Near Duplicate URL Detection for Removing Dust Unique Key

R. Vijaya Santhi
Research Scholar, Department of Computer Science, Tamil University College, Thanjavur, Tamil Nadu, INDIA

ABSTRACT

Regular parallel mining algorithms for mining frequent item sets intends to balance load by equally partitioning data among a group of computing nodes. But those existing parallel frequent item set mining algorithms have serious performance issues. In big data environment existing mining algorithm suffer high communication and mining overhead induced by redundant data transmitted among computing nodes. We explore this problem by developing a data partitioning approach using the MapReduce programming model. The aim of this paper is to enhance the performance of parallel frequent item set mining on Hadoop clusters. Incorporating the similarity metric and the Locality-Sensitive Hashing technique, in this proposed model VUK (Valid Unique Key) DUST removing technique LDA-CRATS mining data is used to run this approach. This approach is to derive quality rules that take advantage of a multi-sequence alignment strategy. It demonstrates that a full multi-sequence alignment of URLs with duplicated content, before the generation of the rules, can lead to the deployment of very effective rules. By evaluating this method, it observed it achieved larger reductions in the number of duplicate URLs than our best baseline, with gains of 85 to 150.76 percent in two different web collections.

Keywords-- MapReduce, URL duplicated, multi-sequence, Hadoop

I. INTRODUCTION

The dust problem is that: The web is abundant with dust, different URLs with Similar Text. For example, the URLs http://google.com/news and http://news.google.-com return similar content. A single web server often has multiple DNS names, and any can be typed in the URL.

Many are artifacts of a particular web server implementation. For example, URLs of dynamically generated pages often include parameters; which parameters impact the page’s content is up to the software that generates the pages. Some sites use their own conventions; for example, a forum site we studied allows accessing story number “num” both via the URL http://domain/story?id=num and via http://domain/story num. Our study of the CNN web site has discovered that URLs of the form http://cnn.com/money/whatever get redirected to http://money.cnn.com/whatever.

Universal rules, such as adding http:// or removing a trailing slash are used, in order to obtain some level of canonization. By knowing dust rules, one can dramatically reduce the overhead of this process. But how can one learn about site-specific dust rules?

Detecting dust from a URL list. Most of our work therefore focuses on substring substitution rules, which are similar to the “replace” function in many editors.

Dust Buster uses three heuristics, which together are very effective at detecting likely dust rules and distinguishing them from false rules. The first heuristic is based on the observation that if a rule α → β is common in a web site, then we can expect to find in the URL list multiple examples of pages accessed both ways. For example, in the site where story?

II. LITERATURE SURVEY

2.1 URL NORMALIZATION FOR DE-DUPLICATION OF WEB PAGES

Presence of duplicate documents in the World Wide Web adversely abets crawling, indexing and relevance, which are the core building blocks of web search. In this paper, we present a set of techniques to mine rules from URLs and utilize these learnt rules for de-duplication using just URL Strings without fetching the content explicitly. Our technique is composed of mining the crawl logs and utilizing clusters of similar pages to extract specific rules from URLs belonging to each cluster. Preserving each mined rules for de-duplication is not ancient due to the large number of specific rules. We present a machine learning technique to generalize the set of rules, which reduces the resource footprint to be usable at web-scale. The rule extraction tech-Inquest is robust against web-site specific URL conventions. We demonstrate the effectiveness of our techniques through experimental evaluation.

2.2 DO NOT CRAWL IN THE DUST: DIFFERENT URLS WITH SIMILAR TEXT

We focus on URLs with similar contents rather than identical Ones, since different versions of the same
2.5 MODELS AND ALGORITHMS FOR DUPLICATE DOCUMENT DETECTION

As information management and networking technologies continue to proliferate, document image databases are growing rapidly in size and importance. A key problem facing such systems is determining whether duplicates already exist in the database when a new document arrives. This is challenging both because of the various ways a document can become degraded and because of the many possible interpretations of what it means to be a “duplicate.”

3.1 PROBLEM DEFINITION

We view URLs as strings over an alphabet Σ of tokens. Tokens are either alphanumeric strings or non-alphanumeric characters. In addition, we require every URL to start with the special token ^ and to end with the special token $ (and $ are not included in Σ). For example, the URL http://www.site.com/index.html is represented by the following sequence of 15 tokens: ^http://wwwsitecom/index.html$. We denote by U the space of all possible URLs.

A URL u is valid, if its domain name resolves to a valid IP address and its contents can be fetched by accessing the corresponding web server (the http return code is not in the 4xx or 5xx series). If u is valid, we denote by doc (u) the returned document.

DUPLICATE: Two valid URLs u1, u2 are called dust, if their corresponding documents, doc (u1) and doc (u2), are “similar”. DUPLICATE RULES: In this thesis, we seek general rules for detecting when two URLs are dust. A dust rule φ is a relation over the space of URLs. φ may be a many-to-many relation. Every pair of URLs belonging to φ is called an instance of φ. The support of φ, denoted support (φ), is the collection of all its instances.

3.2 PROPOSED SYSTEM

This proposed technique uses the naval URL reduplication technique on URL data and also priorities user results based upon their geo-social data. URL de-duplication is performed through map reduce algorithm that removes duplicate data based on eliminating same type and same set of data on particular group of URL data.

