ISAR International Journal of P2P Network and Techniques -– Volume 1 Issue 3, May – June 2016

RESEARCH ARTICLE OPEN ACCESS

Retrieving Deep Web Using Aggregate Queries with Check Box Interface

S.Shagitha[1], E.Dhilipkumar[2]

[1]PG Student, [2]Associate Professor

[1, 2,] Department of MCA,Dhanalakshmi srinivasan Engineering College of technology

Abstract:

A large number of web data repositories are hidden behind restrictive web interfaces, making it an important challenge to enable data analytics over these hidden web databases. Most existing techniques assume a form-like web interface which consists solely of categorical attributes (or numeric ones that can be discretized). Nonetheless, many real-world web interfaces (of hidden databases) also feature checkbox interfaces—e.g., the specification of a set of desired features, such as A/C, navigation, etc., for a car-search website like Yahoo! Autos. We find that, for the purpose of data analytics, such checkbox-represented attributes differ fundamentally from the categorical/numerical ones that were traditionally studied. In this paper, we address the problem of data analytics over hidden databases with checkbox interfaces. Extensive experiments on both synthetic and real datasets demonstrate the accuracy and efficiency of our proposed algorithms.

1. INTRODUCTION

Invisible web is that which we cannot find using general web search engines. Terms like invisible web, hidden web also refers to same thing. A huge chunk of data is buried in database and other research resources. The hidden data is estimated to be in terabytes or more. In 2000, this topic was new and baffling to many web searches but many crawlers have overcome many technical problems that made web searchers impossible to find invisible web pages. This buried content is estimated to be 500 times bigger than surface web. This data is searchable and available in online web but sometimes ignored by conventional web search engines.

When a user inputs a search query through the input interface, the web crawler selects a limited number of tuples satisfying the search conditions specified by users. Hence web users who rely on search engines are unable to discover and access a large amount of information from non-index able part of the web. Many hidden databases deliver their top N results and hence a large data is getting hidden. Moreover search engine companies exclude some web pages to avoid cluttering their databases with unnecessary content. Especially dynamic web pages generated based on the search interfaces are not recognized by search engines and hence they stay hidden and are not seen in searcher's results.

Due to this many databases on the web are hidden behind accessible only through restrictive web search interfaces. Many web interfaces does not provide functions like COUNT and SUM for aggregate estimation of invisible web. Hence it is necessary to extend the search engine capacity and to use aggregate functions such as COUNT and SUM in standard SQL. Then this makes many applications to take both hidden and visible database as their data sources. Then the invisible data becomes visible. Another problem is that sometimes user need to perform a broad search and should visit unnecessary pages if he is longing for a particular website. Too many documents should be retrieved before hitting the target.

There are some websites which use a technique to solve this problem .Many real world databases have some interfaces but contain combination of check box attributes. This attributes clearly provide the results satisfying user's query. There are some websites which use checkbox interfaces. For example a job search website (Monster.com) has nearly 95 checkboxes. But this concept is not applied in search engine.User need to crawl many pages to get desired results. Checkbox interface in search engines make easy and comfortable search.

Current web search engines include in their indices only a portion of the Web. There are a number of reasons for this, including inevitable ones, but the most important point here is that the significant part of the Web is unknown to search engines. It means that search results returned by web searchers enumerate only relevant pages from the indexed part of the Web while most web users treat such results as a complete (or almost complete) set of links to all relevant pages on the Web.

In this thesis our primary and key object of study is a huge portion of the Web hidden behind web search interfaces, which are also called web search forms. These interfaces provide web users with an online access to myriads of databases on the Web. At the same time, web forms are formidable barriers for any kind of automatic agents, e.g., web crawlers, which, unlike human beings, have great difficulties in filling out forms and retrieving information from returned pages.

Hereafter we refer to all web pages behind search interfaces as the deep Web. The deep Web is not the only part of the Web, which is badly indexed by search engines. For better understanding the subject we start with a brief description of the non-index able portion of the Web in general.

