Detection of Duplicate and Near-Duplicate Content for Web Crawlers

Hadi Hussain Khan[1], Dr. Husnain Mansoor Ali[2]

[1,2]*Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST) Karachi, Pakistan*

me@hadikhan.com
husnain.mansoor@szabist.edu.pk

Abstract - There is an abundance of duplicated web documents on the internet. For example, two documents online could be very similar to each other except for a very small portion, such as URLs and advertisements. While such differences are not important with regards to web searches, they do tamper with web search results due to duplication. Therefore, if web crawlers could check the duplication percentage of newly crawled pages by a previously crawled page, the quality of web search will significantly increase. The main objective of this research is to propose a method which is able to check the duplication ratio of the content on the page with the one already crawled previously. The solution includes running a web crawling algorithm in order to calculate the ratio of duplication at the time of web crawling. In order to effectively achieve the goals of this research, Charikar's SIMHASH finger printing-technique has been used. Using this, a new technique for the purpose of detection of exact and near duplication method will be devised which will work to check the duplication ratio with the newly crawled page. The experiment is carried out on multiple pages of two major B2B website namely Ali Baba and Trade key. More than 300 pages from two similar categories on each portal were selected for this experiment. These selected pages were first calculated using a third party duplication detection tool to set the bench mark. The results obtained from the test looked to be very promising and close to the benchmark set. The system running time was very short. However, the results show an average curve variation of 10% away from the bench mark which in this case is fine. Based on the results obtained from the experiment carried out, it can be said that Charikar's SIMHASH finger printing technique can be effectively used to detect duplication and near duplication.

Keywords - Algorithms, Hamming distance, near-duplicate similarity, search, sketch, fingerprint, web crawler, web document.

I. INTRODUCTION

In today's world of technology, everything is easily accessible on the internet in the form of data. This data is available in the form of "content"; which, with the passage of time, has grown massively. It is almost impossible to find the desired content from such a large amount of data available. This is where search engines play a vital role in finding the desired data from the trillions of megabytes available. Web crawling is useful during such situations.

Web crawlers play a vital role in the helping search engines find results. There are mainly two types of web crawlers, the Generic WebCrawler [1] and Focused WebCrawler [2]. The Generic WebCrawler focuses on all the generic content available on the web. Slightly less powerful in comparison, their function is to capture content generally available on the web. Focused Web crawlers; however, are special web crawlers whose main job is to search for relevant and specific content. These particular crawlers use specially created logic/knowledge to limit the crawling to limited pages, offering specific domains of content. Thus, Focused crawlers mainly look for specific topics already defined and decided.

Documents that are exact clones of other documents are easy to identify and detect. Any conventional hash mechanism like MD5 (Message-Digest) [3] and SHA (Secure Hash Algorithm) [4] can effectively detect another document with 100% duplication. However, when it comes to detecting documents with near-duplication, things get more difficult. Near duplicate content on the web is much more challenging. With regards to content, in any two given pages, everything can be same except for a small portion that differs such as featured advertisements. Even though the content and other section of the page remains the same on every refresh, the advertisement URL keeps changing. Due to this small change, the entire page is categorized by crawlers as different each time. Similarly, counters and time stamp may also make the page different at every crawl.

A. Research Contributions

i. For this particular research, Charikar's SIMHASH [5] was used. Charikar's Simhash mechanism is a very powerful way to detect near duplicate content. Simhash is basically a finger-printing technique, a contrast of other hashing mechanisms available. It keeps track of any changes and

the difference between two hash keys. Using this SIMHASH mechanism, the near duplication percentage can be checked and calculated.

ii. A system will be created which will work to efficiently detect near duplicated content in a short amount of time. This system will also be capable of making quick decisions on the difference between two a finger-print that either the provided content is near duplicate content or not.

II. LITERATURE REVIEW

Multiple techniques have been developed over the years in order to detect duplication among web pages [6, 7, 8]. These attempts are mainly undertaken with different sets of data to make sure that the same approach can be used in different problem sets. Pamulaparthy et al. did a survey in their research paper to find the best duplication detection methods for web crawling [9]. Manku et al. also research a multi-billion-page repository in order to prove that the Charikar's SimHash Method is appropriate for this goal [10]. Meyerzon et al. also researched on the same field referencing a "content identifier (CID)", which refers to a unique storage location in the system [11]. Broder [12] discovered enough during his research to keep each document as a "sketch" of "shingles" in order to calculate the similarity of two documents. Any document pair with at least one common shingle was examined to check whether it exceeded the threshold for resemblance. Broder's shingling method works well in duplicate detection in AltaVista search engine.

