International Journal of Computational Intelligence Systems, Vol. 6, No. 1 (January, 2013), 1-13

Near-Duplicate Web Page Detection: An Efficient Approach Using Clustering, Sentence Feature and Fingerprinting

J. Prasanna Kumar*
MVSR Engineering College, Osmania University, India
prasannakumarphd@gmail.com

P. Govindarajulu
Department of CS, S.V.U. College of CMIS, Sri Venkateswara University, India

Received 27 December 2011

Accepted 27 July 2012

Abstract

Duplicate and near-duplicate web pages are the chief concerns for web search engines. In reality, they incur enormous space to store the indexes, ultimately slowing down and increasing the cost of serving results. A variety of techniques have been developed to identify pairs of web pages that are “similar" to each other. The problem of finding near-duplicate web pages has been a subject of research in the database and web-search communities for some years. In order to identify the near duplicate web pages, we make use of sentence level features along with fingerprinting method. When a large number of web documents are in consideration for the detection of web pages, then at first, we use K-mode clustering and subsequently sentence feature and fingerprint comparison is used. Using these steps, we exactly identify the near duplicate web pages in an efficient manner. The experimentation is carried out on the web page collections and the results ensured the efficiency of the proposed approach in detecting the near duplicate web pages.

Keywords: Web Crawling, Web page, Duplicate web page, Near duplicate web page, Near duplicate detection, fingerprinting.

1. Introduction

Web Mining is the branch of data mining which deals with the study of World Wide Web [9]. It refers to the use of data mining techniques to automatically find out and extract information from World Wide Web documents and services. The origin of web mining can be referred from the concepts from various areas such as Data Mining, Internet technology and World Wide Web, and recently, Semantic Web [10, 9, 11, 12, 13 and 38]. A program or automated script that traverses the World Wide Web in a systematic and automated manner is said to be a web crawler or web spider or web robot [14]. A URL selected by the crawler from the frontier, downloads the web resource, collects URLs from the downloaded resource and adds the new URLs to the frontier. The crawler proceeds in this manner till the frontier is empty or some other condition causes it to stop [15], [16]. Duplicate and near duplicate web page detection becomes an important step in web Crawling [5]. Web crawlers are become an unavoidable part of all search engines and are highly becoming important in data mining and other indexing applications [17]. Web crawling is engaged by the search engines to populate a local indexed repository of web pages which is in turn utilized to answer user search queries [18]. Such search engines depend on huge collections of web pages that are obtained with the help of web crawlers, which traverse the web by subsequent hyperlinks and storing downloaded pages in a large database which is later pointed for efficient execution of user queries [17].

Duplicate web pages and mirrored pages are seen in plenty in the web. Even with the point that near duplicates are not bit wise identical, they are blindingly alike and differ in slight extents of the document like the advertisements, counters and timestamps [19]. Typographical faults, versioned, mirrored or imitative web pages, several depictions of the same physical object, spam emails made from the same template, and many such conditions may also end up in producing a

near duplicate web page [9]. Removal of near duplicates results in preservation of network bandwidth, decrease in storage costs and improvement in the quality of search indexes. Moreover, the load on the remote host that serves the web pages is also reduced. Near duplicate webpage detection systems are susceptible to several challenges. Prime concern is about the scale of webpages, where the search engine index has billions of web-pages, which leads to having a multi-terabyte database. Second issue is the capability of the crawl engine to crawl billions of web-pages [11]. For every single web page, calculating a fingerprint that is a succinct (say 64-bit) digest of the characters on that webpage is the modest method for detecting duplicates. When the fingerprints of two webpage documents are the same, at that point we have to examine if the pages are the same and if so, and then state that one of those to be a duplicate copy of the other. The check summing techniques are used to find out the web pages that are exact duplicates of each other due to mirroring or plagiarism [4].

A number of applications are benefited by the detection of the near duplicates [36]. The determination of the near duplicate web pages helps the following the focused crawling, enhanced quality and diversity of the query results and identification on spam [20], [21], [2]. Perfectly and adeptly determined near duplicates are relied on different web mining applications, for example, document clustering [3], collaborative filtering [25], detection of replicated web collections [26], discovering large dense graphs [34], detecting plagiarism [31] and community mining in a social network site [32]. The removal of the near duplicate pages [33] helps in reduced storage costs and improved quality of search indexes in addition to considerable bandwidth conservation. Above all, the crawled web pages are preprocessed using document parsing, that eliminates the HTML tags and java scripts present in the web documents, and which is followed by the removal of common words or stop words from the crawled pages. Then the stemming algorithm is used to filter the prefixes and the suffixes in the crawled documents for obtaining the keywords [15]. The next task is to arrange the large number of documents into meaningful clusters so document clustering can be engaged to browse a set of documents or to arrange the results given by a search engine in answer to a user's query. This can considerably improve the accuracy and recall in information retrieval systems, and it is a better way to find the nearest neighbors of a document [27].

Here, we formulate an efficient approach for near duplicate web page detection based on a similarity measure using fingerprint. The objective of this research is to effectively avoid the near duplicates which results in better conservation of network bandwidth and reduction in storage costs. In the proposed approach, initially the preprocessing takes place. There web crawled web pages are preprocessed using a parsing technique to remove the HTML tags, where java scripts and other less relevant data present in the web pages and then, we remove the stop words from the crawled web pages. Finally, the stemming process can be done, where words are converted to base form and keywords are extracted from the crawled Web pages for further processing. Then, cascade filtering is applied, which includes sentence level extraction and fingerprint as the filtering techniques. Finally, a bit-by-bit difference measure is calculated between the fingerprint of the crawled Web page and other web pages. If the bit-by-bit difference measure computed is less than a predefined threshold value, the web page crawled is termed as a near-duplicate. Otherwise, the Web page is added to the database.

The main contributions of the paper are we have added two methods for the near duplicate detection of web pages. The main process, which constitutes the proposed method, is the sentence level feature extraction, cascade filtering and finger printing. In the case of large dataset the proposed method incorporates clustering for the efficient processing.