Two Deep Web With the advances in web technologies [1, 2, 5, 7, 45, 48, 54, 60, 84], web pages are no longer confined to static HTML files that provide direct content. This leads to more interactivity of web pages and, at the same time, to ignoring a significant part of the Web by search engines due to their inability to analyze and index most dynamic web pages. Particularly, query-based dynamic portion of the Web known as the deep Web is poorly indexed by current-day search engines. There is a slight uncertainty in the terms defining the part of the Web that is accessible via web search interfaces to databases.

2. LITERATURE SURVEY

Recently, there has been increased interest in the retrieval and integration of hidden-Web data with a view to leverage high-quality information available in online databases. Although previous works have addressed many aspects of the actual integration, including matching form schemata and automatically filling out forms, the problem of locating relevant data sources has been largely over-looked. Given the dynamic nature of the Web, where data sources are constantly changing, it is crucial to automatically discover these resources. However, considering the number of documents on the Web (Google already indexes over 8 billion documents), automatically finding tens, hundreds or even thousands of forms that are relevant to the integration task is really like looking for a few needles in a haystack. Besides, since the vocabulary and structure of forms for a given domain are unknown until the forms are actually found, it is hard to define exactly what to look for. We propose a new crawling strategy to automatically locate hidden-Web databases which aims to achieve a balance between the two conflicting requirements of this problem: the need to perform a broad search while at the same time avoiding the need to crawl a large number of irrelevant pages. The proposed strategy does that by focusing the crawl on a given topic; by judiciously choosing links to follow within a topic that are more likely to lead to pages that contain forms; and by employing appropriate stopping criteria. We describe the algorithms underlying this strategy and an experimental evaluation which shows that our approach is both effective and efficient, leading to larger numbers of forms retrieved as a function of the number of pages visited than other crawlers.

A significant part of the information available on the Web is stored in online databases which compose what is known as Hidden Web or Deep Web. In order to access information from the Hidden Web, one must fill an HTML form that is submitted as a query to the

underlying database. In recent years, many works have focused on how to automate the process of form filling by creating methods for choosing values to fill the fields in the forms. This is a challenging task since forms may contain fields for which there are no predefined values to choose from. This article presents a survey of methods for Web Form Filling, analyzing the existing solutions with respect to the type of forms that they handle and the filling strategy adopted. We provide a comparative analysis of 15 key works in this area and discuss directions for future research.

An increasing amount of Web data is accessible only by filling out HTML forms to query an underlying data source. While this is most welcome from a user perspective (queries are easy and precise) and from a data management perspective (static pages need not be maintained; databases can be accessed directly), automated agents have greater difficulty accessing data behind forms. In this paper we present a method for automatically filling in forms to retrieve the associated dynamically generated pages. Using our approach automated agents can begin to systematically access portions of the “hidden Web.”

Today, information integration has assumed a completely different, complex connotation than what it used to be. The advent of the Internet, the proliferation of information sources on the surface Web as well as the deep Web, the presence of structured, semi-structured, and unstructured data - all have added new dimensions to the problem of information integration as known earlier. From the time of distributed databases leading to heterogeneous, federated, and multi-databases, retrieval and integration of information from heterogeneous sources has been an important and complex problem. Currently, the problem is even more complicated as repositories exist in various formats (HTML, XML, spatial data sources to name a few) and schemas, and both the content and the structure of the data within them are changing autonomously. As the number of repositories/sources will continue to increase in an uncontrolled manner, there is no other option but to find extensible techniques for answering a complex search/query whose (partial) answers have to be retrieved and integrated from multiple sources. In this survey paper, we identify the set of challenges that need to be addressed for this form of heterogeneous information integration, and compare the current state-of-the-art as to how they fare. We then propose a framework with functional components – termed Info Mosaic, that aims to address some of these important challenges, and briefly elaborate on the data and control flow involved in answering a complex query/search.

In this paper, we study the problem of automating the retrieval of data hidden behind simple search interfaces that accept keyword-based queries. Our goal is to automatically retrieve all available results (or, as many as possible). We propose a new approach to siphon hidden data that automatically generates a small set of representative keywords and builds queries which lead to high coverage. We evaluate our algorithms over several real Web sites. Preliminary results indicate our approach is effective: coverage of over 90% is obtained for most of the sites considered.