Finally, the research undertaken by Manku et al. proved that Charikar's simhash [5] is realistically useful for finding near-duplicates in large-scale web page repository [10]. Simhash is a fingerprint technique having the quality that finger-prints of near-duplicates vary in a minor number of bit positions [5, 10]. If the simhash fingerprints of two calculated documents are similar, they are considered to be near-duplicates.

A. Understanding Simhash

Simhash's entire process and procedure is well defined and elaborated by Charikar in his research paper Charikar [5]. In the ASCII code, 'a' can be represented as 97 and its binary representation is 01100001 as illustrated in figure 1. The entire English alphabets represent in either there ASCII or Binary codes.

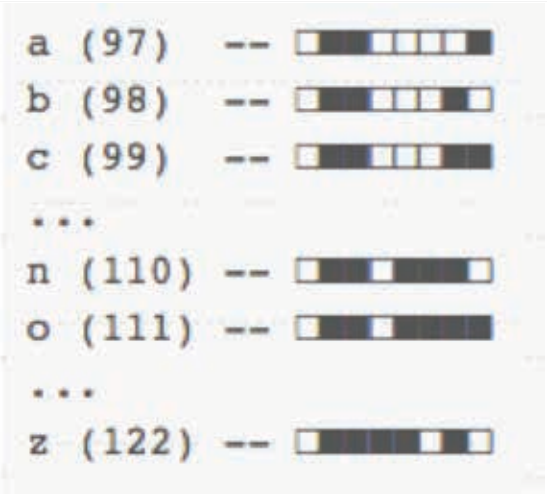

Fig. (1). Conversion table of ASCII to Binary [13]

These conversions are well understood by programmers. This is fundamental conversion and nothing new is introduced. In this way, all English alphabets can also be represented in their respective Binary formation.

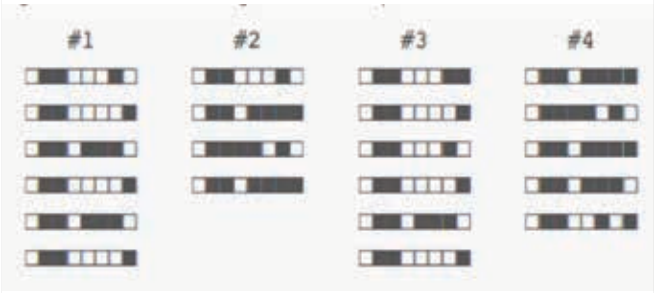

Fig. (2). Binary Representation of English Alphabets [13]

Some programmers are good enough to read directly the alphabets in their ASCII or Binary formats. From Figure 2, it is obvious that there is a repetition on the most important 2 bits from the left side on all letters. These identical bits can simply be removed. Resulting in the following:

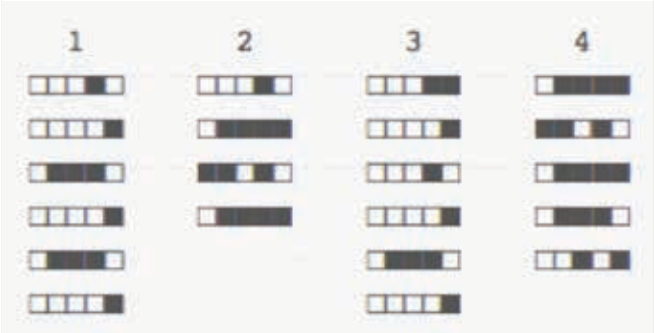

Fig. (3). Binary Representation of English Alphabets [13]

Based on Figure 3, the alphabets can be grouped based on their binary representations. Just by looking at the binary representation of these alphabets, naturally, our brains will start to detect a unique sequence in these representations. Our minds take these discussions based on the position of each bit in the sequence, the total length of the individual sequence and based on the color density (Empty boxes VS. Dark Boxes) some relationship between these sequences can be identified with similarities between these sequences.

