A NEAR-DUPLICATE DETECTION ALGORITHM TO FACILITATE DOCUMENT CLUSTERING

Lavanya Pamulaparty¹, Dr. C.V Guru Rao² and Dr. M. Sreenivasa Rao³

¹Department of CSE, Methodist college of Engg. & Tech., OU, Hyderabad
²Department of CSE, S R Engineering College, JNT University, Warangal
³Department of CSE, School of IT, JNT University, Hyderabad

ABSTRACT

Web Mining faces huge problems due to Duplicate and Near Duplicate Web pages. Detecting Near Duplicates is very difficult in large collection of data like "internet". The presence of these web pages plays an important role in the performance degradation while integrating data from heterogeneous sources. These pages either increase the index storage space or increase the serving costs. Detecting these pages has many potential applications for example may indicate plagiarism or copyright infringement. This paper concerns detecting, and optionally removing duplicate and near duplicate documents which are used to perform clustering of documents. We demonstrated our approach in web news articles domain. The experimental results show that our algorithm outperforms in terms of similarity measures. The near duplicate and duplicate document identification has resulted reduced memory in repositories.

KEYWORDS

Web Content Mining, Information Retrieval, document clustering, duplicate, near-duplicate detection, similarity, web documents

1. INTRODUCTION

The WWW is a popular and interactive medium to disseminate information today. Holocene epoch has detected the enormous emergence of Internet document in the World Wide Web. The Web is huge, diverse, and dynamic and thus raises the scalability, multimedia data, and temporal issues respectively. Due to these situations, we are currently drowning in information and facing information overload [1]. In addition to this, the presence of duplicate and near duplicate web documents has created an additional overhead for the search engines critically affecting their performance [2]. The demand for integrating data from heterogeneous sources leads to the problem of near duplicate web pages. Near duplicate data bear high similarity to each other, yet they are not bitwise identical [3] [4] but strikingly similar. They are pages with minute differences and are not regarded as exactly similar pages. Two documents that are identical in content but differ in small portion of the document such as advertisement, counters and timestamps. These differences are irrelevant for web search. So if a newly-crawled page P_duplicate is deemed a near-duplicate of an already-crawled page P, the crawl engine should ignore P_duplicate and its entire out-going links (intuition suggests that these are probably near-duplicates of pages reachable from P) [4, 35]. Near Duplicate web pages from different mirrored sites may only differ in the header or footnote zones that denote the site URL and update time [5].

DOI : 10.5121/ijdkp.2014.4604
Duplicates and Near Duplicate Web pages are creating large problems for web search engines like they increase the space needed to store the index, either slow down or increase the COST of saving results and annoy the users [34]. Elimination of near-duplicates saves network bandwidth, reduces storage costs and improves the quality of search indexes. It also reduces the load on the remote host that is serving such web pages [10].

The determination of the near duplicate web pages [28-29] [21] aids the focused crawling, enhanced quality and diversity of the query results and identification on spam. The near duplicate and duplicate web page identification helps in Web mining applications for instance, community mining in a social network site [20], plagiarism detection [24], document clustering [29], collaborative filtering [30], detection of replicated web collections [31] and discovering large dense graphs [34]. NDD removal is required in Data Cleaning, Data integration, Digital libraries and electronic published collections of news archives. So in this paper, we propose a novel idea for finding near duplicate web pages from a huge repository.

2. RELATED WORK

The proposed research has been motivated by numerous existing works on and near duplicate documents detection. Duplicate and near-duplicate web pages are creating large problems for web search engines: they increase the space needed to store the index, either slow down or increase the cost of serving results, and annoy the users. This requires the creation of efficient algorithms for computing clusters of duplicates [4, 6, 7, 16, 21, 22, 29, 30]. The first algorithms for detecting near-duplicate documents with a reduced number of comparisons were proposed by Manber [25] and Heintze [17]. Both algorithms work on sequences of adjacent characters. Brin [40] started to use word sequences to detect copyright violations. Shiva Kumar and Garcia-Molina [31] continued this research and focused on scaling it up to multi-gigabyte databases. Broder et al [5] defined two concepts resemblance and containment to measure the similarity of degree of two documents. He used word sequences to efficiently find near-duplicate web pages.

The dimensionality reduction technique proposed by Charikar's Simhash [34] is to identify near duplicate documents which maps high dimensional vectors to small-sized fingerprints. They developed an approach based on random projections of the words in a document. Henzinger [9] compared Broder et al.'s [7] shingling algorithm and Charikar's [34] random projection based approach on a very large scale, specifically on a set of 1.6B distinct web pages.