This paper reports a pragmatic experiment, consisting on the integration of a Web search engine with a traditional library's catalogue. PORBASE is the Portuguese bibliographic union catalogue. It holds more than one million of bibliographic and nearly half a million of authoritative structured records about authors, families, organizations and subjects. tumba! is a search engine specialized in indexing web pages in Portuguese language. It indexes actually more than one million of pages, found potentially everywhere in the web. Since the early days of the Web that we have been assisting to a discussion about the pros and cons of web indexes built automatically by search engines as an alternative to traditional human-built databases, as the library's catalogues are. The main argument against search engines has been that they might be very effective in indexing the surface skins of the web, but they miss the richest contents hidden behind the more complex web sites and databases, the so called "deep web". On the other side their defenders point the relatively low cost of those solutions, which make it possible to provide good services without the costs of manual cataloguing, classification, indexing of the resources. This paper reports about an experiment to explore the best of both worlds, by using PORBASE in conjunction with tumba!. We extracted from PORBASE the major index of the authorities (names of persons), along with the frequency

of each entry, and integrated it within the tumba! search engine. We also developed a very simple HTTP-based interface, ESPONJA, which tumba! can use to launch queries in PORBASE using the authoritative descriptions as arguments. In the scenario of this experiment, a user performing a search session using tumba! will be able to search also in PORBASE using the authoritative form of the search terms without knowing them in advance. The process starts when tumba! presents its own results complemented with suggestions (tips) to search in PORBASE. If the user chooses to launch a search there, a new window is opened showing the results of a search for the term. After that it is up to the user to continue interacting with PORBASE, to explore new results and take advantage of the specific functions of its specialized catalogue. This work aims to be only a proof of concept for this approach. Next steps will comprise the tuning of the algorithms to process the authority information in tumba!, and the formalization of the ESPONJA interface toward a stable, open, generic and reliable interface for interoperability with BN's specialized bibliographic systems.

3. PROBLEM DEFINITION

Many databases on the web are "hidden" behind.Terabytesof information is buried in databases. Accessible only through their restrictive web search interfaces.Allow a user to specify the desired values for one or a few attributesReturn to the user a small number (e.g., k = 50 or 100) of tuples that match the user-specified query.Searchable and accessible online but often ignored by conventional search engines, these resources exist by the thousands. Known in research circles as the deep web, invisible web, or hidden web.Current searchers deal well with informational (give me the information I search for) and navigational (give me the URL of the site I want to queries, but transactional (show me sites where I can perform a certain transaction, e.g., shop, access database, download a file, etc.) queries are satisfied only indirectly.It is needless to say that data in the deep Web is indispensable to use when answering transactional queries that constitute about 10% of web search engine queries.The fact that the deep Web is still poorly indexed is another major issue for search engines, which are eager to improve their coverage of the Web.

The main objective is to make HIDDEN web or INVISIBLE web to be accessed by users without visiting many pages and hyperlinks. It is used to reduce the searching time.To solve the problem we use check box interfaces.We perform an unbiased search through this we can make the hidden web visible to the user.

4. PROPOSED WORK

In our proposed system we introduce checkbox interface to make hidden data visible which contains many checkbox attributes. This check box attributes are available as a drop down list in the web search engine. The checkbox attributes appear according to the query given by the user.

Here to make the hidden data visible the data is given in checkbox interface which contains many checkbox attributes .This check box attributes are available as a drop down list in the web search engine. The checkbox attributes appear according to the query given by the user. For example When the user search query is 'online shopping' then item types are represented in checkboxes (example: groceries, stationary, electronic devices and gadgets, accessories etc…) the checkbox interface has its specialty. By checking the check boxes the crawler collects the results set of data corresponding to their particular check boxes and represents back in the search interface. The uncheck boxes are interpreted as 'do not care' insteadof 'notcontaining' in the interface. Hence all existing techniques relay on an assumption that the invisible web has an interface (checkboxes or drop down list).The user has to check or select his desired value for an attribute. By checking a list of checkboxes drop down list, the user excludes all other search related to that query. We can represent the checked attributes with value one and unchecked as zero. Suppose consider Boolean values then checked attributes are represented as TRUE and unchecked attributes are represented as FALSE. Hence this makes a desired search without any unwanted information is represented in fig 1.