Interestingly all these sequences in Figure 3 are completely unique, but based on this orientation, the similarities and differences can be identified. Compared to other hashing methods like MD5, this is respectively a hard job.

72b302bf297a228a75730123efef7c41
e1c8d6347c0c24e5cbc60e508f3fc1b5
da5e04f2b94deec3c70632f77644a041
1b9ac7aecd52eab14089942f8267f22a

From the above mentioned MD5 hash keys, there is no way to predict or create any patterns which should not come as a surprise as the fundamental principle of MD5 is not to get similar hashes.

It is possible to predict matching words and come up with a combination by observing the binary sequences. However, it is not a good idea to implement this in computers; since it will take decades for computers to find out the particular result sets. There is one thing which can be definitely said though: the idea to use and create a decent signature mechanism, which can be used as a hash key method - a simhash.

The idea will be to compress the provided information in a bit. By using this information, a histogram of all the binary representation can be created putting 1 for each value in the column and letting the remaining sections stay blank. This can give us a comprehensive layout (figure 4).

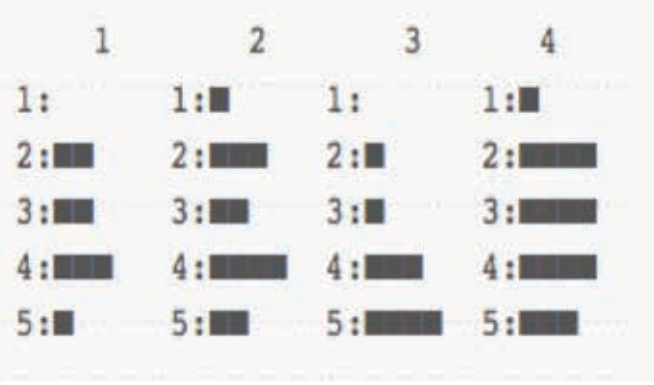

Fig. (4). Horizontal Bar Histogram [13]

While this approach sounds logical, there is a serious problem associated with it. Only half of the data will be focused on. All the 1's are kept and all the 0's are ignored. Instead of just leaving the 0's in this approach, let us mark the empty areas (0's) as -1; whereas the 1's are represented by +1.

Fig. (5). Binary Representation of 0's as -1 and 1's as +1 [13]

A more visually explained picture is attempted by converting Figure 5 in histogram representation of same data set.

Looking at Figure 6, it is much easier to identify which is closest to which. It is also more comprehensible when compared to viewing it in the full pattern method. Despite the fact that only a small amount of data is being observed, a prominent signature which is easily understandable and can be easily distinguished is observed.

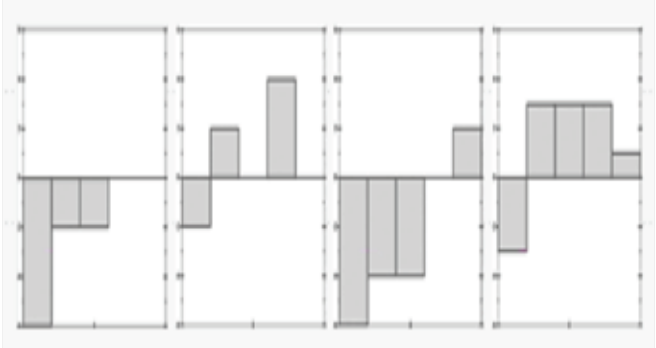

Fig. (6). Histogram representation of Figure 5 [13]

Under consideration of the histogram in figure 6, a general formula is applied to further elaborate this concept. If one replaces 1 for all the values greater then and equal to 0, and for all the values that are less than 0, the bit is being ignored.

Fig. (7). Including all the 1's and excluding 0's [13]

In Figure 7 above, the ideas are becoming clearer still. Keeping in view the figure, 4 mentioned signatures can be grouped into two, i.e. signature 1 and 3 can be grouped together and signature 2 and 4 together.

The question now arises is: how can the separation be made possible between the signatures, when none of them are distinguishable? For example, if there is data available whose resultant key is generated as 10011, what will be the closest value for these type of keys? Using the Hamming Distance technique, it is easier to effectively calculate the key values before comparing them for the duplication detection website. Based on the explanation provided above, the Hamming Distance technique is a procedure that is used to calculate the difference between numbers, letters, or whatever piece of data and find out the ratio of difference between them.

