Indexing for Vertical Search Engine: Cost Sensitive

Sudhakar Ranjan¹, Komal Kumar Bhatia²

¹Assistant Professor, Apeejay Stya University, Gurgaon, India
²Associate Professor, YMCAUST, Faridabad, India.

Abstract—The information on the WWW is growing exponentially and the dynamic, unstructured data & structured data needs to locate as useful resources, web pages and online database in enormous quantity. In this paper we propose the novel indexing technique to download the hidden web pages which is based on domain specific. This technique keeps the related documents in the same domain so that searching of documents becomes more efficient in terms of time complexity and cost sensitive.

Keywords—Indexing, Vertical Search Engine, Domain Specific, Distributed System, Hidden documents.

I. INTRODUCTION

As the volume of information in the surface and hidden web grows, there is increased interest in techniques and tools that allow users and applications to influence this information. There is a remarkable amount which is hidden behind the search forms. This amount is crawled by Hidden web Crawler. These pages are often referred to as the Hidden Web, the Deep Web or Vertical Search, because search engines typically cannot index the pages and do not return them in their results (thus, the pages are essentially “hidden” from a typical Web user). In paper [10] a large amount of on-line information resides on the invisible web - web pages generated dynamically from databases and other data sources hidden from current crawlers which retrieve content only from the publicly indexable Web. Specially, they ignore the tremendous amount of high quality content “hidden” behind search forms, and pages that require authorization or prior registration in large searchable electronic databases. To extracting data from the hidden web, it is necessary to find the search forms and fill them with appropriate information to retrieve maximum relevant information. To fulfill the complex challenges that arise when attempting to search hidden web i.e. lots of analysis of search forms as well as retrieved information also, it becomes eminent to design and implement a distributed web crawler that runs on a network of workstations to extract data from hidden web. In this paper describe the architecture of the distributed, scalable system and also present a number of novel techniques that went into its design and implementation to extract maximum relevant data from hidden web for achieving high performance.

In paper [7] the DSHWC downloads the Hidden web documents and stores them in the repository. After this some index function needs to be applied over this repository in order to index the Hidden web contents. Moreover, simple indexing technique cannot be applied to index the Hidden web contents due to the following reasons:

II. RELATED WORK

In the literature, there are many researchers mentoring about the Rank based [1], Topic Relevance Algorithm [2], Rank Queries [3], Data Extraction [4], Correlation of vertical search engine [5], Topics based framework [6]. In [1] a novel query on uncertain databases, namely uncertain top-k influential site query, define this query based on the intuitive expected rank semantics and also propose pruning techniques and partition-based algorithms to improve the querying performance. The experimental results verify the effectiveness and efficiency of the techniques in this paper. In [2] Hits algorithm assumes that all the link between web pages have a equal value.
It ignores difference between these links. Then it has a serve problem of topic drift. The improved Hits algorithm that doesn’t treat all links equally. It assigns different weight to different links. In[3] To adapt important properties those guide the definition of ranking queries in deterministic databases and analyze characteristics of existing top-k ranking queries for probabilistic data. These properties naturally lead to the ranking approach that is based on the rank distribution for a tuple across all possible worlds in an uncertain domain. Efficient algorithms for two major uncertainty models ensure the practicality of our approach. The experiments demonstrate that ranking by expected ranks, median ranks and both attribute-level and tuple-level uncertainty models. In [4] a prototype system of domain and keyword specific system is presented. The developed prototype systems provide more valuable information from the hidden databases at one single location, which will provide effective search environment to end user. Here the system deals with both methods of form submission i.e. get and post. The results obtained were encouraging. In [5] design the Vertical search engine of HR, the decision method of theme correlation based on ontology was used. To facilitate the handling spiders crawling hyperlinks and related calculations, the design of 4 URL queues: Waiting Queue, Running Queue, Completed Queue, and Exceptions Queue. At first, all of URL into Waiting Queue, and analyze each URL by spider program, if the hyperlink that the URL pointing cannot be crawled by the spider or calculation values of theme correlation less the threshold predetermined, then put the URL into exceptions queue and pass this URL to do the next one, else put it into the Running Queue for analyze and deal with, after finish, put it into the Completed Queue and get the next URL from Waiting Queue for analyze. In [6] presented a general framework to build classification and matching/ranking models for short and sparse text/Web data by taking advantage of hidden topics from large-scale external data collections. The framework mainly focuses on several major problems. When processing such kind of data: data sparseness and synonyms/ homonym problems. The approach provides a way to make sparse documents more related and topic-focused by performing topic inference for them with a rich source of global information about words/terms and concepts/topics coming from universal data sets.