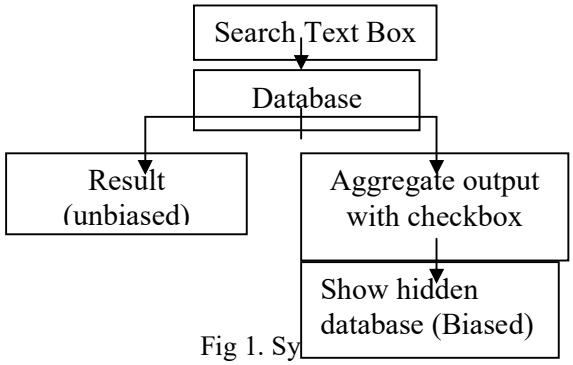

Fig 1. Sy

In the proposed system, query issued by the user to search engine will show results based on keywords. When queried the search engine provide checkbox interfaces which satisfy the user by giving accurate information. User gets desired search results without performing a broad search. User need not visit unwanted pages unnecessarily or spend too much time in searching.

The existing problems are addressed by using algorithms like

- Efficient crawling algorithm
- Aggregate accuracy algorithm

4.1Efficient crawling algorithm:

The query that is issued by the user along with his checkbox selection are focused and crawled efficiently by our algorithm. This algorithm first selects the query that is issued by the user and gives the results (unbiased). It then downloads all the Hidden Web pages from the site on the basis of the checkbox preference; results (unbiased) will be produced. The process is repeated until all the available data are completely used up. Focused crawling is achieved by our algorithm. A web crawler, an automatic tool used by search engines to collect web pages to be indexed, browses the Web in the following way: it starts witha predefined list of URLs to visit (called the seeds) and, as the crawlerprocesses these URLs, it extracts all hyperlinks from visited pages and adds them to the list of URLs to visit (called the crawl frontier). URLs from thefrontier are recursively visited according to a set of crawler's rules. Clearly,web pages which URLs are neither in the seeds nor in the frontier are notadded to search engines' indices.

Step1. while (data available) do
// select the preference
Step2. qi = SelectQuery()
//selected query
Step 3.R(qi) =WebsiteQ(qi)
// where R(qi) is the result page for query qi.
Step 4. Download(R(qi));
Step 5. End;

4.2 Aggregate accuracy algorithm:

We use the notation q(ri) to represent the first non-overflowing query for drilldown ri and let Q = fq(r1); : : : ; q(rs)g. parent(q(ri)) corresponds to parent of q(ri) in the query tree.

Aggregate tracking with checkbox interface as selection condition: So far, we have focused on aggregate queries that select all biased results from the database. To support aggregates with selection conditions a check box interface is fixed. Through checkbox selection, a condition is specified via conjunctive constraints over a subset of attributes. Given selection conditions Q, we alter our query tree to be the sub tree corresponding to Q. We build a query tree with all queries producing a new tree. With the new query tree, drill-downs can be computed directly over it to a get an unbiased estimate.

Step 1:Randomly generate signature set S = fr1; : : : ; rsg
Step 2: In R1, perform drill downs in S till exhausting query budget. Update Q.
Step 3: for each round Ri do
Step 4: for drill-down rj 2 S do
Step 5: if q(rj) overflows then
Step 6: Do drill-down from q(rj) till a non-overflowing node
Step 7: else if q(rj) underflows then
Step 8: Do roll-up from q(rj) till a non-under flowing node or an under flowing node with an overflowing parent
Step 9: end if
Step 10: end for
Step 11: Issue new drill-downs from S for remaining query budget
Step 12: Produce aggregate estimation according to Q
Step 13: end for

5. MODULES DESCRIPTION

In this project we use three modules they are

- Unbiased estimation.
- Check box interface.
- Biased result with aggregate estimation.