From the combination of groups of bits of almost identical lengths, the procedure requires looking at these groups sequentially. The procedure is pretty simple and straight forward. Every time, while looking at the signatures, if a similar signature is found then 1 is inserted, and if the signatures are not similar, it will remain as it is. After the procedural form of this approach, the signatures will look like as illustrated in figure 8:

Fig. (8). Grouping together similar signatures [13]

This new pattern has a hamming distance of 1 compared to the previously calculated signatures 1 and 3, and vice versa. It has the distance of 3 when compared with the signatures of 2 and 4. From these calculations, it can be rightly said that the new signatures which will be generated by the system will be closer to 1.

A simhash of completely different string value is similar to the simhash of completely different value. For example, if the string is 'sssss' under consideration, its simhash will be closer to 'Banana' than 'bozo'.

III. METHODOLOGY

The research that has been done aims to detect duplicate and near-duplicate content for web crawlers. This section is divided into multiple sub-sections for better understanding, and organization of this research paper.

A. Technologies

In order to successfully undertake all the experiments to support the claims, a custom system has been created on which all the experiments will be done.

B. Language

As a core programming language, PHP has been used. The version of PHP used here is 5.5. The reason of choosing this particular version of PHP was that it was the most stable version at the time when this research paper was written.

C. SIMHASH Library

For the simhash functionality, an open source SIMHASH library under the license of MIT written by Titouan Galopin [14] has been used. The Titouan Galopin approach is a simple approach, which works in the following manner:

- Choose a hash key size. For this experiment, a 64-bit binary hash key has been picked.
- Let the initial variable V is a combination of 64 zeroes.
- Break the provided phrase into small sections. The parameter which has been chosen will ignore all the value 3 and only select split ted words with more than 3 characters.
- Run the algorithm and calculate the hash value for all the chosen values from the system.
- SIMHASH value of biti is only possible if Vi > 0 and 0 elsewhere.

D. Computer Hardware and software used

In order to conduct the experiment, the following Hard ware has been used:

- Processor: 2.5GHz Intel Core i5
- Memory: 8GB 1600 MHz DDR3
- Manufacturer: Apple
- Machine Name: MacBook Pro (13-inch, Mid 2012)
  - OS: Mac OSX El Capitan
  - Storage: 500 GB

E. Database technology

For this experiment, MySQL Version 14.14 Distribution 5.6.26 was used.

F. Initial Decisions

Before elaborating on the steps taken to perform this research, some initial decisions have already been taken to run the system.

- SIMHASH key will not include any internal or external JS or CSS code. Therefore, any code written in script or style tags should be removed before the key calculation.

- Any content between the opening and closing tags of the body will be considered. Content other than the body tags includes mainly supporting files, which are introduced in the system either to revamp the UI of web page, or mainly as supporting files used to support/enhance the functionality of the web page.
- Within the body tags, remove all the content between script and style tags if found. These tags, within content are mainly for UI and have nothing to do with the website content.
- All the internal and external URLs' should be removed prior to taking these steps. However, the anchoring word (if any) should be kept intact to make sure the removal of URL does not affect the meaning of the word. This procedure is also decided prior to the system creation. This step was taken because only text based content has been considered in this research and keys of that content will be generated in order to make sure Simhash keys generate decent results.
- All the image tags will be removed as a whole. This decision was taken because upon analyzing a few websites, it was found that many image names are system generated which does not make any sense for humans. Those systems generated names are mainly created for some sort of automated tracking. However, alt tags mainly do contain some appropriate name conventions for the linked image. Still those names are not a matter of interest in this research as the main source of content will be text based content of the website.

G. Approach

This research requires experimentation to support claims. This entire experimentation was initially planned and discussed before implementation. An overview diagram is shown in the figure 9 which depicts the step by step explanation of the steps involved in this research.

