Fall 2015

Scalable Techniques for Similarity Search

Siddhartha Reddy Nagireddy
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Nagireddy, Siddhartha Reddy, "Scalable Techniques for Similarity Search" (2015). *Master's Projects*. 438.
DOI: <https://doi.org/10.31979/etd.w9ed-wqnd>
https://scholarworks.sjsu.edu/etd_projects/438

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Scalable Techniques for Similarity Search

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfilment

Of the Requirements for the Degree

Master of Computer Science

By

Siddartha Reddy Nagireddy

December 2015

©

Siddartha Reddy Nagireddy

ALL RIGHTS RESERVED

SAN JOSE STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

SCALABLE TECHNIQUES FOR SIMILARITY SEARCH

By

Siddartha Reddy Nagireddy

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

Dec 2015

Dr. Tsau Young Lin	Department of Computer Science

Dr. Chris Tseng	Department of Computer Science

Mr. Karthik Rajagopalan	Lead Data Engineer, Target

ABSTRACT

SCALABLE TECHNIQUES FOR SIMILARITY SEARCH

By Siddartha Reddy N

Document similarity is similar to the nearest neighbour problem and has applications in various domains. In order to determine the similarity / dissimilarity of the documents first they need to be converted into sets containing shingles. Each document is converted into k-shingles, k being the length of each shingle. The similarity is calculated using Jaccard distance between sets and output into a characteristic matrix, the complexity to parse this matrix is significantly high especially when the sets are large. In this project we explore various approaches such as Min hashing, LSH & Bloom Filter to decrease the matrix size and to improve the time complexity. Min hashing creates a signature matrix which is significantly smaller compared to a characteristic matrix. In this project we will look into Min-Hashing implementation, pros and cons. Also we will explore Locality Sensitive Hashing, Bloom Filters and their advantages.

Acknowledgements

First and foremost I would thank Dr. Tsau Young Lin, who has helped me in preparation of this project report. He has guided me throughout the project with patience and has been my source of inspiration.

Also I would like to thank Dr. Chris Tseng & Mr. Karthik R for their invaluable advice and for accepting to be my committee members.

Siddartha Reddy N

Table of Contents

1. Introduction.....	8
2. Problem Formulation.....	13
3. Minhashing & LSH.....	16
3.1Nearest Neighbour search.....	16
3.2Minhashing.....	17
3.3Locality Sensitive Hashing.....	18
3.4Bloom Filter.....	20
4. Design Techniques.....	24
4.1Sets.....	24
4.2Distance Metrics.....	25
4.3Shingling.....	26
4.4Characteristic Matrix.....	27
4.5Minhash Signatures.....	29
4.6LSH on Signatures	33
4.7Banding Technique.....	34
4.8Analysis.....	35
5. Implementation and Analysis.....	37
6. Conclusion.....	41
7. Future Scope.....	42
8. References.....	43

List of Figures

Fig 1.1 Jaccard Similarity of two sets.

Fig 3.1 General Hashing vs LSH

Fig 3.2 (d_1 , d_2 , p_1 , p_2) – sensitive function

Fig 3.3 Working of a Bloom Filter

Fig 4.1 Banding Technique

Fig 4.2 the S – Curve

Fig 5.1 Shingles of documents

Fig 5.2 Estimated JS vs Actual JS

Fig 5.3 Estimated JS vs Actual JS for another run

Fig 5.4 Time Taken JSIM vs Min-Hash

Fig 5.5 Processed DNA input data

List of Tables

Table 4.1 Matrix representation of sets

Table 4.2 Permutation of the table 4.1

Table 4.3 Hash Functions computer for Table 4.1

Table 4.4 Signature matrix of table 4.1

Table 4.5 Signature matrix after scan of first row

Table 4.6 Signature matrix after scan of second row

Table 4.7 Signature matrix after scan of third row

Table 4.8 Signature matrix after scan of fourth row

Table 4.9 Signature matrix after scan of final row

Table 5.1 Time taken for JS vs MinHash

Table 5.2 Similarity est. by no. of hash functions

Table 5.3 Alergia results t0980 vs t1190

Table 5.4 Alergia results t0980 vs sorted t1190

Table 5.5 Min-Hash results on DNA data

Chapter 1

Introduction

The vast variety of heterogeneous data being dumped onto the web recently has seen overloading of internet data repositories, libraries and warehouses. This has led to a great difficulty in extracting useful information in a quick time considering the volume of data that is being uploaded each day. String search is one that is widely used for searching, wherein performance forms the most important aspect in retrieval of information. Adding to the complexity are the various variety of data such as video, text documents, graphical images, pages on the web etc.

With increasing data, similarity search has become the most computationally expensive task in a range of application areas such as pattern matching, data mining, multimedia information retrieval, bio medical datasets, machine learning, data compression, computer vision and statistical data analysis. These areas require near matches and exact matches rarely has any meaning. Here proximity and distance concepts are much more useful and beneficial for searching. For example, let us say we are given a set of documents which describe the professions about each individual. If we wish to cluster people with same professions, exact match would be of no use. We would need some kind of approach to determine the distances between these documents and cluster only those which fall near to each other.