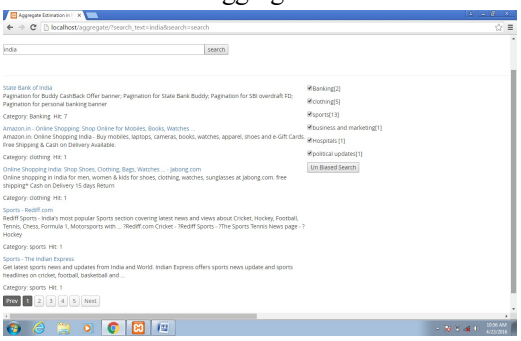

Fig 5

5.1. Unbiased estimation

A statistic is said to be an unbiased estimate of a given parameter when the mean of the sampling distribution of that statistic can be shown to be equal to the parameter being estimated.For example, the mean of a sample is an unbiased estimate of the mean of the population from which the sample was drawn.

s^2 calculated on a sample is an unbiased estimate of the variance of the population from which the sample was drawn.

s^2 divided by n (the size of the sample) is an unbiased estimate of the variance of the sampling distribution of means for random samples of size n and the square root of this quantity is called the standard error of the mean. It is a commonly used index of the error entailed in estimating a population mean based on the information in a random sample of size n.

We develop the data structure of left-deep-tree and define the concept of designated query to form an injective mapping from tuples to queries supported by the web interface.

The aggregate tracking algorithm is based on the fixed weight allocation of edges and the estimator To estimate the number of tuples in D, the algorithm will first use some fixed probabilities to assign weights to edges of the left-deep tree, then randomly drill down and estimate the number of tuples in D. The second step will be recursively executed until the query cost exceeds the query budget. The weight allocation does not affect the unbiasedness of the estimation algorithm as long as each tuple has a probability to be retrieved.

5.2 Check box interface:

Checkbox arethose that permit the user to make a binary choice, i.e. a choice between one of two possible mutually exclusive options.

In the hidden database with checkbox interface, a checkbox attribute is represented as a checkbox in the web interface. For example, in the home search website, features (e.g., central air, basement) for a home are represented by checkboxes. The checkbox interface has its specialty. By checking the checkbox corresponding to a value v1, it ensures that all returned tuples contain the value v1. But it is impossible to enforce that no returned tuple contains v2 – because unchecking v2 is interpreted as ”do-not-care” instead of ”not-containing-v2” in the interface.

5.3 Biased result with aggregate estimation:

In this module we use aggregate tracking algorithm as the same with UNBIASED-estimation except crawling the whole subtree of a node if the probability of the node is lower than a given threshold.At the very beginning of the sampling process, the drill-down is unlikely to encounter many rarely-hit tuples.

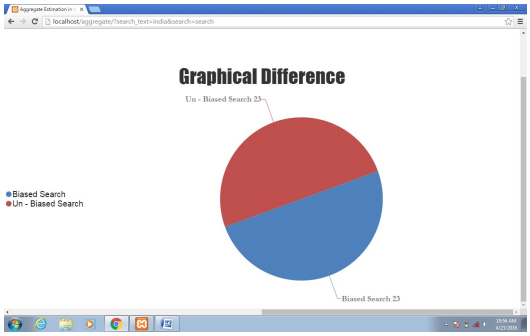

Fig 5 Graphical Difference

6. CONCLUSION

In this paper we proposed two algorithms, namely efficient crawling algorithm and aggregate accuracy algorithm for obtaining aggregate results from the hidden databases. Our algorithms along with checkbox interface automatically discover hidden-Web databases. It is able to efficiently perform a broad search by focusing the search on a given topic; by learning to identify links; and by using appropriate stop criteria which avoid unproductive searches within individual

sites. Hence the search results satisfy user without performing a broad search.

REFERENCES

[1] C. Sheng, N. Zhang, Y. Tao, and X. Jin, "Optimal algorithms for crawling a hidden database in the WEB," Proc. VLDB Endowment, vol. 5, no. 11, pp. 1112–1123, 2012.

[2] A. Dasgupta, N. Zhang, and G. Das, "Turbo-charging hidden database samplers with overflowing queries and skew reduction," in Proc. 13th Int. Conf. Extending Database Technol., 2010, pp. 51–62.

[3] S. Raghavan and H. Garcia-Molina, "Crawling the hidden web,"in Proc. 27th Int. Conf. Very Large Data Bases, 2001, pp. 129–138.

[4] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep Web: A survey. Communications of the ACM, 50(2):94