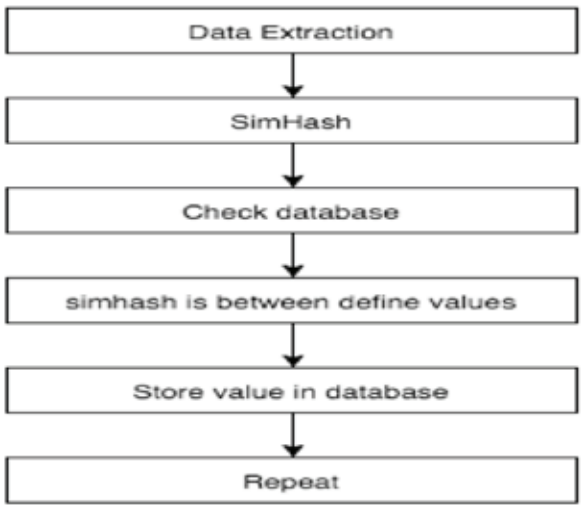

Fig. (9). Including all the 1's and excluding 0's

H. Data Extraction

As a first part of this experiment, a URL is provided into the system. The URL should be accessible and does allow any restriction for accessing it on the server. For experimental purposes, a demo URL has been created with dummy content. That is why any copyright privacy policies were not violated during this experiment.

For this purpose, a very simple interface was given where a URL was given in to the system. The interface is simple HTML based form with a simple submit and reset button. This form was created using traditional HTML and no CSS is involved.

Upon clicking on the button, a simple URL validation runs which checks the validation of the URL and URL box. The actual implementation of this code will involve any sort of crone job which automatically goes for the URL provided by the system. However, because of the experimental basis of this research, a simple HTML form was used to submit the URLs in the system to calculate the values.

Once the submitted URLs are properly validated across multiple check points a new object is created for a custom created class. The Webpage Filter class is divided into 3 functions and a constructor. At the time of object creation, the constructor takes the provided and validated URL as parameters. Once the URL is provided as a parameter, the constructor can then perform certain basic functions as shown under Section 2.

The first line is a PHP method to get the HTML content of the page. Any URL provided in this method will result in making a local/server copy of the HTML code whatever is provided as parameter in the URL. This line of code will download all the HTML code of the provided URL and in this way the content of the page of provided URL can be fetched. This functionality is called web scraping.

I. Check Duplication

Once, all the extra attributes are removed from the newly crawled website as discussed in section 3.2, the simhash values of the provided content are now ready for calculations. For this process, a pre- built program written by T. Galopin [14] has been used. This program was created and developed in PHP keeping the Charikar's SIMHASH algorithm in mind. The simhash calculation function was created by Galopin making it an open source function. However, this program is under the license of MIT and can be used only under said license. The SIMHASH library is available at the github account of Galopin and at the time of this research version 2.0 was available.

The library works in three parts. The first part is:

- It extracts the data from the given content and converts all the given content into small words. This is mentioned in the section 2.2.3. It also uses the same approach to extract the data.
- Once the data is extracted into small sections, then the system converts the data into 64 bit binary hash keys. These small 64-bit keys then merge together to make a single 64-bit binary hash key.
- Later, a similarity check function is available in the system to compare the difference between two binary hash keys. The results the system provides are in the form of floating numbers ranging from 0 to 1. While 1 indicated the 100% same content, 0 indicated 100% unique content. Later, simple percentage formula to convert this floating number into percentage was used. For this purpose, the PHP function was used for percentage.

J. Database Storage

Once the values are calculated and the duplication, it stores in the database to keep a record of the duplication found in the system. For this, 2 separate tables have been used. The first table will be serving as a simhash value storage, whereas the second table will be used as to store the relation of duplication between 2 tables.

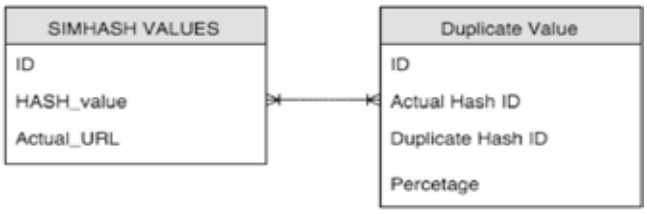

Fig. (10). ERD Representation of storing the values

Figure 10 refers to the database schema designed to store the simhash values during experiments. The database follows the strict rules of RDBMS. Both the tables had the relationship of one-to-many, as one simhash value can have multiple duplicate values. The IDs on both the tables are unique integral value to maintain the uniqueness of the field. Hash_value of Simhash Values table stores the algorithm calculated values where as Actual_URL column stores the URL of scraped web page.

