2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)
IPCSIT vol. 53 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V53.No.2.19

Web Databases Sampling Approach Based on Valid Probability of Query Attributes

Tian Jian-Wei[1+] and Xu Yang[2]

[1]School of Computer, Wuhan University, Wuhan 430079, China

[2]International School of Software, Wuhan University

Wuhan 430079, China

Abstract—A great portion of information is hidden behind in the query restricted Web databases, which makes it difficult to obtain these high-quality data samples. The existing hidden database samplers are very inefficient for the underflow in the sampling walk. Aiming at this problem, in this paper we propose a new approach for sampling data from deep Web based on the valid probability of query. In order to get the valid query result, we leverage the historical valid paths to calculate the valid probability of attribute value combinations in the queries, and then we give priority to the query with largest valid probability. The experimental results indicate that our approach can efficiently extracting data from the deep, and the extraction result set can well cover the deep web database.

Keywords: valid probability; data sampling; deep Web;web database

1. Introduction

As the development of the Web database, the resources in the Web database have been growing very fast. It has been estimated that the scale of these hidden resources is about 500 times [1],[8] larger than static pages, and these resources have better information quality than the resources in static pages. So the hidden resources which construct the hidden-Web have larger scale and better quality information. However, most of the data in the hidden databases can only be accessed through the query interfaces. The query interfaces are generally represented as a Web form which takes user's input and translated them into SQL queries. These queries are provided to the hidden databases and then the top-k results are returned to the users. Many online service sites are worked on this model.

Database sampling method has been used to collect statistical information from the traditional databases, which has provided the complete and unrestricted access. However, hidden databases are present behind a query interface, so the traditional sampling approaches aren't suitable and some new ways are needed to obtain statistical information from the hidden databases with restriction. Some previous approaches have been proposed to sample the hidden databases.

In this paper, we propose a new hidden databases sampling approach which is based on probability selection. Like the random sampling approach [2], we also consider the hidden database with single-tables which are only consisted of Boolean, categorical and numeric attributes for simplicity. The query interfaces allow users to specify the values on a subset of the attributes and the system returns the top-k query results. In order to improve the sampling efficiency in our approach, we constitute the query according to valid probability of the attribute values on the one hand. The valid probability indicates the probability of the query

[+] Corresponding author.
E-mail address: tianjw0509@163.com

constituted with the attributes value leading to a valid result, which can be calculated from the historical successful paths. Our experimental results show that the valid probability model based databases sampling approach can not only avoid many duplicate samples and improve the uniformity, but also enhance the sampling efficiency by using the probability model.

The outline of our paper is as follows. In section 2, we review some related work. Section 3 specify the problem and discuss the problems in the sampling algorithm proposed in [2]. In section 4, we proposed our sampling approach to solve the problems. And the experimental setting and evaluation are given in section 5. We draw the conclusion in the section 6.

2. Related Work

Deep Web (or hidden Web) resource is the major resource in the World Wide Web and there are many research work on deep Web such as such form matching, form integration, obtaining back-end data resources [3].

Traditionally database sampling has been used to extract data and gather statistical information from the databases. Random sampling methods have been studied in many literatures [4], [5]. Sampling techniques include estimation methodologies for histograms and approximate query processing using techniques [6], [7]. However, all of these sampling techniques have the same precondition that they have the complete and unrestricted access to the databases.

There are only a few works on sampling the hidden database in recent years. HIDDEN-DB-SAMPLER [2] is proposed as a random walk algorithm to sample hidden database. In this sampling algorithm, the early termination is used to enhancing the sampling efficiency, the random re-ordering attribute method is used to improve uniformity, and there is a parameter to tradeoff the efficiency and the uniformity. The sampling algorithm proposed in [2] improves efficiency and uniformity to certain extend. However, this algorithm has shortcomings which would be elaborated in section 3.

3. The Objective of Sampling Approach

Like the traditional database, the hidden database is also constituted with a set of tuples over a set of attributes. There are two kinds of attributes in the sampling process: queriable attributes and result attributes. Queriable attributes are referred to the attributes which are used to construct the SQL sentence on the query interface. The result attributes are given to the name of the attributes only displayed in the returned pages. The objective of sampling process is to acquire high quality of the samples as well as high efficiency of sampling.