Consider the web, there are huge number of web pages which are mostly the same but differ from each other by only a small portion of text. Identifying such web pages is very importance, it can have an effect on search results, cleaning, identifying the source of information etc. Identifying exact documents is quite easy as we can hash the documents and compare, but even a small change in documents would output a different hash code. In such cases we need to follow approaches on near duplicate matches. Academia, industry and various organizations have shown a keen interest in identifying such near duplicates for use in numerous applications such as multimedia applications, web scale search & detection and elimination of near-duplicates.

In order to approach such a problem, the data must be first transformed into vectors. Several applications / similarity algorithms first represent the data as high dimensional vectors. The similarity between the documents can be assessed by determining the proximity between the features extracted from the documents. For example, bag of words model can be applied to textual documents wherein every word is its feature and dimension which are essentially vectors. Also an image can be broken down into vectors wherein the density of the colour associated with dimensions is the value contained in each dimension [1]. Once the data is represented as vectors, the similarity between the data can be deduced by calculating the proximity between two vectors using distance metrics such as Jaccard distance, Hamming distance, Euclidean distance, Cosine similarity, Edit Distance etc.

Shingling [4] refers to the detection of near-duplicate items. Given a shingle of size of length k , it is nothing but a sequence of terms in a document D , the k -shingles are a set of all the successive sequences of k -terms in D . The set of these shingles contain no duplicates and take up a space complexity of $O(kn)$ wherein k is the length of shingle and n is the length of the document. Consider a shingle length of $k = 3$ then the document “Locality Sensitive Hashing reduces time complexity” can be represented as 3-shingles set {“Locality Sensitive Hashing”, “Sensitive Hashing reduces”, “Hashing reduces time”, “reduces time complexity”}.

Since now we understand how to deduce vectors from data, a distance measure would be a function $\text{dist}(v1, v2)$ in vector space having points $v1, v2$ as input and outputs the distance between those points which satisfies the conditions below [6]:

- $\text{dist}(v1, v2) \geq 0$
- $\text{dist}(v1, v2) = 0$ if and only if $v1 = v2$
- $\text{dist}(v1, v2) = \text{dist}(v2, v1)$

There are various metrics which are used to calculate the proximity between the vectors in some given space. Each metric has its own advantages and disadvantages, here we would be looking at Jaccard Similarity.

The Jaccard Distance $J\text{-DIST}(S1, S2)$ can be defined as inverse to the Jaccard Similarity of both the sets $S1$ & $S2$ i.e. $J\text{D}(S1, S2) = (1 - J\text{-SIM}(S1, S2))$. Jaccard

Distance relates to set similarity and union, the Jaccard similarity is calculated between two sets as a ratio of the intersection and union of the sets.

- The cardinality of intersection of two sets S_1 and S_2 is less than or equal to the cardinality of the union of both the sets, thereby the J-DIST can never be negative for two points. [4]
- Since the intersection of two sets is always less than or equal to the union of the two sets, the JSIM for any two sets is always less than or equal to 1. The JSIM equals to 1 only when both the sets are identical. [4]
- JSIM and J-DIST both satisfy the triangular inequality property. [4]

Since we now have a value which determines the proximity of the vectors thereby determining if documents are similar, we can consider this to be a problem of similarity search wherein all the pairs need to be compared [5]. This can be defined as the problem to cluster or group all the pairs of items which have value obtained from distance metric above a predetermined value. Applications such as search engines, recommendation engines would require to group similar items as these applications have large volume of entries of in the matrix, which are usually sparse.

Example: In fig 1.1, there are two sets S and T . The intersection i.e. items which appear to be common in both sets of these sets is 3 and the union is 8 i.e. total number of items in both the sets.

Fig 1.1 JSIM [3] of two sets. [2]

Here the JSIM [3] of these sets would be its intersection by union which is $3/8$.

Chapter 2

Problem Formulation

Similarity search has been a fundamental problem since many years in data mining and retrieval which has its applications in machine learning, data mining, and recommendation engines. Current similarity search techniques such as exact search and near dup search algorithms take a lot of time which makes them unsuitable for high dimensional and unstructured data.

Challenges

- Clustering / near dup detection on high dimensional data is a very expensive task. The process can take many days to complete.
- The process is very memory intensive as many algorithms require the entire dataset to fit in the memory.
- Due to above reasons algorithms cannot be directly applied to the entire dataset and there needs to be some kind of blocking / partitioning technique.

Objectives

- To study and analyse the problem of similarity search and near dup detection.
- To analyse the process of vector space modelling, distance metrics and clustering techniques

- To explore the similarity search techniques for high dimensional data such as LSH and to explore the techniques for dimensionality reduction.
- To implement, test and validate the techniques for high dimensional similarity search.

Methodology

Aim of this project is to study, analyse and implement various techniques for high dimensional similarity search. Understand the process in identifying sets of similar pairs, process flow and if there is any possibility of tweaking the algorithms.