In Duplicate Value table, Actual Hash ID stores the relationship of two tables, similar to the Duplicate Hash ID; storing the ID from which the duplication is detected. Finally, the percentage column stores the duplication percentage calculated by the algorithm.

IV. RESULTS

So far the research shows how Charikar's SIMHASH algorithms works; how the Hardware was practically used in order to complete these experiments; what is the process under which these experiments will go through; the supporting libraries, languages and, finally, the storing mechanism.

As data set, more than 300 pages from two famous B2B website were taken. These pages were selected from 2 similar parent categories of the website (Machinery and Agriculture); with a word count ranging between 300 – 1020 words. At each step, multiple pages were used in experiments, and later they were averaged out to take the duplication percentage. These experiments were firstly taken out using a third party tool for comparison [15] to set the bench mark. However, this algorithm can work on any number of words given in the system.

A. Overall Experiments Results

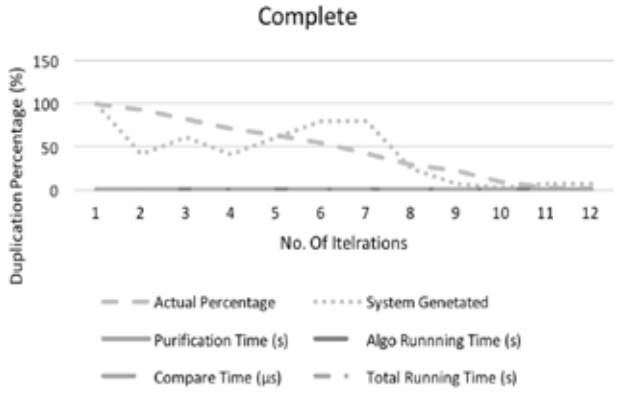

Fig. (11). Complete Experimental result

In Figure 11, an overall picture is shown of the experiment conducted for this research. There are mainly 6 dimensional parameters in this graph: Actual Results (Discussed in sub-section B), System Generated Results (Discussed in sub-section C), Purification time (Discussed in sub-section D), Similarity calculations (Discussed in sub-section E) and Total running time (Discussed in sub-section F). Each point with proper results will be discussed in detail.

In figure 12, actual duplication detection results are shown. These results were obtained by using a third party tool for comparison [15]. The provided data was initially taken as 100% duplicate on both the HTML pages. Gradually, it started decreasing the duplication percentage from the pages. As the graph line showed, there is a continuous decrease in the duplication of the two files.

B. Actual Results

The actual result graph is under:

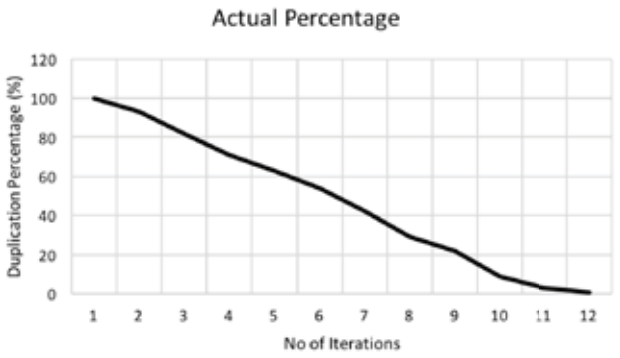

Fig. (12). Actual Results for Duplication detection

This straight line is generated using a third party tool [15] therefore, such findings are not results of this research. The results of this research will be based on the reference line to check the amount of duplication SIMHASH library is generating.

C. System Generated Results

Below Graph in figure 13 Represents the results formulated from experiments.

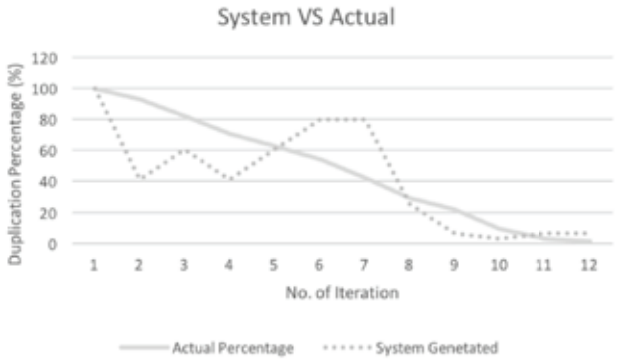

Fig. (13). System VS. Actual Results

The straight downwards line, discussed in section 4.1.1 is detected by the system set up as bench mark. The calculated line is although not a straight line like bench mark, but the line is detecting the duplication at any provided percentage.