The rest of the paper is organized as follows: In Section 2, a brief review of the latest researches related to near-duplicate web page detection has been provided. Section 3 describes the novel approach for the detection of near-duplicate Web pages using cascade filtering which employs sentence level feature and the fingerprint. Enhancement of our proposed approach incorporating clustering for the detection of near-duplicate web pages is described in Section 4. The experimental results are given in Section 5 and conclusions are summed up in Section 6.

2. Review of Related Works

Recently, near duplicate web page detection techniques have gained high popularity and lot of researches is concentrated in this core area. This literature survey

presents a number of researches that deal with the detection of near duplicate web documents.

V. A. Narayana *et al.* [1] have presented an efficient approach for the detection of near duplicate web pages in web crawling where, the keywords were extracted from the crawled pages and the similarity score between two pages was calculated. A document which is considered as near duplicate if it having similarity score is greater than a threshold. Monika Henzinger [2] have performed a comparison study over the two "state-of-the-art" algorithms, Broder *et al.'s* [3] shingling algorithm and Charikar's [4] random projection based approach for finding near-duplicate web pages. Both algorithms were used either for development or by popular web search engines. They compared the two algorithms on a very large scale, namely on a set of 1.6B distinct web pages. The results showed that both of the algorithms were failed in finding near-duplicate pairs on the same site, while both provide high precision for near-duplicate pairs on different sites. Since Charikar's algorithm obtained a lot of near-duplicate pairs on different sites that possessed a better precision overall, namely 0.50 versus 0.38 for Broder et al.'s algorithm. They have produced a combined algorithm which obtains a precision of 0.79 for 79% of the recall of the other algorithms.

In the case of developing a near-duplicate detection system for a multi-billion page repository, Gurmeet Singh Manku *et al*. [5] conducted two research contributions. Initially, they stated that Charikar's fingerprinting technique [4] was suitable for their goal. After that, they showed an algorithmic technique for identifying existing f-bit fingerprints which differ from a given fingerprint in at most k bit-positions, for small k. Their technique works well for both online queries and all batch queries. Experimental evaluation using real data proved the practicality of their design.

A. Kolcz and A. Chowdhury [6] concentrated on I-Match and offered a randomization-based method of improving its signature stability with the suggested methodology steadily outdoing traditional I-Match by having 40 to 60% in terms of the relative increase in near-duplicate recall. Notably, the great improvements in detection accuracy were offset by only slight surges in computational requirements. They also tackled the secondary difficulty of false matches, which was very vital for I-Match when it comes to fingerprinting lengthy documents. Identification of near-duplicate Web pages was definitely thought-provoking in the web-scale due to the huge data. Hence, a mechanism needs to be presented for sensing the duplicate data so that related search results can be provided to the user. Ranjna Gupta *et al*. [7] presented a design that presents methods that will work in both online and offline depending on favored and disfavored user inquiries to identify duplicates and near duplicates. Hung-Chi Chang and Ten-Hour Wang [8] have offered a sentence-level statistics-based method to identify near-duplicate documents, which is language independent, modest but effective. The experimental outcomes exhibited high efficiency and good accuracy of the proposed approach in detecting near-duplicates in news archives.

Hannaneh Hajishirzi *et al.* [28] developed a near-duplicate document detection method which can be used for a particular domain. In their method, they considered every document as a real-valued sparse k-gram vector, in which the weights were learned to optimize for a particular similarity function, such as the cosine similarity or the Jaccard coefficient. In addition, for efficient similarity computation, these vectors were mapped to a small number of hash-values as document signatures through locality sensitive hashing scheme. Bingfeng Pi *et al.* [29] suggested the seriousness of existence of near duplicate. Then, they demonstrated how a SimHash works and its merits for finding near-duplicate and reviewed a series of findings, including the problem itself and the benefits obtained by SimHash-based approach.

3. Proposed Approach For Near Duplicate Detection Using Cascade Filtering

This section deals with an approach for the detection of near duplicate web pages. Duplicate documents are frequently found in huge databases of digital documents such as the government declassification effort or in digital libraries. The occurrence of near duplicates increase the space required to store the index, decreases serving results, and irritate the users [11]. Capable duplicate document detection is important not only to allow querying for similar documents, but also to filter out unnecessary information in the huge document databases [9]. The removal of the near duplicate pages provides reduced storage costs and improved quality of search indexes in addition to considerable bandwidth management. In the proposed approach, we are using sentence level extraction and simhash algorithm [4] to

detect near duplicates. Our proposed approach reduces the storage space greatly and allows quicker comparison and search. Initially, the crawled web documents are parsed, it includes the removal of HTML tags, java scripts, stop words/common words and stemming of remaining words. Then, the sentence level feature and simhash is applied to detect and eliminate the near duplicates. K-Mode clustering technique [37] is added in our proposed approach to get faster and better results. The important steps used in the proposed approach for near-duplicate web page detection is given in fig 1.

