



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
CENTRAL CAMPUS

A PROJECT REPORT ON DOCUMENTS SIMILARITY DETECTION BASED
ON BLOOM FILTER

By: Sujit Maharjan 071/MSCS/668

A PROJECT REPORT SUBMITTED TO DEPARTMENT OF ELECTRONICS
AND COMPUTER ENGINEERING, LALITPUR, NEPAL

MAY 2016



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
CENTRAL CAMPUS

A PROJECT REPORT ON DOCUMENTS SIMILARITY DETECTION BASED
ON BLOOM FILTER

By: Sujit Maharjan 071/MSCS/668

A PROJECT REPORT SUBMITTED TO DEPARTMENT OF ELECTRONICS
AND COMPUTER ENGINEERING, LALITPUR, NEPAL

MAY 2016

ACKNOWLEDGEMENT

I owe my sincere gratitude to our Head of Department **Dr. Deewakar Raj Pant, Prof. Dr. Sashidhar Ram Joshi, Prof. Dr. Subarna Shakya, Dr. Aman Shakya** and **Dr. Sanjeeb Prasad Panday** for their kind effort and help in guiding continuously during the project.

Dr. Panday, who is also program coordinator of MSCS, deserves an special vote of thanks. He helped us a great deal since selection of title to completion of this project.

My final thanks also goes to all my friends, seniors, teachers and all those people who have helped me directly or indirectly in bringing this project up to this stage .

ABSTRACT

Comparison of documents has always been field of research for many years. Different ways of comparing documents and matching similarity has been published. In the similar way, comparing documents using bloom filters of fingerprints of the documents make comparison of documents possible in constant time. The documents are first divided into chunks based using W-Shingling. The W-shingling is a set of unique "shingles" that can be used to gauge the similarity of two documents. The sequence of tokens obtained from the shingling is used as fingerprint to form bloom filter. Bloom Filter is a probabilistic data structure that will reduce $O(n)$ search time to constant time. It is represented by an array of bits and the intersection of the bits between bloom filters of documents tells similarity of the documents. So, if we used jaccard method of document coparision, the traditional method of searching the strings into a document then to would take long time based on the size of the document however the use of bloom filter reduces the searching time to some constant time.

Key Words:

Document Similarity, Bloom Filter, w-Shingling

TABLE OF CONTENTS

ACKNOWLEDGEMENT	I
ABSTRACT.....	II
LIST OF FIGURES	IV
1. INTRODUCTION	1
2. PROBLEM DEFINITION	1
3. OBJECTIVES	2
4. METHODOLOGY.....	2
a. LITERATURE REVIEW	2
b. ALGORITHMS	2
Rabin Fingerprinting	2
W-Shingling	2
Bloom Filter	3
Jaccard Similarity.....	3
c. TOOLS USED	3
Programming Language	3
Framework	3
Project Management Tools	3
5. RESULTS AND DISCUSSIONS	4
6. CONCLUSIONS.....	5
7. LIMITATIONS.....	5
8. FUTURE ENHANCEMENTS	6
9. REFERENCES AND BIBLIOGRAPHY	7

LIST OF FIGURES

Figure 1 Jaccard Similarity with and without stopwords.....4

Figure 2 Time complexities in existing and proposed algorithms5

1. INTRODUCTION

Internet is a gigantic source of resources. Everyday new documents are being added into it in very huge size. Enterprise and web search has become ubiquitous part of the web experience. Numerous studies have shown that the ad-hoc distribution of information on the web has resulted in a high degree of content aliasing [1]. We are facing an ever increasing volume of text documents. The abundant texts flowing over the Internet, huge collections of documents in digital libraries and repositories, and digitized personal information such as blog articles and emails are piling up quickly everyday [2]. Internetlivestat has already recorded 1 billion websites over the internet in September 2014. These have brought challenges for the effective and efficient organization of text documents.

In the same way, it has increased the challenge to find right document within a desired time. Also, the interest of people has grown to find originality of the contents. The originality of the contents can be checked using very simple linear search algorithm. The linear search algorithms work very fine when there are few documents but it doesn't work in a real time application when the dimension of the data goes very higher. For the larger dimension of data, the time and space complexity goes exponentially higher so, a space and time efficient algorithm is required.

For finding similar documents, we compare the Bloom filter of one with that of the other. In case the two documents share a large number of 1's (bit-wise AND) they are marked as similar. In this case, the bit-wise AND can also be perceived as the dot product of the two bit vectors. If the set bits in the Bloom filter of a document are a complete sub-set of that of another filter then it is highly probable that the document is included in the other.