The integration of hidden topics helps uncover and highlight underlying themes of the short and sparse documents, helping us overcome difficulties like synonyms, hyponyms, and vocabulary mismatch, noisy words for better classification, clustering, matching, and ranking. In addition to sparseness and ambiguity reduction, a classifier or matcher built on top of this framework can handle future data better as it inherits a lot of unknown words from the universal data set. Also, the framework is general and flexible to be applied to different languages and application domains. In this paper carried out two careful experiments for two evaluation tasks and they have empirically shown how our framework can overcome data sparseness and ambiguity in order to enhance classification, matching, and ranking performance.

III. PROPOSED WORK

COST SENSITIVE: Index Integrator Techniques plays important role in Cost-sensitive classification. In fig.2 (Index Integrator) load balancer distribute the hardware and software traffic load. The user searching document request sends to the domain specific hidden web crawler. In indexing used segmentation and automatic segmentation techniques. The advantage of automatic segmentation is no maintains cost and easy to realize, and segmentation gives effective and high precision result.

INDEX INTEGRATOR: A domain specific indexing technique based on label and their value set for hidden web documents. Join indexes are useful to perform a query on two relations. A join index on a relation involves an attributes values from different relation through a common link encounter join. Multiple join indexing is performing a sequence of bitwise operations. This indexing technique is very useful for reducing the space complexity, optimizes speed and performance for finding documents, but also gives more specified results for a search query. Vertical search engine which demands high accuracy, for high accuracy we require efficient indexing techniques. The Index Integrator and Query Interface Integrator give effective, efficient and accurate results. As fig.2 (Index Integrator) and fig.3 (Workflow diagram for Query Interface Integrator (QII)), the all major components of brief description given in table 1.
### Table 1: Brief component description

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Balancer</td>
<td>Hardware and Software solution to distribute traffic over web servers.</td>
</tr>
<tr>
<td>DSHWC</td>
<td>The Domain specific hidden web crawler’s goal is to automate the process of searching, viewing, filling in and submitting the search forms and analyzing the response pages.</td>
</tr>
<tr>
<td>Indexer</td>
<td>An Indexer is a program that “reads” the pages, which are downloaded by spiders. The indexer does most of the work deciding what the web site is about. Web indexing includes back-of-book-style indexes to individual websites or web documents and the creation of metadata to provide a more useful vocabulary for Internet search engines. The indexer also examines the HTML code, which makes up the webpage and looks for words that are considered important. Words in bold, italics or header tags are given more priority.</td>
</tr>
<tr>
<td>Repository</td>
<td>A DSHWC repository stores and manages a large number of web pages.</td>
</tr>
<tr>
<td>Indexing Module</td>
<td>Indexing Module is the core part of the system. Indexes are used for searching.</td>
</tr>
<tr>
<td>Index DB</td>
<td>Store information and provide quick access to the search result.</td>
</tr>
<tr>
<td>Web Log</td>
<td>Whenever user surfs a website every web server maintains the list of actions performed/requested by the user into a web log files.</td>
</tr>
<tr>
<td>Query Log</td>
<td>It is a collection of web query. Whenever user generated and modified query.</td>
</tr>
<tr>
<td>Local query Interface</td>
<td>A local query is generated by using the local language and data manipulation languages for each local query.</td>
</tr>
<tr>
<td>Query Processing</td>
<td>Query processing is the procedure of transforming a high level query (SQL) into a correct and efficient plan expressed in low level languages.</td>
</tr>
<tr>
<td>Result Integrator</td>
<td>Local result is sent for result integration by the network administration as a message. The results of local queries (LQ’s) must be interpreted and assembled according to the global join conditions.</td>
</tr>
</tbody>
</table>

**Fig. II: Index Integrator**

**Fig. III Work flow diagram for QII**

### IV. Results

The Relation R1, R2, ….., Rn are to be integrated for a relation S. The QI does not allow the same attributes to appear multiple times, for airline domain we used three relations {Option, Status and Passenger} and one view for improve query performance in term of access time, maintenance and cost. The matching between the attributes of a set of query interfaces in the same application domain (e.g. book, airline, movies, real state).
The global interface must resemble the look and feel of local interface as much as possible despite being automatically generated without human support. Query interface organized in groups like passenger information: {Adults, Children, Infant} Option group (From, To, One way, Round Trip, Multicity, Package, Class). Status group {Leave, Return, Leave Date, Return Date}. Interface group of fields organized in super groups like {Adults, Children, Infant, From, To, One way, Round trip, Multicity, Package, Class, Leave, Return, Leave Date, Return Date}. When user choose One way trip then interface group of fields organized in super group appear after choosing Leave status {Adults, Children, Infant, From, To, Leave, Leave Date}. Bottom up grouping leads to a hierarchical structure for query interface.

V. CONCLUSION

In this paper we proposed, improved indexing techniques for hidden web contents. The indexing technique not only optimizes speed and performance for finding documents but also gives more specified results for a search query. The Global Query Interface is designed by interface matching techniques. This Global Query Interface is employed to generate the required index. This indexing technique is very useful for reducing the time, space complexity, optimizes speed, performance for finding documents and cost sensitive, but also gives more specified results for a search query.

REFERENCES