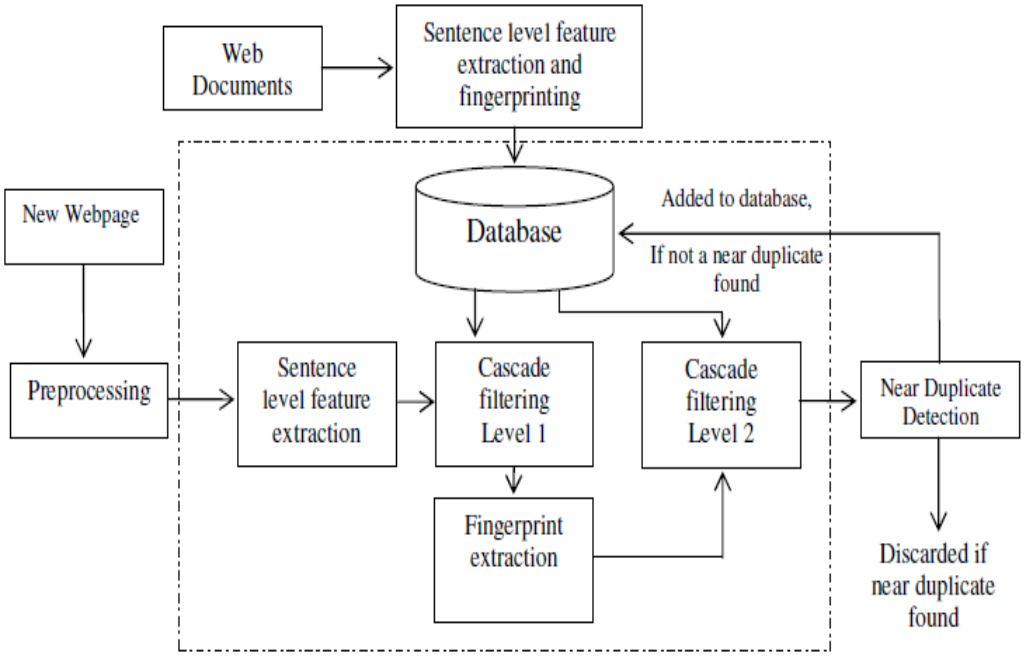


Fig.1. Steps used in the proposed approach for near-duplicate web page detection

3.1Preprocessing

Preprocessing is done prior to the sentence level feature comparison and fingerprint comparison. Preprocessing converts a web page into a set of keywords. The preprocessing consists of crawling, parsing, stop word removal and stemming. In crawling, the web pages are crawled from the data base. In parsing, the web pages are represented by a linear structure with a given grammar. The parsed web documents are then subjected to the stop word removal process, where connecting words such as “is”, “as”, etc. are removed. The words in the document are converted into its basic form by a process, called stemming. The preprocessed web pages are passed to cascade filters for further processes.

a.Web Crawling

A web crawler is a comparatively an automated program, or script which methodically scans or "crawls" over Internet pages to build an index of the data that is in concern. Web spider, web robot, crawler, and automatic indexer are the other names for web crawler. The common use of the web crawling is with search engines. Search engines force the web crawlers to collect information about what needed on public web pages. Their chief goal is to collect data so that when Internet surfers give a search term on their site, they can quickly give the surfer with applicable web sites. In spite of the large number of applications for Web crawlers, at the center they are all basically the same. Following is the main process through which Web crawler’s work [17]: (1) Downloading the Web page, (2) Parse through the downloaded page and retrieve all the links and (3) For each link retrieved, repeat the process. The process will continue till the crawler get shutdown [22]. A crawling loop comprises of obtaining a URL from the queue, downloading the equivalent file with the help of HTTP, traversing the page for finding a new URLs and including the unvisited URLs to the queue [35].

b. Parsing

Parsing is the procedure of structuring a linear representation according to a given grammar [24]. After fetching a webpage, we have to parse its content to extract information that will feed and possibly guide the future path of the crawler. In parsing, it may imply simple hyperlink/URL extraction or it may involve the more complex process of tidying up the HTML content in order to analyze the HTML tag tree. Parsing also involves the steps to convert the extracted URL to a canonical form, and it will remove stop words and perform stemming on the remaining words.

c. Stop Word Removal

In a web document, there are commonly utilized words that carry less important meaning than keywords, hence it is necessary and beneficial to remove these words. Usually search engines remove these commonly utilized words or rather known as 'stop words' from a keyword phrase to return the most relevant result. In searching, all stop words, for instance, most used words like 'a' and 'the', are detached from multiple word queries for increasing search performance [9]. Stop words like "it", "can", "an", "and", "by", "for", "from", "of", "the", "to", "with" are the common stop words. Stop word removal is done while parsing a document to obtain information about the content or while scoring fresh URLs that the page recommends.

d. Stemming

In many cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of many applications. For this reason, a lot of stemming Algorithms or stemmers has been developed to reduce a word to its stem or root form. The stems are used to represent the key terms of a query or document instead of the original word. Lemmatization [30] is an algorithm which attempts to convert a word to its linguistically correct root which ultimately facilitates the reduction of all words, possessing an identical root to a single one. This is obtained by removing each word of its derivational and inflectional suffixes [23]. For instance, "orient," "oriented" and "orientation" are all condensed to "orient", which is the base form, similarly "runs"," run " are all condensed into "run".

3.2 Cascade Filtering

After the preprocessing steps, the designed cascade filtering is used for finding the duplicate web pages. The proposed cascade filtering contains two filtering techniques that are applied serially one after the other. The methods used in our approach are sentence level feature comparison and fingerprint comparison. The reason behind using these methods are to make use of time reduction in execution by the use of sentence level feature filtering technique and increasing the precision of the result by the use of fingerprinting comparison technique. By combining these methods, we can obtain an effective, precise and less time consuming result. In sentence level feature comparison, it compares only the sentence features rather than the keywords, and hence the time gets reduced. Applying this sentence level feature to a very large number of web pages, it filters out web documents which do not satisfy the criteria, resulting in lesser number of documents filtered in for the subsequent processes and hence, acting as a filter and saving time. In the fingerprint technique, fingerprint of the document is computed by using Sim-Hash algorithm and comparison is made in order to detect the near duplicates.

a. Sentence Level Feature Extraction

Sentence level feature extraction is the first filtration technique that is a key feature used for near duplicate web page detection. Here, total number of sentences in a web document is used as a feature vector. Let S_{di} be the total number of sentences in the d_i th document and S_k be the total number of sentences in the newly added web page. The newly added webpage is compared with each of the webpage in the database and the comparison is made by extracting the total number of sentences in each of the web document. If the new webpage and the compared webpage differ in the total number of lines in the web page by a number which is less than the sentence threshold (T_s) $|S_{di} - S_k| < T_s$, then the compared page is filtered in. This action results in narrowing of number of web document input to the second level of filtering and hence it acts as a filter. Here, only numerical values are compared rather than the string value so it lead to the less computation time and also, consume less disk space.

By this process, we can eliminate the far similar documents. Hence, the number of inputs to the fingerprint comparison will get reduce, so that

fingerprint of only a limited number of web page is to be found out, rather than all the web pages in the database reducing the time incurred. Subsequently, we can find the fingerprint by applying simhash algorithm for each of the d_i filtered in.

b. Fingerprint Comparison

Fingerprints computed from a web page acts as a second filter in series with the sentence level comparison filter. In fingerprint computation, we use Sim-Hash algorithm for detection of near duplicates. Sim-Hash has two important but somewhat conflicting properties that make it an ideal technique for the detection of near duplicates. They are: (1) The fingerprint of a document is a "hash" of its features, and (2) Similar documents have similar hash values. Thus, it aid in determining whether the two documents are similar or not by comparing their corresponding hash values. Initially, it converts each of the web documents that are filtered in into a set of keywords. Each keyword is tagged with its weight, which is the number of times the keyword appears in the document. Then, we transform such a high dimensional vector into a $f-$ bit fingerprint, where' f ' is quite small compared with the original dimensionality.