In the syntactical approach we define binary attributes that correspond to each fixed length substring of words (or characters). These substrings are a framework for near-duplicate detection called shingles. We can say that a shingle is a sequence of words. A shingle has two parameters: the length and the offset. The length of the shingle is the number of the words in a shingle and the offset is the distance between the beginnings of the shingles. We assign a hash code to each shingle, so equal shingles have the same hash code and it is improbable that different shingles would have the same hash codes (this depends on the hashing algorithm we use). After this we randomly choose a subset of shingles for a concise image of the document [6, 8, and 9]. M. Henzinger [32] uses like this approach AltaVista search engine. There are several methods for selecting the shingles for the image: a fixed number of shingles, a logarithmic number of shingles, a linear number of shingle (every nth shingle), etc. In lexical methods, representative words are chosen according to their significance. Usually these values are based on frequencies. Those words whose frequencies are in an interval (except for stop-words from a special list of
about 30 stop-words with articles, prepositions and pronouns) are taken. The words with high frequency can be non informative and words with low frequencies can be misprints or occasional words.

In lexical methods, like I-Match [11], a large text corpus is used for generating the lexicon. The words that appear in the lexicon represent the document. When the lexicon is generated the words with the lowest and highest frequencies are deleted. I-Match generates a signature and a hash code of the document. If two documents get the same hash code it is likely that the similarity measures of these documents are equal as well. I-Match is sometimes unstable to changes in texts [22]. Jun Fan et al. [16] introduced the idea of fusing algorithms (shingling, I-Match, simhash) and presented the experiments. The random lexicons based multi fingerprints generations are imported into shingling based simhash algorithm and named it "shingling based multi fingerprints simhash algorithm". The combination performance was much better than original Simhash.

3. ARCHITECTURE OF PROPOSED WORK

The paper proposed the novel task for detecting and eliminating near duplicate and duplicate web pages to increase the efficiency of web crawling. So, the technique proposed aims at helping document classification in web content mining by eliminating the near-duplicate documents and in document clustering. For this, a novel Algorithm has been proposed to evaluate the similarity content of two documents.

![Proposed Architecture](image)

**Figure 1. Proposed Architecture**
3.1. Architectural Steps

Architectural steps which includes: (i) Web page Dataset Collection (ii) Pre-processing (iii) Store in database (iv) Rendering DD Algorithm (v) Verifying the similarity content (vi) Filtering Near Duplicates (vii) Refined Results

3.1.1. Dataset Collection

We have collected data using Wget, a software program that retrieves content from webservers and have gathered our data sets. Here, we have collected by giving some URLs of the news websites which usually have replicas, so we can get more number of web documents with relevant information. Wget can optionally work like a web crawler by extracting resources linked from HTML pages and downloading them in sequence, repeating the process recursively until all the pages have been downloaded or a maximum recursion depth specified by the user has been reached. The downloaded pages are saved in a directory structure resembling that on the remote server.

3.1.2. Parsing of Web Pages

Once a page has been crawled, we need to parse its content to extract information that will feed and possibly guide the future path of the crawler. Parsing might also involve steps to convert the extracted URL to a canonical form, remove stop words from the page's content and stem the remaining words [33]. HTML Parsers are freely available for many different languages. They provide the functionality to easily identify HTML tags and associated attribute-value pairs in a given HTML document.

3.1.3. Stop-listing

When parsing a Web page to extract content information or in order to score new URLs suggested by the page, it is often helpful to remove commonly used words or stop words such as “it” and “can”. This process of removing stop-words from text is called stop-listing [26].

3.1.4. Stemming Algorithm

Stemming algorithms, or stemmers, are used to group words based on semantic similarity. Stemming algorithms are used in many types of language processing and text analysis systems, and are also widely used in information retrieval and database search systems [25]. A stemming algorithm is an algorithm that converts a word to a related form. One of the simplest such transformations is conversion of plurals to singualars, another would be the derivation of a verb from the gerund form (the "-ing" word). A number of stemming or conflation algorithms have been developed for IR (Information Retrieval) in order to reduce morphological variants to their root form. A stemming algorithm would normally be used for document matching and classification by using it to convert all likely forms of a word in the input document to the form in a reference document [22]. Stemming is usually done by removing any attached suffixes, and prefixes from index terms before the assignment of the term. Since the stem of a term represents a broader concept than the original term, the stemming process eventually increases the number of retrieved documents [23].
### 3.1.5. Duplicate Detection (DD) Algorithm

Step 1: Consider the Stemmed keywords of the web page.