The spikes shown in the line are due to the generation of the same key on same words, although the sequence of the words in the two documents is different. There are many words which can be found on both documents. Due to this word repetition in the web documents, the keys generated overlap each other and create spikes.

A plagiarism solution is not proposed in these solutions that's why the exact percentage of the duplication is not a matter of interest. As far as the system is used in detecting duplication, the purpose of this experiment has been achieved.

D. Purification Time

Below mentioned graph in figure 14 represents the graphs on the time spend on purification.

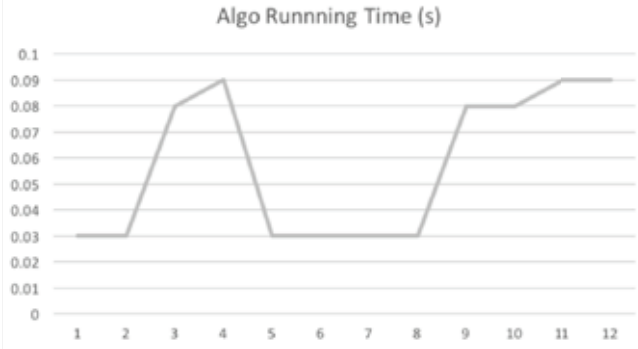

Fig. (14). Purification Time (ms)

The system which was created can logically be divided into three parts.

- Extraction and Purification
- Actual Hash key generation
- Calculating Similarity

The term purification here means the cleansing of the HTML and CSS script from the code. This entire section is already discussed in section 3.2. Once the URL is provided in the system and system completes extraction of all the content of the page, it automatically runs this piece of code which removes all the unwanted sections as discussed in section 3.2 from the content and prepares it to be used as input for simhash key generation.

E. Similarity Calculation

The graph shown in figure 15 below represents the time taken for calculation of similarity percentage(s).

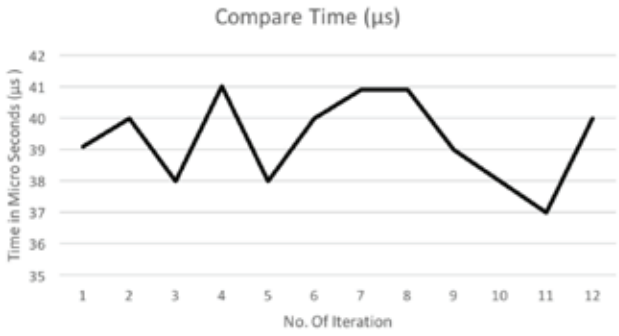

Fig. (15). Time Taken to calculate similarity between two hash keys

The Graph Clearly indicates no comparison time is needed for comparing two keys. Virtually it can be said that no time is needed to calculate the difference between two simhashes values, and it can generate results within no time (as the time calculated is in Micro Seconds).

These results do not include the query time from the database. This research does not cover the area to optimize the database queries for quick retrieval of the results. This straight line indicated some prominent results in this research.

F. Total Running Time

The graph shown in figure 16 below represents the total running time of the system.

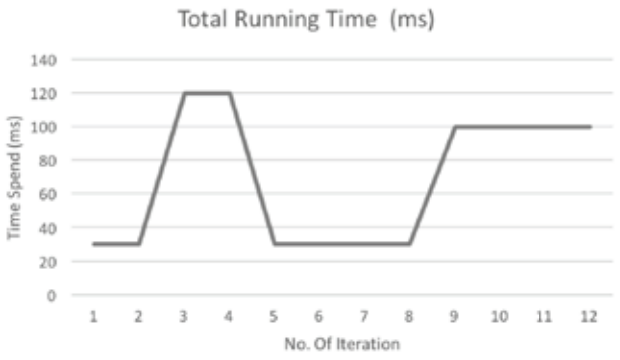

Fig. (16). Total Running time of the algorithm (ms)

This Graph actually gives the most results desired. As can be seen, the results on the graph are static ranging in second. The entire system can calculate the similarity between two hash values within seconds. Overall the system can be used for online – real time results calculations.