Here, let the input document ' D ' be pre-processed and composed with a series of keywords. We initialize a $f-$dimensional vector V with each dimension as zero. Then, for each keyword of the document, it is hashed into a $f-$ bit hash value. These f bits increment or decrement the f components of the vector by the weight of that word. Finally, the signs of the components determine the corresponding bits of the final fingerprint of the document. The process is repeated for each of the documents and comparison is made of fingerprint of the new webpage to those web pages filtered in order to detect the near duplicates. The pseudo code of the simhash algorithm is given in fig 2 and the detailed process is given in fig 3.

```
SimHash (document D)
{
   Init vector Sim[0…(f - 1)] = 0;
   For (each keyword K_w in document D) Do
        K_w is hashed into f bit Hash value X;
              For (i = 0; i < f; i + +) Do
                    If(X[i]== 1) Then
                          Sim[i] = Sim[i] + weight (K_w);
                    Else
                          Sim[i] = Sim[i] - weight (K_w);
              endfor
   endfor
   For (i = 0; i < f; i + +) Do
        If(Sim[i] > o) Then Sim[i] = 1;
        Else Sim[i] = 0;
   endfor
}
```

Fig.2. Pseudo code of the simhash algorithm

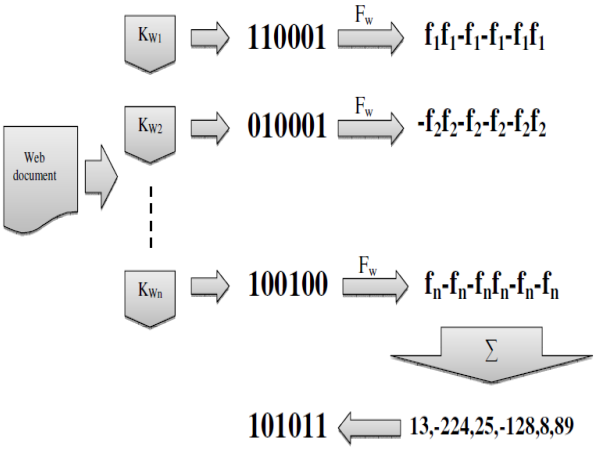


Fig.3. Working procedure of the simhash algorithm

3.3 Near Duplicate Detection

After the cascade filtering, we obtain the fingerprint of all the documents filtered in and the new web document which was inputted for checking if it's a near duplicate of the existing web pages in the database. A threshold is set for the comparison process, which is usually a user input. The fingerprint of the new web document is compared with the fingerprints of the other documents and comparison is made bit by bit. The bit-by-bit difference (b_f) in number of bits, known as bit-by-bit difference measure is taken into account and used to check if two web pages in consideration are near

duplicates or not. If the bit-by-bit difference is lesser than or equal to the threshold (T_B), then the document is termed near duplicates and is discarded. Similarly the newly added web page is compared with all the other webpage fingerprints and if it is found that the newly added web page is not a near duplicate to any of the web pages then, the new page is selected and added to the database.

Considering the following finger prints, in which one is that of the new webpage's fingerprint and other is that of some other web page in the database,

b_f

1 1 0 0 1 0

1 0 1 0 1 0

In the above fingerprints, the bit-by-bit difference b_f is 2 as the second and third bits are showing difference. Suppose, we provide threshold, $T_B = 3$, then

$b_f < T_B$ (Threshold, $T_B = 3$)

Hence, the new web document is discarded as the web pages are near duplicates as the bit difference falls below the threshold set.

Now consider the case,

b_f b_f

1 0 0 1 1 0

1 1 1 0 1 1

Here, $b_f > T_B$ (Threshold, $T_B = 3$)

Hence, the web document is selected and added to the database.

4. Enhancement of Our Proposed Approach

In databases that consist of large amount of web documents, it is a tedious process for our proposed approach and consumes time to find the near duplicates of the web page, where it compares with all other web pages in the database. In order to overcome this difficulty some specified web documents have to be selected based on some similarity measure and compared with those selected, rather than comparing with all the documents. Clustering is an effective mechanism to group large number of documents into small clusters according to a specified criterion. So, here, we select K-mode clustering [37] for grouping the web documents. We proceed by extracting the keywords from each of the web documents and then clustering on the basis of the keywords. New web page is grouped with a cluster which has most similarity. Finally, we have done the comparison procedure consists of previously specified filtering and near duplication technique. It is done only with the documents inside the cluster, which aiding in easy computation. The procedure used in the enhancement of our proposed approach in fig 4.