Bloom filters, when applied to similarity detection, have several advantages. First, the compactness of Bloom filters is very attractive for storage and transmission whenever we want to minimize the meta-data overheads. Second, Bloom filters enable fast comparison as matching is a bit-wise-AND operation. Third, since Bloom filters are a complete representation of a set rather than a deterministic sample, they can determine inclusions effectively.

2. PROBLEM DEFINITION

Similarity detection is one of the basic problems in the field of computer science like machine learning, data mining, and plagiarism detection and recommender system [3]. The linear search can be applied to random data while it is possible to apply partition search to already sorted data but in both of case it takes a lot of time and makes it unsuitable for unstructured and high dimensional data. The existing similarity search algorithms like jaccard cannot perform well when the size of the document increases. So, it is necessary to build a data structure which is computationally cheaper and time efficient to search strings in a large size of document.

3. OBJECTIVES

The main objectives of this project are below:

- a. To build application to detect similarity between two documents
- b. To effectively use bloom filter as similarity measure

4. METHODOLOGY

A. LITERATURE REVIEW

The similarity detection can be easily achieve with the nearest neighbour search algorithm for less-dimensional data, but both the time and space complexity increases exponentially as the size or number of documents increases. There are also various other methods which have reduced the complexity in linear similarity search.

The shingling algorithm by Broder et al.'s [4] and random projection based approach by Charikar's hare studied as for determining the near-duplicate web pages. Malcolm Slaney described the various optimization techniques in web scale research for nearest-neighbor retrieval. Most of the algorithms use LSH as fundamental concept for probability to find nearest neighbor element of query.

The Bloom filter has received the attention of the research community in the recent past. Bloom filters are nowadays used in varied application and in different forms. And it has also been successfully implemented in comparing document similarity.

B. ALGORITHMS

Rabin Fingerprinting

Given an n-bit message m_0, \dots, m_{n-1} , we view it as a polynomial of degree n-1 over the finite field.

$$f(x) = m_0 + m_1x + \dots + m_{n-1}x^{n-1}$$

We then pick a random irreducible polynomial $p(x)$ of degree k over finite field and we define the fingerprint of the message m to be the remainder $r(x)$ after division of $f(x)$ by $p(x)$ over finite field.

W-Shingling

In natural language procesing a w-shingling is a set of unique "shingles" that can be used to gauge the similarity of two documents. The w denotes the number of tokens in each shingle in the set.

The document, "a rose is a rose is a rose" can be tokenized as follows:

$\{(a,rose,is,a),(rose,is,a,rose),(is,a,rose,is),(is,a,rose,is),(a,rose,is,a),(rose,is,a,rose)\} =$
 $\{(a,rose,is,a),(rose,is,a,rose),(is,a,rose,is)\}$

Bloom Filter

A Bloom filter of a set U is implemented as an array of m bits. Each u ($u \in U$) of the set is hashed using k independent hash functions h_1, \dots, h_k . Each hash function $h_i(u)$ for $1 \leq i \leq k$ maps to one bit in the array $\{1 \dots m\}$ [1]. Thus, when an element is added to the set, it sets k bits, each bit corresponding to a hash function, in the Bloom filter array to 1. If a bit was already set it stays 1. For set membership checks, Bloom filters may yield a false positive, where it may appear that an element u is in U even though it is not. From the analysis, given $n = |U|$ and the Bloom filter size m , the optimal value of k that minimizes the false positive probability [5], p^k , where p denotes that probability that a given bit is set in the Bloom filter, is $k = \frac{m}{n} \ln 2$. Previously, Bloom filters have primarily been used for finding set-membership.

Jaccard Similarity

Consider two sets $A = \{0,1,2,5,6\}$ and $B = \{0,2,3,5,6,9\}$.

The Jaccard Similarity is defined

$$JS(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

More, given a set A , the cardinality of A denoted $|A|$ counts how many elements are in A . The intersection between two sets A and B is denoted $A \cap B$ and reveals all items which are in both sets. The union between two sets A and B is denoted $A \cup B$ and reveals all the items which are in either set.

C. TOOLS USED

Programming Language

Python is used as the server side scripting language.