Consider a hidden database table D with a set tuples $T = \{t_1, ..., t_n\}$ over a set attributes $A = \{A_1, \ldots, A_m\}$ and each attribute A_i with a set attribute values $V_i = \{v_1, \ldots, v_k\}$. And all of the attribute values in the V_i construct the value space $V = \{V_1, \ldots, V_m\}$. The values of the attributes are Boolean or categorical and other kinds of attributes can be translated to above three kinds of attributes [2]. This hidden database is accessible to the users through a query interface, where users can submit queries by specifying the values of the attributes. These queries are translated into SQL queries in the form "SELECT * FROM D WHERE $A_1=v_1$ AND … AND $A_S=v_S$". Let $R(Q) \subseteq T$ be the answer tuples set that satisfy the query Q. most of the result pages are designed to list only the top-k tuples of the answer according the ranking function, where k is a fixed constant such as 10 or 15.there are three scenarios: (a) overflow**:** the answers can not be entirely returned; (b) underflow: there is no result returned; (c) valid: the system return k or less tuples.

The high quality samples is defined as that there are less duplicated samples in the answer set $R(Q)$. there is a metric variant *skew* to calculate the sampling quality:

$$skew = \sqrt{\frac{\sum_{1 \le i \le n} (p(t_i) - 1/n)^2}{n}} \tag{1}$$

The efficiency is measured by the number of queries need to be submitted in order to gather a desired size of answer.

4. Valid Probability Model Based Sampling Approach

4.1. HIDDEN-DB-SAMPLER

The HIDDEN-DB-SAMPLER approach is based on three main ideas:

- **Early termination**: To prevent wasted queries, the algorithm detects the events that not lead to a tuple as early as possible and restart a new random walk.
- **Ordering of attributes***:* In different databases, fixed and random ordering of attributes leads to different sampling quality and efficiency.
- **Parameter to tradeoff skew versus efficiency**: the algorithm is equipped with a parameter that can be tuned to provide tradeoffs between uniform and efficiency.

Based on above three ideas, the HIDDEN-DB-SAMPLER improves the sampling efficiency as follows: the sampler firstly selects a fixed ordering of the attributes and constructs a tree-like query space for these attributes. Then instead of taking the random walks all the way until get a valid answer, it submits the queries when coming down the path. Specifically, suppose that we have reached the *i*th level that the *where* clause in SQL is "where $A_1=v_1$ AND … AND $A_{i-1}=v_{i-1}$". Before proceeding further, it executes the query and check the answer result: if the outcome is underflow, it immediately abort the random walks; if valid, it add the result tuples to the sample set; and only the outcome is overflow, it goes on along the path. But following fixed attribute query space may lead to skew, so the algorithm random reorders the attributes at the beginning of every sampling process and reconstructs a new query space. At last, the algorithm adopts acceptance probability to tradeoff skew versus efficiency.

The HIDDEN-DB-SAMPLER enhances efficiency and uniformity to certain extend. However, this algorithm has shortcoming as follows:

The method of entirely random choosing values for the attribute in the algorithm always leads to a sharply transformation from overflow to underflow. Specifically, consider a random walk with *where* clause is "where $A_1=v_1$ AND … AND $A_{i-1}=v_{i-1}$" and the result of this query is overflow. According the algorithm, when v_i is added to the query sentence, however, the query sentence with *where* clause "where $A_1=v_1$ AND … AND $A_{i-1}=v_{i-1}$ AND $A_i=v_i$" will lead to an underflow result, and as result, we must restart a new sampling walk. Now we can see that we have submitted *i*-1 queries before adding "$A_i=v_i$" to the where clause and for the attribute value v_i all these submitted queries are wasted. This problem is very common in the sampling process. Now we define sharply transformation ratio *r* as follows:

$$r = \frac{ST(Q)}{Num(Q)} \tag{2}$$

4.2. Valid Probability Based Sampling Approach

Through our observation, there following reasons that cause above three problems:

Different combinations of attribute values lead to different sampling results. Some attribute values easily lead to overflow sampling result, some lead to underflow sampling result. For example, in a Chinese real estate finder interface, the attribute house type has a value set V_{ht}={*one-room &one-hall, two-room&one-hall,…,five-house&five-hall*}, the attribute price has a value set V_p={*$0~500 thousand, $500~1500 thousand, $1.5~3 million, $3~5 million, more than 5 million*}, the attribute city also has a value set V_c={*Beijing, Shanghai, Guangzhou, Shenzhen, Wuhan, …* } .if the values {Wuhan, *$0~500 thousand* } are chosen as the query condition in a sampling walk, it is easy to lead to a overflow result. The reason is that the number of houses whose price scope is *$0~500 thousand* in Wuhan is large. But when the values combination {*Wuhan*, *$0~500 thousand*, *,five-house&five-hall* } will come out to be an underflow result, for there are scarcely houses. It is easy to see that the different combinations of attribute values will lead to different sampling results. And if we can find the combinations that will lead to valid sampling result in the sampling process, we can save many queries and improve the sampling efficiency.

