. [NNVK02] assumes that query classes with more answers tuples will be accessed more frequently, and learns coverage statistics w.r.t. large query classes. Although the size-based approach can be seen as complementary to the frequency-based approach introduced in this paper, it's worth mentioning that the underlying technical details of the approaches are significantly different.

There has been some previous work on using probing techniques to learn database statistics both in multi-database literature and data integration literature. Zhu and Larson [ZL96] describe techniques for developing regression cost models for multi-database systems by selective querying. Adali et. al [ACPS96] discuss how keeping track of rudimentary access statistics can help in doing cost-based optimizations. More recently, the work by Gruser et. al. [GRZ⁺00] considers mining response time

statistics for sources in data integration scenario. Given that both coverage and response time statistics are important for query optimization (c.f. [NK01,DH02]), our work can be seen as complementary to theirs.

The utility of quantitative coverage statistics in ranking the sources is first explored by Florescu et. al. [FKL97]. The primary aim of both these efforts was however was on the "use" of coverage statistics, and they do not discuss how such coverage statistics could be learned. In contrast, our main aim in this paper is to provide a framework for learning the required statistics.

There has also been some work on ranking text databases in the context of key word queries submitted to meta-search engines. Recent work ([WMY00], [IGS01]) considers the problem of classifying text databases into a topic hierarchy. While our approach is similar to these approaches in terms of using concept hierarchies, and using probing and counting methods, it differs in several significant ways. First, the text database work uses a single topic hierarchy and does not have to deal with computation of overlap statistics. In contrast we deal with classes made up from the cartesian product of multiple AV hierarchies, and are also interested in overlap statistics. This makes the issue of space consumed by the statistics quite critical for us, necessitating our threshold-based approaches for controlling the resolution of the statistics.

5 Conclusions

In this paper we motivated the need for automatically mining the coverage and overlap statistics of sources w.r.t. frequently accessed query classes for efficient query processing in a data integration scenario. We then presented a set of connected techniques that discover frequent query classes and use a limited number of probing queries to estimate the coverage and overlap statistics for these classes. We described the details and implementation of our approach. We also presented a preliminary empirical evaluation of the effectiveness of our approach in a realistic setting. Our experiments demonstrate that (i) we can systematically trade the statistics learning time and number of statistics remembered for accuracy by varying the frequent class thresholds. (ii) The learned statistics provide tangible improvements in the source ranking, and the improvement is proportional to the type (coverage alone vs. coverage and overlap) and granularity of the learned statistics.

We are currently evaluating the effectiveness of our approach by applying it to *BibFinder* , where we automatically learn AV hierarchies from it's recorded query list. The learned AV hierarchies are used by the DFC algorithm to discover frequent query classes and to map a user's query into a query class.

References

[ACPS96] S. Adali, K. Candan, Y. Papakonstantinou, and V. S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proceedings of SIGMOD-96*, 1996.

[AS94] Rakesh Agrawal, Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *VLDB*, Santiage, Chile, 1994.

[DH02] A. Doan and A. HaLevy. Efficiently Ordering Plans for Data Integration. In *Proceedings of ICDE-2002*, 2002.

[DGL00] Oliver M. Duschka, Michael R. Genesereth, Alon Y. Levy. Recursive Query Plans for Data Integration. In *Journal of Logic Programming, Volume 43(1)*, pages 49-73, 2000.

[FKL97] D. Florescu, D. Koller, and A. Levy. Using probabilistic information in data integration. In *Proceeding of the International Conference on Very Large Data Bases* (VLDB), 1997.

[GRZ$^+$00] Jean-Robert Gruser, Louiqa Raschid, Vladimir Zadorozhny, Tao Zhan: Learning Response Time for WebSources Using Query Feedback and Application in Query Optimization. VLDB Journal 9(1): 18-37 (2000)

[HK00] Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmman Publishers, 2000.

[IGS01] P. Ipeirotis, L. Gravano, M. Sahami. Probe, Count, and Classify: Categorizing Hidden Web Dababases. In *Proceedings of SIGMOD-01*, 2001.

[LKG99] E. Lambrecht, S. Kambhampati and S. Gnanaprakasam. Optimizing recursive information gathering plans. In *Proceeding of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[LRO96] A. Levy, A. Rajaraman, J. Ordille. Query Heterogeneous Information Sources Using Source Descriptions. In *VLDB Conference*, 1996.

[NLF99] F. Naumann, U. Leser, J. Freytag. Quality-driven Integration of Heterogeneous Information Systems. In *VLDB Conference* 1999.

[NK01] Z. Nie and S. Kambhampati. Joint optimization of cost and coverage of query plans in data integration. In ACM CIKM, Atlanta, Georgia, November 2001.

[NK02] Z. Nie and S. Kambhampati. Frequency-Based Coverage Statistics Mining for Data Integration. ASU CSE TR 02-004. Dept. of Computer Science & Engg. Arizona State University. $http://www.public.asu.edu/ \sim zaiqingn/tech\_freqc.pdf$

[NNVK02] Z. Nie, U. Nambiar, S. Vaddi and S. Kambhampati. Mining Coverage Statistics for Websource Selection in a Mediator. Proc. CIKM 2002.

[PL00] Rachel Pottinger , Alon Y. Levy , A Scalable Algorithm for Answering Queries Using Views Proc. of the Int. Conf. on Very Large Data Bases(VLDB) 2000.

[WMY00] W. Wang, W. Meng, and C. Yu. Concept Hierarchy based text database categorization in a metasearch engine environment. In WISE2000, June 2000.

[ZL96] Q. Zhu and P-A. Larson. Developing Regression Cost Models for Multi-database Systems. In Proceedings of PDIS. 1996.