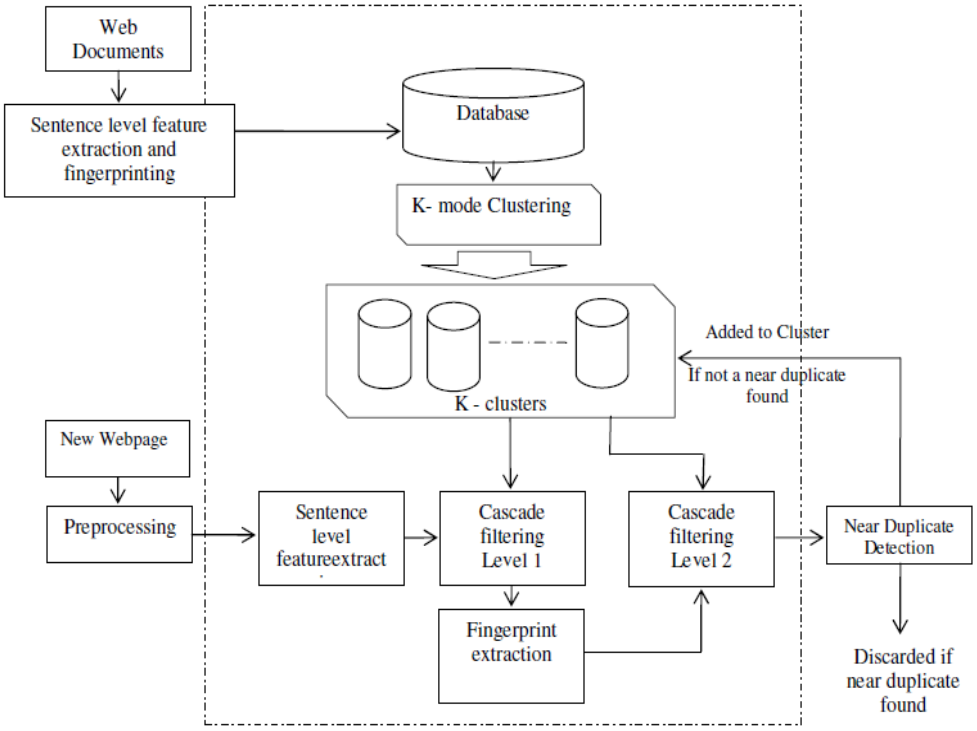


Fig.4. Procedure used in the Enhancement of our Proposed Approach

4.1 Clustering Of Large Webpage Documents

Clustering is the process of grouping the data according to some similarity measure. Here, the large web documents in the database are represented by a group of cluster so that near duplicate detection can be done only on the web pages given in the most relevant clusters. This process automatically reduces the time complexity compared with previous approaches. For clustering a large number of web documents, at first, we extract the keywords that are a set of significant words in a document describing the content of the web pages. Let, $O_s = \{d_1, d_2, \ldots\ldots\ldots\ldots, d_n\}$ represents the web page document obtained after the preprocessing steps, where, n is the total number of web page documents. Then, keywords are extracted from each web page document denoted as, $d_i = \{k_{w1}, k_{w2}, \ldots\ldots\ldots\ldots, k_{wm}\}$, where m is the total number of keywords in the we page document, d_i. Subsequently, for each keywords extracted from web page document d_i, we find the frequency of each unique keywords, $d_i = \{(k_{w1}, f_1), (k_{w2}, f_2), \ldots\ldots\ldots\ldots, (k_{wl}, f_l)\}$, Where l is the total number of unique keywords and f_l corresponds to the frequency of the l^{th} keyword.

The unique keywords are sorted in the decreasing order with respect to the frequency of the keywords. Finally, we select the top f keywords, called as representatives of each document (R_i). Then, these representatives are used to find the cluster of large set of web documents. The basic steps used in clustering is described as: (1) Initialize k-centroids, one for each cluster, (2) Compute the similarity $s(R_i, d_l),\ i = 1,2,\ldots,k$ of each k-centroids with the representatives of each web page document, (3) Assign web page document d_i to cluster C_l whose similarity measure is high. (4) Update the k representatives (5) Repeat Step 2 to step 4, until there is no movement of the web page documents between the clusters.

4.2 Detection Of Near Duplicate Webpages

After applying k-Means to large set of web documents, it result in 'k' number of clusters such a way that web documents belonging to each cluster have maximum similarity based on the keywords present in the respective documents. When a new crawled webpage is added, its similarity with all existing k cluster representatives is checked and is added to the cluster

which has the most similarity. The advantage of incorporated clustering is ease of computation and saving time. Because, when a new webpage is added, it has to compare sentence level feature with only those in within a cluster, and not with all the web pages. After adding the new web page to the respective cluster, cascade filtering is employed using the sentence level extraction technique and the fingerprint. The resultant fingerprint of the new webpage is compared with other webpage fingerprints in the cluster and the bit-by-bit difference measure, b_f is calculated, which is the bit difference. If the calculated bit-by-bit difference measure b_f is more than a preset threshold value T_B it is considered near duplicate web pages and neglected. Otherwise, the new webpage is added to the repository.

5. Results and Discussion

This section presents the results and discussion of the proposed approach to find the near-duplicate web pages. We experiment our proposed approach in a system configured with Intel core 2 duo processor, 3 GB RAM and 360 GB hard disk capacity. The approach is implemented using Net beans 6.9.1 IDE as the frontend and Microsoft access as the backend. The Program is coded using java programming language. The experimental results of the proposed approach are given in following sub-sections for the input dataset. The analysis of the experimental results have confirmed that our proposed approach is efficient and less time consuming in accordance with the experiments conducted.

a. Experimental results

The sample of results obtained from each intermediate step of the proposed approach is given in this section. For providing sample results, we take 3 web page documents from web crawling procedure. The web pages are initially subjected to preprocessing which includes stop word removal and stemming. Preprocessing results in webpage documents represented as a set of keywords. The web page documents and its keywords are given in the table 1.

Table 1. Extracted keywords from the web documents

Web page documents	Keywords
WP_1	[adapt, learn, base, architecture, knowledge, reusable, intelligent, system, present, paper]
WP_2	[power, paper, learn, system, adapt, education, knowledge, crossroad, emerge, field]
WP_3	[learn, track, user, content, real, time, fine, grain, environ, order]

At first, the total number of sentence in each of the web page document is computed and it is stored as sentence level feature. Then, the finger print of each document is obtained using the SimHash algorithm. The features extracted from the taken documents are given in the table 2 and table 3.

Table 2. Sentence level feature of the web page documents

Web Pages	No. of Sentences
Wp_1	6
Wp_2	5
Wp_3	4

Table 3. Fingerprints of each web documents

Webpage	Fingerprint
Wp_1	00011000001011110010100000000111001011100001101000110010000010100
Wp_2	01011110001101010110101000101011010100010000110100100101010000011
Wp_3	01000100010110100100110000111100000111010001010000101001010000011

For detecting near duplicate web page detection, we take W_{pnew} as newly added web page and then, webpage document undergo the cascade filtering process to check whether W_{pnew} is near duplicate web page or not. Cascade filtering technique is applied on the web pages in two levels, namely, sentence level feature extraction and the fingerprint comparison. In this first level of the cascade filtering, we consider only the sentence level features of the webpage documents. Here, comparison with respect to the sentence level

feature is made between the new webpage document and the features stored for other web pages. If the comparison result is valued lower than or equal to the provided threshold T_S , then such web page documents are considered for further processing and the rest of the webpage documents are discarded.