Framework

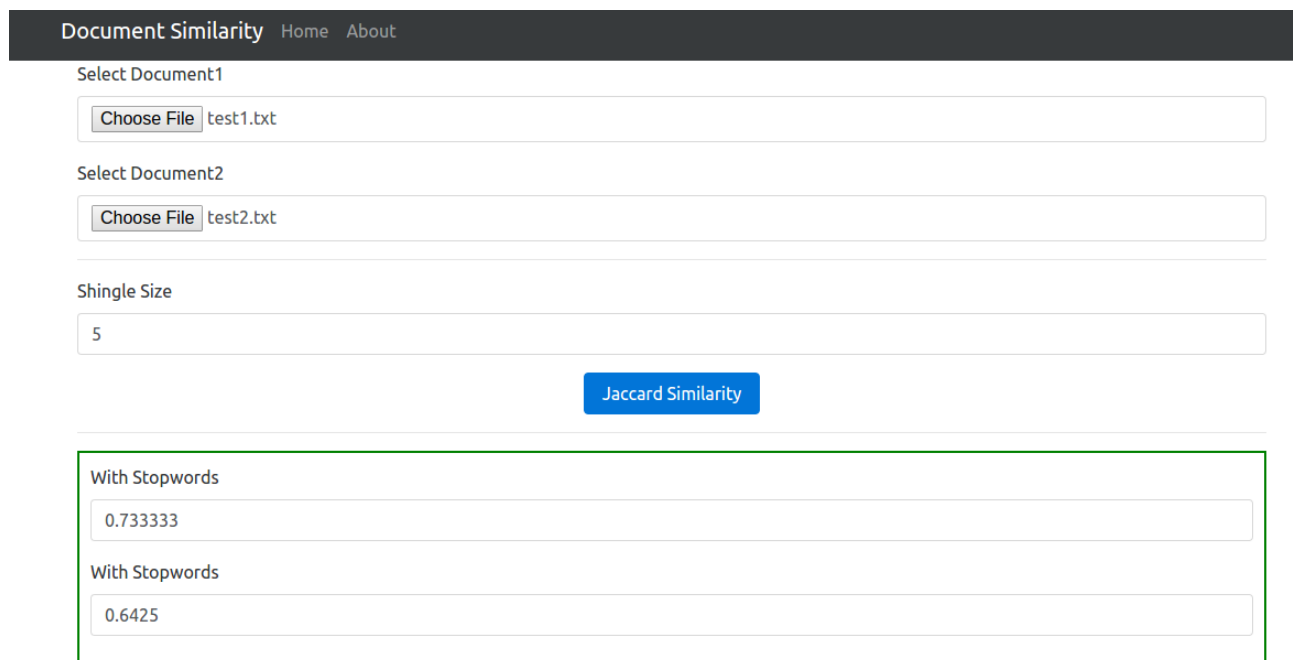
Python Django framework was used for coding web application.

Project Management Tools

Github was used for as code repository and Trello was used for task management and issue tracking.

5. RESULTS AND DISCUSSIONS

The similarity of two documents is compared by using Jaccard similarity and the similarity is calculated by removing the stop words. Stop words are removed because they do not provide meaningful result. The result of similarity lies between 0 and 1. It tells how much two documents are similar. If the similarity of two documents is 1 it means both the documents are similar. If the similarity of two documents is 1 it means both the documents are similar, on the other hand if similarity is 0, it means that the documents are entirely different. Jaccard Similarity of two text files, with and without removing stop words is calculated as shown figures below.



Document Similarity [Home](#) [About](#)

Select Document1

test1.txt

Select Document2

test2.txt

Shingle Size

With Stopwords

Without Stopwords

Figure 1 Jaccard Similarity with and without stopwords

The Jaccard algorithm has been run for four text files which contain texts from wikipedia on same content in different text files. Shingles sizes for those files were taken as 5, 6, 7, 8, 9 and 10. For the graph below we can clearly see that the comparison using bloom filter lookup has constant search time complexity.