Step 2: Based on the starting character i.e. A-Z we here by assumed the hash values should start with 1-26.

Step 3: Scan every word from the sample and compare with DB (data base) (initially DB Contains NO key values. Once the New keyword is found then generate respective hash value. Store that key value in temporary DB.

Step 4: Repeat the step 3 until all the keywords get completes.

Step 5: Store all Hash values for a given sample in local DB (i.e. here we used array list)

Step 6: Repeat step 1 to step 6 for N no. of samples.

Step 7: Once the selected samples were over then calculate similarity measure on the samples hash values which we stored in local DB with respective to webpages in repository.

Step 8: From similarity measure, we can generate a report on the samples in the score of % forms. Pages that are 80% similar are considered to be near duplicates.

---

**Figure 2. DD algorithm Work flow**

![DD Algorithm Workflow Diagram]

---

43
This algorithm consists of three major steps. The first step is to calculate the hash values of the each keyword in a web page \( w_1 \). Then in the second step they are stored in a temporary data base. This procedure is repeated for all the samples collected. The third step is to calculate the similarity of the pages on the hash values generated which are stored in a local DB (data base). If the threshold is above 80% then it is considered as a near duplicate web page. Whenever a new page \( w_n \) is coming and needs to be stored in the repository, its similarity with the web pages already existing in the repository needs to be compared. Similarity is calculated using the following formula.

\[
\text{SimScore} = \frac{\text{Number of hash values in } w_1, w_2, \ldots, w_n}{\text{Total } w_1, w_2, \ldots, w_n \text{ under consideration in Repository}} \times 100
\]

4. EXPERIMENTAL ENVIRONMENT AND SETUP

The proposed near duplicate document detection system is programmed using Java (jdk 1.6) and the backend used is MS Access. The experimentation has been carried out on a 2.9 GHz, i5 PC machine with 4 GB main memory running a 32-bit version of Windows XP. The Web pages are collected using Wget computer program.

Table 1. Web pages dataset details

<table>
<thead>
<tr>
<th>S. No</th>
<th>Documents data set</th>
<th>Number of Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Current Affairs</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Technical articles</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Shopping sites articles</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Raw files</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>59</td>
</tr>
</tbody>
</table>

Figure 3. Data Collection using Wget tool.
5. RESULTS

Every dataset mentioned in the above section undergoes web page parsing, stop word removal and stemming process. It is observed that after applying the above mentioned steps the no. of keywords are reduced to an extent. The final output that is stemmed keywords are provided to the DD algorithm. The results are obtained by executing the DD program on the datasets which are shown in Table 2 and Table 3. These are representing the outcome obtained by checking every page of the dataset with all web pages from the database. As shown in Fig. 4 and Fig. 5, SimScore range is divided into 4 categories that <60%, 60% to 70%, 70% to 80% and >80%.