Table 4. Features extracted from the newly added web page document.

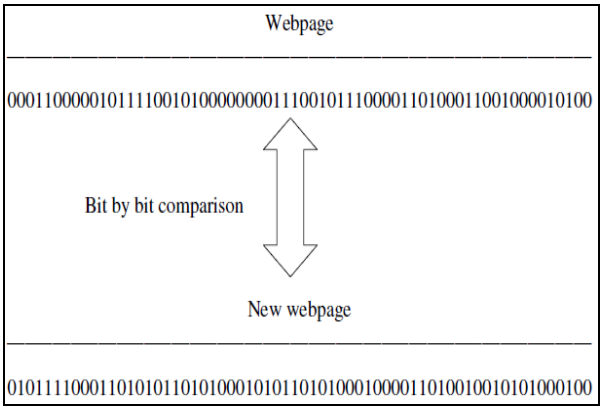
Web Page	Keywords	No. of Sentences
Wp_{new}	Queri,expans,method,summer,problem,feedback,system,unique,monument,show	4

Here, from the table 4, the number of sentences in the new web page document (Wp_{new}) is 4 and threshold provided is 1. By comparing the feature stored in the database with Wp_{new}, W_{p2} and Wp_3 are taken for further processing because these two pages is more similar with the newly added web pages with respect to the sentence length. Then, we compute the fingerprint of the newly added web page document (Wp_{new}) shown in table 5.

Table 5. Fingerprint of the new web document

Webpage	Fingerprint
Wp_{new}	0101111000110101011010100010101101010001000011010010010101000100

The next process is comparing the finger prints of each document that has been selected from the former process to the new web page document's fingerprint. After finding the fingerprints of the newly added web document, comparison of the fingerprints is computed and processed under the basis of a pre-stored threshold value, T_B . A bit by bit comparison process is conducted for the detection of duplicates and near duplicates.



The bit-by-bit difference between the new web page document and the other web pages are found out and total number of bit difference is computed. The bit-by-bit difference value is compared with the predefined threshold T_B . If the bit difference value is less or equal to than the threshold, the web page is considered as a near duplicate webpage. By comparing the fingerprint of the Wp_{new} with theW_{p2} and Wp_3, the bit by bit difference value exceeds the threshold value. Hence, Wp_{new} is not considered as near duplicate web page and it is added to the database with it feature values.

b. Performance evaluation

For evaluating the performance of the proposed approach, we have used dataset that consists of a web page documents obtained through web crawling. It totally contains 100 web pages in which 21 web pages are duplicates. The performance of the proposed approach is evaluated with the help of evaluation metrics such as, Precision, Recall and F-measure. The accuracy is governed by these three factors defined as follows,

$$\Pr ecision(P) = \frac{No.of\ true\ duplicates\ \text{detected}}{Total\ No.of\ duplicates\ \text{detected}}$$

$$\text{Re}call(R) = \frac{No.of\ true\ duplicates\ \text{det}ected}{Total\ No.of\ duplicates\ in\ the\ dataset}$$

$$F - measure(F) = \frac{2 \times P \times R}{P + R}$$

(1) Accuracy: The input dataset is given to the proposed approach to evaluate the accuracy that is calculated for different values of the sentence threshold (T_S) . The computed values are plotted as a graph given in fig 5. From the graph, we can infer that the recall and the f-measure depend on the threshold set. Here, we can see that the precision remains the same even with the variation of the sentence threshold, Ts . In addition to, we can conclude that when we increasing the threshold, the recall and the f-measure are

decreased. Hence, it is required to fix a threshold value such that web duplication detection is possible and also that the recall and the f-measure is high.

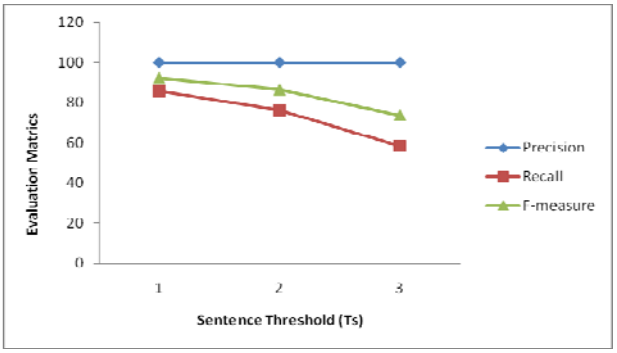


Fig. 5. Performance evaluation of the cascade filtering

(2) Computation Time*:* Time incurred in detection of near duplicates differ for different WebPages and also vary with the number of documents with which the new webpage document is compared. Near duplicates are found out by the comparison between the web documents in the database and the new webpage. So, time incurred usually vary with the number of webpage documents in the database. More number of documents in the database means that the new webpage have to be compared with more number of web documents. In the above case, three new web pages are taken and processed for near duplication with a fixed number of web documents in the database. The time response for each case is plotted. An illustration of performance by our proposed approach is described in fig 6.

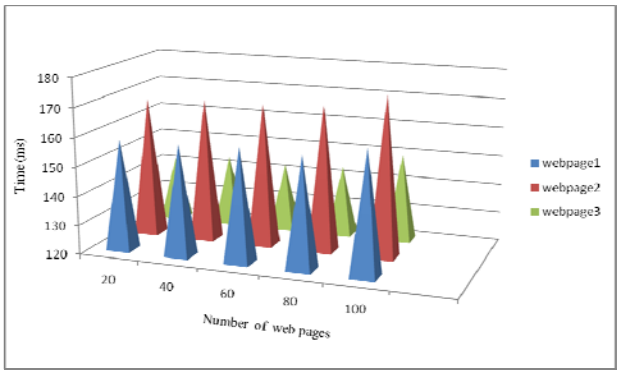


Fig. 6. Computation time of cascade filtering.

c. Performance evaluation of the clustering-based enhanced Approach

The performance of the clustering-based approach is described in this sub-section. When we consider a large amount of dataset, the time consumed will be high to find the near duplicate web pages. So, in order to overcome this problem, we enhance our method making use of the clustering technique. It effectively reduces the total number documents given for the cascade filtering process. The result shows that the clustering results in making the method more efficient than the previous approach. In clustering, Web pages in the database are grouped into K number of clusters.