Definition 1: valid probability Consider an attribute values combinations $c=(v_1,\ldots,v_k)$, $v_i \in V_i$, where V_i is values set of the attribute A_i. Valid probability $p(c)$ is the probability of a sampling walk leading to a valid result and the query condition of the sampling walk is *c*.

Considering in a sampling walk, the query condition is the attribute values combination $s=(v_1, \ldots, v_i)$. If this walk leads to valid walk, then we define that s is a success path or valid path. All the success path constitute the success path set $S=\{s_1, s_2, \ldots, s_n\}$.

After defining the valid walk, we need to address how to calculate the valid probability. In our sampling method, we leverage the historical success path to calculate the valid probability. The valid probability $p(v_i)$ of the attribute v_i is calculated as follows:

$$p(v_i) = \frac{num(v_i)}{|S|} \tag{3}$$

where the $num(v_i)$ is the number of success paths which contain the attribute value v_i.

Then the valid probability of attribute values combination $c = (v_1, \ldots, v_n)$ is calculated by the following equation:

$$p(c) = \prod_{i=1}^{n} p(v_i) \tag{4}$$

Example 1: Considering a deep Web query interface constituted with attributes A, B and C, each attribute with values: $V_A=\{a_1, a_2\}$, $V_B=\{b_1, b_2\}$, $V_C=\{c_1, c_2\}$. The historical success path set is $\{(a_1, b_2), (a_2, b_1, c_2), (b_2), (b_1, c_1)\}$. Now there is a sampling walk with sampling path (a_2, b_2, c_2). according to equation (3) and (4), the $p(a_2)=1/4$, $p(b_2)=1/2$, $p(c_2)=1/4$, and $p(a_2, b_2, c_2)=1/4*1/2*1/4=1/32$.

According to equation (1) and (2), we know that the less of the number of attribute values, the higher of valid probability. So the single attribute values are easier to lead to the valid result. This is obviously inconsistent to with the reality, for the walks with single value usually obtain too many tuples and come out to be overflow result. To solve this problem, we firstly see definition 2 and theorem 1.

Definition 2: Containing Relationship Considering attribute values combinations $c_1 = (v_1, \ldots, v_k)$ and $c_2=(v_1, \ldots, v_m)$, $v_i \in V$, where V is the attribute value space. If $\forall v_i \in c_1$ and $v_i \in c_2$, then $c_1 \subseteq c_2$.

Theorem 1: If there are attribute values combinations $c_1$and c_2, where $c_1 \subseteq c_2$. Q_1 and Q_2 are the queries with the query conditions $c_1$and c_2 respectively. Then we can conclude that $R(Q_2) \subseteq R(Q_1)$.

Theorem 1 is very easy to be demonstrated, for that the queries with more attribute values are more restricting and much easier to get less tuples.

According to theorem 1, in the sampling process, we need to increasingly add the attribute values until the sampling result is underflow or valid. And this process is coinciding with the process of selecting attribute values combinations from small to large according to their valid probability. So a sampling process is a process that the number of the attribute values continuously increases. Also that is the process of selecting attribute values combinations from small to large according to their valid probability. Specifically, the probability selection based sampling approach is as follows:

- At first, randomly reorder the attributes at the beginning of each sampling walk. Since if the attributes order is fixed, it has been demonstrated in [2] that there will be skew in the samples. In order to farthest reduce the skew, we should reorder the attributes.
- Secondly, randomly assign the value for all the attributes at one time. For the attributes among which are not independent, we should deal with them according to the combinations. In order to deal with the non-independent attribute, we adopt the conditional probability method. Concretely, considering attributes A and B which are not independent, there are attribute values v_i and v_j, $v_i \in V_A$, $v_j \in V_B$, when the value of A is v_i, the probability of the B being assigned v_j is $p(v_j \mid v_i)$.

$$p(v_j \mid v_i) = \frac{p(v_i v_j)}{p(v_i)} \tag{5}$$

- At last, calculate the valid probability of the single attribute value and select attribute value which has the largest probability value as the query condition. Submit the query and check the result. If the result is underflow, return and start a new sampling walk; if valid, save this attribute values as the success path and start a new sampling walk; or the result is overflow, calculate the probability of attribute values combinations which are constituted with two values, and select the combinations which has the largest valid probability as the query condition. Submit this query and check the result too. Loop this process until all the attribute values are used. In example 1, according this method, we first choose single attribute value b2 as the query condition and submit the query. If the result is overflow, we then choose (a2, b2) as query. If the result is also overflow, we at last select combination (a2, b2, c2) as query

condition.