Table 2. Samples outcomes

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Para 1:</td>
<td>Para 1:</td>
</tr>
<tr>
<td>23519 5125 31513 20158 31597 5245 14215 16119 12118</td>
<td>20146 19572 20158 31597 23519 3153 3153 5235 14215 16119 12118</td>
</tr>
<tr>
<td>14215 16155 6125 15169 20154 1597 15135 14215 31613 61624 1195</td>
<td>14215 16155 6125 15169 20154 1597 15135 14215 31613 61624 1195</td>
</tr>
<tr>
<td>21444 31315 31245 15169 15195 61624 31513 15151 15119</td>
<td>21444 31315 31245 15169 15195 61624 31513 15151 15119</td>
</tr>
<tr>
<td>12431 31290 21013 15154 5145 5145 5415 5145 51912</td>
<td>12431 31290 21013 15154 5145 5145 5415 5145 51912</td>
</tr>
<tr>
<td>45223 11220 5145 19153 19191 11812</td>
<td>45223 11220 5145 19153 19191 11812</td>
</tr>
<tr>
<td>Para 2:</td>
<td>Para 2:</td>
</tr>
<tr>
<td>20158 32539 16189 23539 12534 14215 52416 18531 14221</td>
<td>20158 32539 16189 23539 12534 14215 52416 18531 14221</td>
</tr>
<tr>
<td>18514 16189 15165 19151 15159 16171</td>
<td>18514 16189 15165 19151 15159 16171</td>
</tr>
</tbody>
</table>
Table 3: DD results for selected dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of documents</th>
<th>&lt; 60%</th>
<th>60% to 70%</th>
<th>70% to 80%</th>
<th>&gt;80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>w10, w5, w8</td>
<td></td>
<td>w4, w6, w7</td>
<td>w3, w5</td>
<td>w2</td>
</tr>
<tr>
<td>w2</td>
<td>w10, w8, w9</td>
<td></td>
<td>w4, w6, w7</td>
<td>w3, w5</td>
<td>w1</td>
</tr>
<tr>
<td>w3</td>
<td>w4, w8, w9, w10</td>
<td>w6, w7</td>
<td>w1, w2, w5</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>w4</td>
<td>w3, w5, w8, w9, w10</td>
<td>w1, w2, w6, w7</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>w5</td>
<td>w4, w8, w9, w10</td>
<td>w6, w7</td>
<td>w1, w2, w3</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>w6</td>
<td>w7, w8, w9, w10</td>
<td>w1, w2, w4, w3</td>
<td>w5</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>w7</td>
<td>w6, w8, w9, w10</td>
<td>w1, w2, w3, w4, w5</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>w8</td>
<td>w1, w2, w3, w4, w5, w6, w7, w9, w10</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w9</td>
<td>w1, w2, w3, w4, w5, w6, w7, w8, w10</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w10</td>
<td>w1, w2, w3, w4, w5, w6, w7, w8, w9</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Simscore Illustrations for the datasets

6. CONCLUSIONS AND FUTURE WORK

Web pages with similarity score of 0% represent totally unique documents. Web pages with similarity score of <60% are not similar and needs to be maintained in the database for future reference. Web pages with similarity score of 60% to 70% are suspicious near duplicates and needs to be maintained in the database for future reference. Web pages with similarity score of more than 70% are almost near duplicates.

Furthermore only top N pages are stored in the repository for each datasets instead of all documents which reduces memory space for the repository. Since web pages having SimScore greater than 60% are not stored in the repository, by detecting them as near duplicate web pages. The search efficiency and effectiveness will be achieved and document clustering can be facilitated.
Web pages which are near duplicates may appear closer to each other in search results, but provide very little benefit to the user. The future work will be research for more robust and accurate methods for near duplicate detection and elimination on basis of the detection. To increase the accuracy and effectiveness of the DD Algorithm a clustering mechanisms can also be applied to perform the effective document clustering.

REFERENCES

[16] Jun Fan, Tiejun Huang “A fusion of algorithms in near duplicate document detection”.


AUTHORS

Lavanya Pamulaparty, Associate Professor and Head of the Dept. Dept of CSE, MCET, Osmania University, Hyderabad, obtained her Bachelor’s degree in computer science from Nagpur University of KITS, Nagpur, India, and Masters Degree in Software Engineering from School of Informatics from JNT University Hyderabad, India. Her research interests include information storage and retrieval, Web Mining, Clustering technology and computing, performance evaluation and information security. She is a senior member of the ACM, IEEE and Computer Society of India.

Dr. Guru Rao C. V received his Bachelor’s Degree in Electronics &Communications Engineering from VR Siddhartha Engineering College, Vijayawada, India. He is a double post graduate, with specializations in Electronic Instrumentation and Information Science & Engineering. He is a Doctorate holder in Computer Science & Engineering from Indian Institute of Technology, Kharagpur, India. With 24 years of teaching experience, currently he is the Professor & Principal, SR Engineering College Warangal, India. He has more
than 25 publications to his credit. He is a life member of Indian Society for Technical Education, Instrumentation Society of India and member of Institution of Engineers, Institution of Electronics & Telecommunications Engineers and Institution of Electrical & Electronics Engineers (USA).

Dr. M Sreenivasa Rao, Professor, School of Information Technology, JNT University, Hyderabad, obtained his Graduation and Post-graduation in Engineering from JNT University, Hyderabad and Ph D from University of Hyderabad. Over 28 Years of IT Experience in the Academia& industry. As a Dean of the MS IT Program, in association with Carnegie Melon University, USA. De- signed and conducted post graduations level MSIT program. Guided more than 10 research students in JNTU, and continuing the research in IT.