(1) Accuracy: Accuracy of the proposed approach has improved using the clustering technique. Precision, recall and F-measure are calculated for different values of the sentence threshold and is plotted shown in fig 7. From the analysis of the graph above, we can see that precision remains the same even after the clustering process. Here also, we can see that recall and the f-measure varies with the sentence threshold set. Though the recall and f-measure plots have almost the similar shapes before and after clustering, it is evident from the graph that after clustering, the values of the recall and the f-measure have increased to a greater level. From this, we can conclude that clustering results in increase of recall and f-measure values.

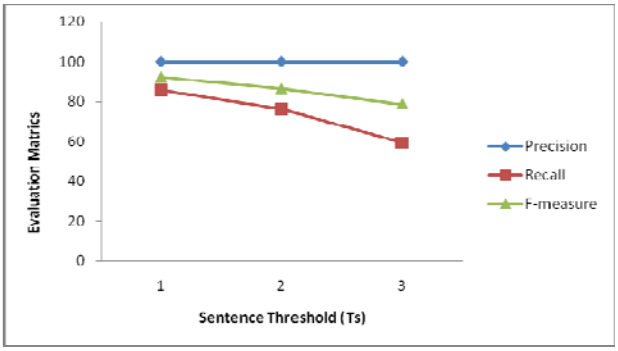


Fig. 7. Accuracy of cascade filtering with clustering.

(2) Computation Time: Time incurred has also significantly reduced using the clustering technique. An illustration of performance by our proposed approach for different cases is described below. After the clustering process, near duplicates are found out by the comparison between the web documents in the respective cluster and the new webpage. Hence, the number of comparisons is less so that the time incurred to find duplicates is also reduced. From the graph, it is

evident that the time incurred has significantly reduced after the use of the clustering technique.

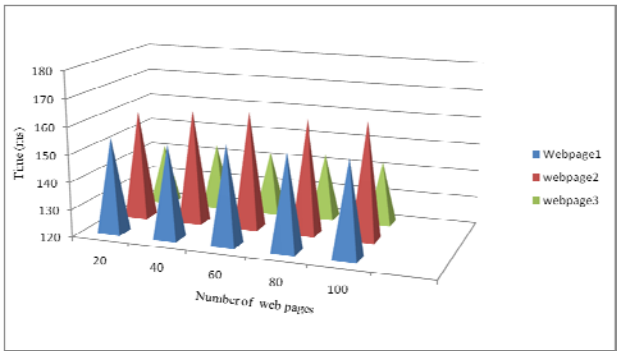


Fig. 8. Computation time of cascade filtering with clustering.

5. Conclusion

Near-duplicate web pages pose a serious threat to the web crawling community and have become the prime concern for the web search engines. Near duplicates increase the cost of serving results, incur large amount of space to store the indexes and ultimately slows down the result, hence affecting both the accuracy and the time for execution. It also results in making the query result less appropriate to the users. There has been a plenty of algorithms designed for the detection of near duplicate detection based on similarity scores and signatures. In this paper, we have proposed an efficient method for detecting the near duplicates using both the sentence level features and the fingerprint technique. These two techniques act as cascade filters and are applied to the preprocessed web page documents. When large number of documents is considered, we have enhanced our approach using clustering for attainment of better results in less computation time. The experimental results have proved that the proposed approach is efficient in both accuracy and time incurred.

References

1. V. A. Narayana., P. Premchand., and A. Govardhan., "Fixing the Threshold for Effective Detection of Near Duplicate Web Documents in Web Crawling", *Advanced Data Mining and Applications*, Vol. 6440, pp : 169-180, 2010.
2. Monika Henzinger, "Finding Near-Duplicate Web Pages: a Large-Scale Evaluation of Algorithms", *In Proceedings of the 29th annu al international ACM SIGIR conference on Research and Development in Information retrie val*, pp : 284-291, 2007.
3. Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig, "Syntactic Clustering of the Web", *In Proceedings of the Si xth International Conference on World Wide Web*, pp : 1157-1166, 1997.
4. Moses S. Charikar, "Similarity Estimation Techniques from Rounding Algorithms", *In Proceedings of th e Thiry-Fourth Annual ACM Symposium on Theory of Computing*, pp: 19-21, 2002.
5. Gurmeet Singh Manku, Arvin Jain, and Anish Das Sarma, "Detecting Near-Duplicates for Web Crawling", *In Proceedings of the 16 th International Conference on World Wide Web*, pp : 141- 150, 2007.
6. A. Kołcz, and A. Chowdhury, "Lexicon randomization for Near-Duplicate Detection with I-Match", *The Journal of Supercomputing*, Vol. 45, No. 3, pp: 255-276, 2008.
7. Ranjna Gupta, Neelam Duhan, A.K. Sharma and Neha Aggarwal, "Query Based Duplicate Data Detection on WWW" , *International Journal on Computer Science and Engineering*, Vol. 2, No. 4, pp: 1395-1400, 2010.
8. Hung-Chi Chang, and Jenq-Haur Wang, "Organizing News Archives by Near-Duplicate Copy Detection in Digital Libraries", *Asian Digital Libraries*, Vol. 4822, pp: 410-419, 2007.
9. V.A. Narayana, P. Premchand, and A. Govardhan, "Effective Detection of Near Duplicate Web Documents in Web Crawling", *International Journal of Computational Intelligence Research*, Volume 5, No. 1 , pp.83–96, 2009.
10. Berendt, B, Hotho, A., Mladenić, D, Spiliopoulou, M, and Stumme, G. "A Roadmap for Web Mining: From Web to Semantic Web", Lecture Notes in Artificial Intelligence, *Springer*, Vol 3209, pp. 1-22, 2004.
11. J Prasanna Kumar, and P Govindarajulu,"Duplicate and Near Duplicate Documents Detection: A Review", *European Journal of Scientific Research*, Vol 32, No. 4, pp.514-527, 2009.
12. Spertus, E., "Parasite: Mining structural information on the web", Computer Networks and ISDN Systems: *The International Journal of Computer and Telecommunication Networking*, vol. 29: pp. 1205 – 1215.,1997.
13. Balabanovic, M., and Shoham, Y. "Learning Information Retrieval Agents: Experiments with Automated Web Browsing", *In On-line Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distrib uted*, Heterogeneous Environments, 1995.
14. Kobayashi, M, and Takeda, K, "Information retrieval on the Web", *ACM Computing Surveys (ACM Press)*, Vol. 32, No. 2, pp. 144–173, 2000.
15. J Prasanna Kumar, and Dr. P Govindarajulu, "Efficient Web Crawling By Detecting and Shunning Near Duplicate Documents", *Georgian Electronic Scientific Journal*, Vol 22, No. 5, pp.109-115, 2009.
16. F. McCown, and M.L. Nelson,"Evaluation of Crawling Policies for a Web-Repository Crawler", *In Proceedings of 17th ACM Co nference on Hypertext and Hypermedia*, pp: 157-168, 2006.
17. Rajashree Shettar, Dr. Shobha G, "Web Crawler on Client Machine", *In Proceedings of the Inter national*