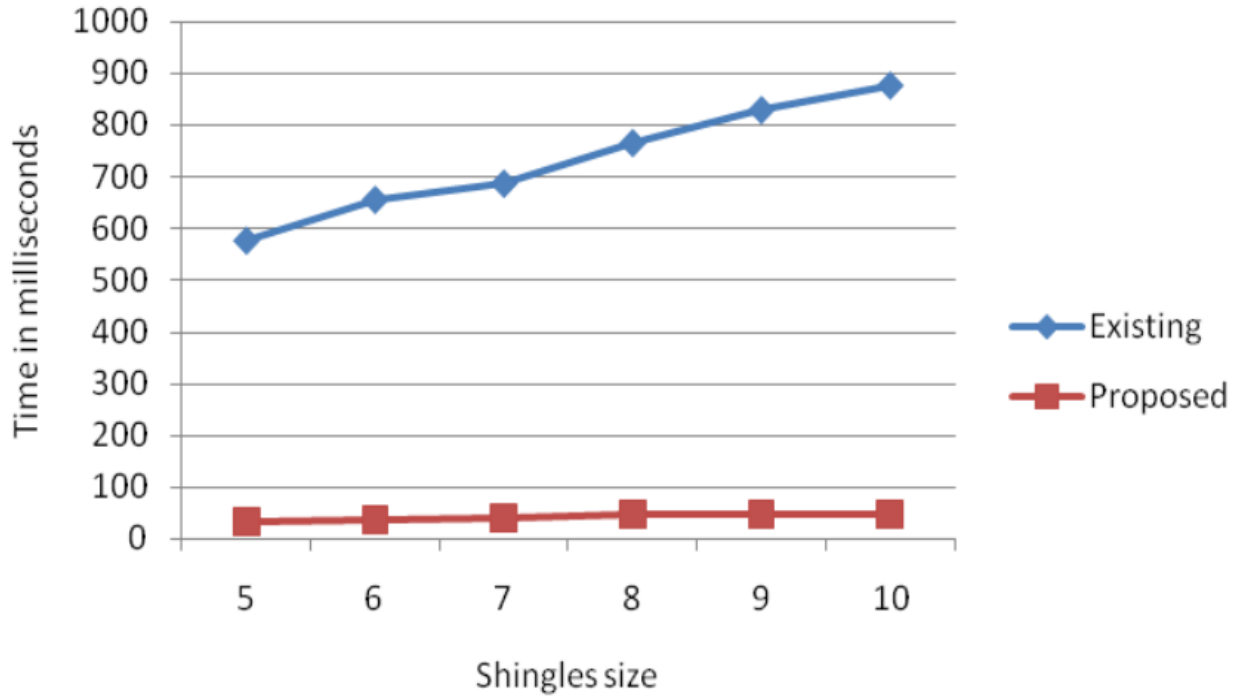


Figure 2 Time complexities in existing and proposed algorithms

6. CONCLUSIONS

In this work, similarity search problem has been solved by using Bloom Filter data structure. While comparing the documents, the shingles is computed and they are stored in the bloom filter so that searching of a shingle is linearly dependent on the file size i.e $O(n)$, which is significantly large time when processing large files. However, the proposed algorithm with Bloom Filter which is used for storing shingle results in constant search time and also compared the time complexity for both of the methods.

7. LIMITATIONS

There were various choices for generating fingerprinting for dynamic shingling. Among them rolling hash which is based on the Rabin fingerprinting was used for generating the fingerprints. However, the result form the fingerprinting using rolling hash was not satisfactory for smaller files.

The fixed sized shingling needs large number of comparisons and lookups into to bloom filters. The size of the bloom filter is kept constant at a very big size to hold very large size of documents. This ultimately degrades the performance for the small sized files.

8. FUTURE ENHANCEMENTS

In this project, a simple Bloom Filter is used to store the shinglings. Both the accuracy and speed depends mainly upon those hash functions. In future, this work can be made more effective in terms of speed and accuracy by using some fast data specific hash functions. The problem of false positive due to bloom filter and the problem of false negative can also be minimized using hash functions. The bloom filter size and number of hashing function can be selected by using some statistical and probabilistic methods.

Apart from enhancing the bloom filter, this similarity matching can be made more efficient in two ways:

- Firstly, a parallel processing environment can be build such that shingling of documents can be distributed in parallel and multiple files can be selected for similarity search.
- Secondly, this work can be extended to for finding semantic similarity in text documents i.e to find the similarity between text files which are semantically similar.

9. REFERENCES AND BIBLIOGRAPHY

- [1] Navendu Jain, Mike Dahlin, and Renu Tewari, "Using Bloom Filters to Refine Web Search," in *Eighth International Workshop on the Web and Databases*, Baltimore, 2005.
- [2] Anna Huang, "Similarity Measures for Text Document Clustering".
- [3] Sachindra Singh Chauhan, "Similarity Search using Locality Sensitive Hashing and Bloom Filter," THAPAR UNIVERSITY, PATIALA, Masters Thesis 2014.
- [4] A. Broder et al., "Min-wise independent permutations," *Proc. Theory of computing*, 1998.
- [5] Burton H. Bloom, "Space / Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, July 1970.
- [6] Chow Kok Kent and Naomie Salim, "Features Based Text Similarity Detection," *JOURNAL OF COMPUTING*, vol. 2, no. 1, January 2010.