The highest peak value system was observed to be 0.12seconds which is still decent as the system has gone through all the necessary steps in order to calculate these values. The variation in the graph can be due to external factors such as traffic on bandwidth at the time of readings, load on server, resources utilities, etc. However, even still the readings are great in terms of time taken to conduct take these readings.

V. FUTURE EXPLORATION

Simhash Algorithm can prove to be the first step in the detection of duplication. If undertaken carefully, there are many other new approaches to improve this algorithm and improve the quality of detection of near-duplicate content detection. It will also make the system more reliable and efficient:

A Can a method be created where web pages can be put into some specialized category, and in other words near duplication can be restricted to the specific categories only?

B One question which remains ambiguous in the research paper published by Manku et al. [10] is already answered in this research. The question was, is there a method where advertisement sections of from the web page can be removed, and only address the content of the page? This entire research is based on the exclusion of the entire external and internal URL as discussed in section 3.2.

C Can simhash be used for the cluster of document? Currently, proposed methods in this research only work for web pages but it can also address on the documents.

D The Birthday Problem should be addressed carefully and removed. If this problem of overlapping hash keys is successfully removed, the same method can then be successfully used for more accurate results. If this problem is addressed, the same system can be used for the plagiarism detection and this way results will improve greatly.

VI. CONCLUSION

Most of the Algorithm currently available in product environment is mainly used in batch-mode over the complete set of documents. This method may be suitable in the case of offline files but for online method, a more efficient method to find the duplication between two web pages is required. If the detection of duplication becomes real time, it will be more feasible.

The system created for this experiment serves best for this research and concluded with promising results. The overall results were close to the bench mark set at the beginning. Based on the observations from the results, it can be proposed that Charikar's SIMHASH finger printing technique can be used in the detection of duplicate and near-duplicate content for web crawlers.

REFERENCES

[1] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. "Searching the web," *ACM Transactions on Internet Technology,* vol. 1, no. 1, pp. 2-43, 2001.

[2] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, "Focused crawling using context graphs" In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00),* 2000, pp. 527-534.

[3] R. Rivest, "The MD5 message-digest algorithm," Net. Work. Grp., MIT Lab. Com. Sci. and RSA Data Sec., Inc., Req. for Comm.: 1321, 1992.

[4] D. Eastlake, and P. Jones, "US secure hash algorithm 1 (SHA1)," Net. Work. Grp., Req. for Comm.: 3174, 2001.

[5] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," In *Proceedings of the 34th annual ACM symposium on Theory of computing (STOC '02),* 2002. pp. 380-388.

[6] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '95),* 1995, pp. 398-409.

[7] N. Shivakumar and H. Garcia-Molina, "SCAM:A copy detection mechanism for digital documents," In *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries,* 1995.

[8] C. Lyon, R. Barrett, and J. Malcolm, "A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector," In *Proceedings of the Plagiarism: Prevention, Practice and Policies Conference,* 2004.

[9] L. Pamulaparthy, M. S. Rao, and C. V. G. Rao, "A Survey on Near Duplicate Web Pages for Web Crawling," *International Journal of Engineering Research & Technology,* vol. 2, no. 9, pp. 1250-1255, 2013.

[10] G. S. Manku, A. Jain, and A. D. Sarma, "Detecting near-duplicates for web crawling," In *Proceedings of the 16th nternational Conference on World Wide Web*, 2007.

[11] D. Meyerzon, S. Shoroff, F. S. Terek and S. Norin, "Method and system for detecting duplicate documents in web crawls." U.S. Patent No. 6,547,829. 15 Apr. 2003.

[12] A. Z. Broder, "*Identifying and Filtering Near-Duplicate Documents," In Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching,* 2000, pp. 1-10.

[13] J. Jones. (2012). *Simhashing (hopefully) made simple* [Online]. Available FTP: http://ferd.ca/simhashing-hopefully-made-simple.html

[14] T. Galopin. (2015). *tgalopin/SimHashPhp* [Online]. Available FTP:https://github.com/tgalopin/SimHashPhp.

[15] *Text Comparison Search - Compare Two Documents For Duplicate Content*. (2015). [Online]. Available: http://www.duplichecker.com/comparison.