Multi Conference of Engineers and Computer Scientists, Vol 2, pp. 1121-1124 , 2008.

18. Cho, J, Garca-Molina, H. and Page, L. "Efficient Crawling Through URL Ordering", *Computer Networks and ISDN Systems*, Vol. 30, No. 1-7, pp: 161-172.1998.
19. Xiao, C, Wang, W. Lin, X. Xu Yu, J., "Efficient Similarity Joins for Near Duplicate Detection", *Proceeding of the 17th in ternational conference on World Wide Web*, pp: 131-140. , 2008.
20. Conrad, J. G., Guo, X. S, and Schriber, C. P, "Online Duplicate Document Detection: Signature Reliability in a Dynamic Retrieval Environment", *Proceedings of th e Twelfth International Conference on Informat ion and knowledge Management*, New Orleans, LA, USA, pp. 443 - 452, 2003.
21. Fetterly, D, Manasse, M, and Najork, M, "On the Evolution of Clusters of Near-Duplicate Web Pages", *Proceedings of t he First Conference on L atin American Web Congress*, pp. 37.2003.
22. Shoaib. M, Arshad.S, "Design & Implementation of Web Information Gathering System," *NITS e-Proceedings*, Internet Technology and Applications, King Saud University, pp.142-144.
23. Lovins, J.B, "Development of a Stemming Algorithm". *Mechanical Translation and Co mputational Linguistics*, vol. 11, pp. 22-31. 1968.
24. Dick Grune, and Ceriel Jacobs, "Parsing Techniques: A Practical Guide ", pp. 13, 1998.
25. Bayardo, R. J., Ma, Y, and Srikant, R, "Scaling up all Pairs Similarity Search", *In Proceedings of t he 16th International Conference on World Wide Web* , pp.131-140,2007.
26. Cho, J, Shivakumar, N, and Garcia-Molina, H.,"Finding Replicated Web Collections", *ACM SIGMOD Record*, Vol. 29, no. 2,pp. 355 - 366,2000.
27. M.H.C. Law, M.A.T. Figueiredo, A.K. Jain, "Simultaneous Feature Selection and Clustering using Mixture Models", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, pp.1154-1166, 2004.
28. Hannaneh Hajishirzi, Wen-tau Yih, and Aleksander Kolcz, "Adaptive Near-Duplicate Detection Via Similarity Learning" , In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval ,SIGIR '10, pp.416-426, 2010.
29. Bingfeng Pi, Shunkai Fu, Weilei Wang , and Song Han, "SimHash-Based Effective and Efficient Detecting of Near-Duplicate Short Message " *Proceedings of the Second Symposium Internati onal Computer Science and Computational Technology(ISCSCT '09)*, pp. 020-025,Dec. 2009.
30. Anton Karl Ingason, Sigrun Helgadottir, Hrafn Loftsson, and Eirikur Rognvaldsson, "A Mixed Method Lemmatization Algorithm Using a Hierarchy of Linguistic Identities (HOLI)", *Advances in Natural Language Processing-Lecture Notes in Computer Science*, Vol. 5221, pp. 205-216, 2008.
31. Hoad T. C, and Zobel, J, "Methods for identifying versioned and plagiarized documents", *JASIST*, vol. 54, no. 3, pp.203–215, 2003.
32. Spertus, E., Sahami, M, and Buyukkokten, O, "Evaluating Similarity Measures: A Large-Scale Study in the Orkut Social Network", *In KDD (2003)*, pp. 678-684, 2005.
33. Arasu, A, Cho, J, Garcia-Molina, H. Paepcke, A. and Raghavan, S, "Searching the Web", *ACM Transactions on Internet Technology*, vol. 1, no. 1: pp. 2-43, 2001.
34. Gibson, D, Kumar, R, and Tomkins, A, "Discovering Large Dense Subgraphs in Massive Graphs", *In Proceedings of the 31st International Conference on Very Large Data B ases*, Trondheim, Norway, pp. 721 - 732, 2005.
35. Pant. G, Srinivasan, P, Menczer. F, "Crawling the Web". Web Dynamics: Adapting to Change in Content, Size, Topology and Use, edited by M. Levene and A. Poulovassilis, *Springer- verlog*, pp: 153-178, 2004.
36. Gong, C, Huang, Y, Cheng, X. Bai. S, "Detecting Near-Duplicates in Large-Scale Short Text Databases", Lecture Notes in Computer Science, *Springer Berlin / Heidelberg*, Vol. 5012, pp. 877-883, 2008.
37. Ohn Mar San, Van-Nam Huynh, Yoshiteru Nakamori, "An Alternative Extension of the *K*-Means Algorithm for Clustering Categorical Data", *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 2, pp. 241–247, 2004.s
38. Aloysius George, D. Binu, "An approach to products placement in supermarkets using PrefixSpan algorithm", *Journal of King Saud University - Computer and Information Sciences*, 2012.