5. Experiments

We have performed performance evaluation of our deep Web databases sampling approach based on probability selection over real hidden database in the real estate domain. All of the query interfaces were crawled from the Chinese real estate port which is www.soufun.com. All experiments were run on a computer having 1.8 Ghz AMD processor with 512 MB RAM running Windows XP. Sampling procedure was implemented using Java in the Eclipse 3.2.2 integrated development environment.

5.1. Efficiency of the samplers

From figure 1 we can see that the efficiency of our deep Web databases sampling approach based on probability selection and rule mining is generally higher than the HIDDEN-DB-SAMPLER method which had been put forward in document [2]. Where when sampling times was smaller than 600 times, the efficiency of our method is instability, that is, sometimes it's higher than HIDDEN-DB-SAMPLER and sometimes is lower than it. But when the number of samplers was larger than 600, the average efficiency of the deep Web databases sampling approach based on probability selection and rule mining was more than 0.16. At the same time, the average efficiency of HIDDEN-DB-SAMPLER was less than 0.15. When the number of samplers was 600 times, the largest efficiency of our method had arrived at 0.2. With the increase of sampling times, the number of successful paths is also increased, so it can calculate the correct and stable valid probability of each attribute value from a large number of successful paths.

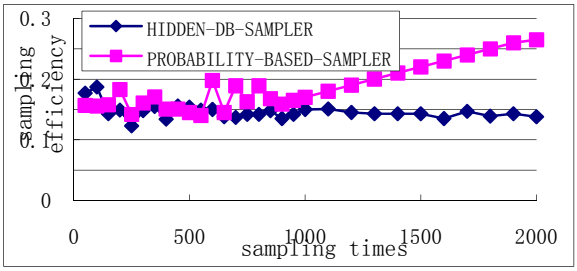

Fig.1. Number of queries from 50 times to 2000 times

5.2. Quality of the Samplers

As shown in figure 2, we can see that with the increase of sampling times, the number of duplicated samples is also increased, and it has less duplicated samples than the HIDDEN-DB-SAMPLER method has in sampling result set. This is because our approach uses probability selection method to choose attribute values combination which has largest effective probability in the sampling process. So it is imperative to deal with the duplicated samples in the sampling result set.

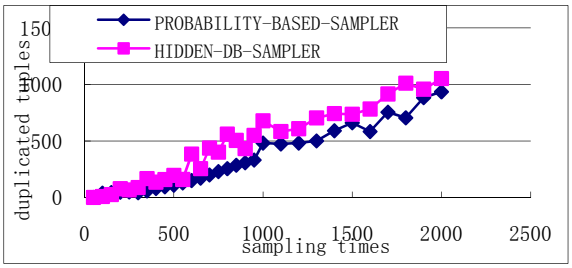

Fig.2. Number of duplicate samples from 50 times to 2000 times

6. Conclusion

In this paper, in order to solve the problems existed in the current deep web database sampling approach, we proposed the deep Web databases sampling approach based on probability selection. In the sampling

process, we choose attribute values combinations with largest valid probability as the sampling query. From the experimental results, we can see that our deep Web databases sampling approach improves the efficiency of the sampler, greatly reduces the number of sampling queries and the sharply transformation from overflow to underflow.

7. Acknowledgment

This work is supported by National Natural Science Foundation of China (No.60773007). The authors are grateful for the anonymous reviewers who made constructive comments.

8. References

[1] M. K. Bergman. The Deep Web: Surfacing Hidden Value (White Paper). Journal of Electronic Publishing, 7(1), August 2001.

[2] A. Dasgupta, G. Das, H. Mannila. A random walk approach to sampling hidden databases SIGMOD Conference 2007: 629-640

[3] Kevin Chen-Chuan, Chang, Bin He, Zhen Zhang. Toward Large Scale Integration Building a MetaQuerier over Databases on the Web. In Proceedings of the Second Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, Ca, January 2005. 44~55.

[4] S. Chaudhuri, G. Das and U. Srivastava. Effective Use of Block-Level Sampling in Statistics Estimation. In Proceedings of ACM SIGMOD, 2004.

[5] P. J. Haas, C. A. Koenig. Bi-Level Bernoulli Scheme for Database Sampling. In Proceedings of ACM SIGMOD, 2004, 275-286.

[6] M. N. Garofalakis and P. B.Gibbons. Approximate Query Processing: Taming the TeraBytes. In Proceedings of VLDB, 2001

[7] J. P. Callan and M. E. Connell, "Query-based sampling of text databases," ACM Transactions on Information Systems, vol. 19, no. 2, pp. 97–130 (2001).

[8] W. Liu, X. Li, X. Meng, et al: A Deep Web Data Integration System for Job Search. Wuhan University Journal of Natural Sciences, 2006,11(5): 1197~1201