Geo-social data are mined through LDA CRATS technique to find the type and which data user is searching for. This makes the result more accurate than the previous suggested works because all previous works are based on user base click and frequent mining technique which is not performing good at hadoop architectures due to its inability to process on large datasets.
MERITS:
- Web-crawler performance will be increased since it uses map reduce to remove duplicate URLs for mining
- More sufficient response time for user to get accurate results on their queries.
- Multiple reduction of alignment reduces the mapping processes between the each nodes with high impact and edge weight of the arbitrary density point that gives better results for the users.

Hadoop is one of the most popular MapReduce implementations. Both input and output pairs of a MapReduce application are managed by an underlying Hadoop distributed file system. At the heart of HDFS is a single Name Node a master server managing the file system namespace and regulates file accesses. The Hadoop runtime system establishes two processes called Job Tracker and Task Tracker. Job-Tracker is responsible for assigning and scheduling tasks; each Task Tracker handles mappers or reducers assigned by Job Tracker.

Map reduce algorithm has three important methods
- Group
- Sort
- Reduce

Grouping: Here we create groups on results that we retrieved. Those groups contains type of data we retrieved type of data category. These types are generated into separate groups for their co-works.
Sort: Sort process makes which group of data has to be displayed on top.
Reduce: This process removes the dust i.e duplicate URL present in data groups.

4.3 APPLYING LDA CRATS DATA
CRATS is that jointly mines the latent Communities, Regions, Activities, Topics, and Sentiments based on the important dependencies among these latent variables. We apply those mined data on our results to produce more accurate and needed results on URL data that retrieved.

4.4 REMOVING DUST USING GEO-SOCIAL DATA
Here we remove the unwanted results based upon the geo social data that has been got from the above CRATS data. For example if a user searches data from one particular geo location output data will search for results that appropriate to that particular location and priorities those set of URLs to be displayed on top order of the results and other valid keys such as time, type of data they will need also produces influence the resultant dataset.

4.5 ALGORITHM
Input: URL For Keywords query
Output: Survived URL sets with user attribute desired key
Step 1: Group the URL data based upon their types.
Step 2: Sort those group based upon query results.
Step 3: Remove duplicate or repeated URLs based upon their Output page results
Step 4: Get User data.
Step 5: Get Geo-social data of user location.
Step 6: Apply geo-social data on resultant data
Step 7: Prioritize URL that matches with user geo-social data.
Step 8: Finalize the results.
Step 9: Return survived URL sets.

V. RESULT AND DISCUSSIONS

We use two document collections in our experiments: GOV2.Dataset consists of a snapshot of their sources fetched from 25,205,179 individual documents from US government domains in 2014. According to the TREC track information some duplicate documents have already been removed from GOV2. The GOV2 TREC dataset contains about 3.42 million duplicate URLs divided into about 1.43 million dup-clusters. These documents were Grouped by creating a small fingerprint of their content and hashing the URLs with identical fingerprints into the same clusters is a collection of over 150 million webpages crawled from the Brazilian domain using an actual Brazilian crawling system. This crawling was performed from September to October, 2014, with no restrictions regarding content duplication or quality. To identify groups of duplicate URLs in WBR10, we adopted the same approach used by the authors in [11]. Thus, we scanned the collection to find out the web sites which explicitly indicate the canonical URLs in their pages. By doing this, we identified about 3.95 million duplicate documents in fora total of about 1.14 million dup-clusters. Although is six times larger than GOV2, it has only 15 percent more DUST identified. This was expected since webmasters are not obliged to identify canonical URLs.

5.1 Existing Method

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Method</th>
<th>Candidates</th>
<th>Valid</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOV2</td>
<td>R(Fanout-10)</td>
<td>7097</td>
<td>2242</td>
<td>31.6%</td>
</tr>
<tr>
<td></td>
<td>R(tree)</td>
<td>2458</td>
<td>718</td>
<td>29.21%</td>
</tr>
<tr>
<td></td>
<td>Duster</td>
<td>1685</td>
<td>1332</td>
<td>79.05%</td>
</tr>
</tbody>
</table>

5.2 OUR METHOD

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Method</th>
<th>Candidates</th>
<th>Valid</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOV</td>
<td>R(Fanout-10)</td>
<td>6894</td>
<td>3482</td>
<td>47.6%</td>
</tr>
<tr>
<td></td>
<td>R(tree)</td>
<td>1789</td>
<td>988</td>
<td>39%</td>
</tr>
<tr>
<td></td>
<td>Duster</td>
<td>1234</td>
<td>2857</td>
<td>89.05%</td>
</tr>
</tbody>
</table>

5.3 Graph Representation

5.3.1 EXISTING METHOD:

These two methods were chosen due to their performance in previous experiments, which indicate they represent the best options found in literature for de-duplicating URLs.

VI. CONCLUSION

Thus this paper work ‘DUST REMOVING MODULES’ has solved all the problems existed in previous systems. Since this system has log of websites then pairing the websites. Thus the user feels free to use the websites and he can be sure that his credentials have been protected. As the system gives the opportunity to change his websites. User view the original websites. The system is simple and user-friendly and they can avail the services easily.

VII. FUTURE WORK
In this paper, we discussed the development of storing the log details into the server without any duplication but the timings of the server can take the more time to do the every operation in the database so in future work we have to reduce the loading timings and efficient in the server db. as well as we have to take the server log in country wise also because it is global server log maintain we follow this algorithm in the every country server.

REFERENCES