- Since we are talking about high dimensional data, Minhashing is used for dimensionality reduction wherein dimensions are reduced significantly while holding the similarity essence of the documents.
- LSH is used for finding candidate pairs so that only subsets of items are compared.
- Bloom Filter helps in improving the space complexity and in providing constant access / search time for membership test.
- The algorithms are implemented and with shingles of varying sizes.

Gap Analysis

Space Complexity: Since we are dealing with very high dimensional data in case of similarity search, we need to decrease the space complexity.

Time Complexity: The current algorithms such as k-means clustering, dbscan, etc. perform well for smaller datasets. For high dimensional data the complexity of these algorithms increases exponentially as the number of dimensions increase.

Accuracy: Using probabilistic methods of similarity search can lead to erroneous results, it should be made sure that the error rate is determined and reduced.

Chapter 3

Minhashing & LSH

Nearest Neighbour Problem

NNS (Nearest Neighbour Problem) is an optimization problem to find nearest neighbours in a given vector space. [7]

Consider a set P of n points which are present in a d -dimensional space $M = (X, d)$ and given a point $q \in X$, find out a point which is closest to point q , let's call this point p [8]

NNS is an optimization problem which is to locate the items which are most similar in a dataset. The similarity or proximity of items is usually expressed as a function which returns a real integer which determines the closeness of the compared items. This problem can also be related to post-office problem which is also an application of nearest neighbour search and direct generalization of this problem is k -NN search, where the k represents the number of closest points required in a given metric space. Generally, M is the metric space and the dissimilarity is usually expressed using the above discussed distance metric techniques such as Edit Distance, Jaccard Distance, Hamming Distance, Cosine Distance etc, which is symmetric and also satisfies the triangle inequality [7]. There can also be an arbitrary dissimilarity function.

There is also a less accurate variant of this problem known as the approximate nearest neighbour search which focuses on speed rather than accuracy. This is beneficial in case of applications which can afford to lose some accuracy in order to gain speed. This freedom of accuracy was introduced by Ciaccia and Patella who

introduced an accuracy parameter ' ϵ ' and a confidence parameter ' α ' which defines the following definition for approximate nearest neighbour search [7].

Consider a set P of n points, which are present in a d -dimensional space X , given an accuracy parameter ϵ and a confidence parameter $\alpha \in [0, 1)$. Design an algorithm that, given a point $q \in X$ output's a closest point p which has a distance of at most $1 + \epsilon$ times the distance of q to its nearest neighbour in P with probability at least $1 - \alpha$. For $\alpha = 0$ it is called $(1+\epsilon, \alpha) - \text{NN}$, for $\alpha > 0$ $(1+\epsilon, \alpha) - \text{NN}$ [8].

Minhashing:

Minhashing is a type of LSH technique which is primarily used for dimensionality reduction. The similarity of two sets are determined by independent permutations of a given column. This technique was first used in the early search engines such as Alta Vista and different variants of this technique are being used in many popular search such as google, yahoo, bing etc [7]. It was first used to detect and remove duplicate web pages, it is also applicable for clustering, in which documents can be grouped based on similarity of their feature sets.

Locality Sensitive Hashing

LSH is not similar to the general hashing techniques which are used to determine the exact matches between data. [3]

In case of locality sensitive hashing similar items tend to be hashed to same bucket. Below is a simple visualization of how the locality sensitive hashing works.

Fig 3.1 General Hashing vs LSH [9]

The pairs which hash to the same bucket are known as candidate pairs, these are obtained when banding technique is applied. Once we obtain the candidate pairs we can simply compare these pairs thereby drastically reducing the number of comparisons required. LSH can be applied to various domains depending on the use case and if there is any room for error. In order to identify similar documents various hash functions are

considered, these hash functions are known as minhash functions. These family of hash functions are then applied to a technique (known as banding technique) to bucket pairs which are at a closer proximity to each other from pairs which are farther away from each other. These hash functions should exhibit the below properties:

- Buckets should consists of hashed pairs which are closer to each other rather than pairs which are farther. These pairs are known as candidate pairs.[3]
- It is important that the functions we use for hashing are independent from each other.[3]
- The time complexity to execute these steps should be less than the time complexity required to compare all the documents. [3]
- Must minimize the no. of errors i.e. false negatives / positives.[3]

Below figure depicts a graph representing the overall probability of hashing similar pairs for a particular function. [7]

at least p_1 if $d(x, y) < d_1$
at most p_2 if $d(x, y) > d_2$

Fig 3.2 (d_1, d_2, p_1, p_2) – sensitive function [7]

Bloom Filter

Bloom filter was developed by Burton H. Bloom in 1970 to simplify change detection and provide constant search time. It uses many hash functions to store the data in a array comprising of only 1's and 0's. It is primarily used for membership queries while dealing with high dimensional data.

There are many applications which require speed and have freedom in terms of accuracy, bloom filter can be an ideal algorithm for such cases. Since it reduces the

search time to constant time and enables you to control the accuracy, it can be put to good use in many applications and domains. Below are certain properties of bloom filters.

The accuracy of the bloom filter can be improved by increasing the number of hash functions.

Let's say S is a set which consists of n items, k be the number of different hash functions. Then in order to have a minimal error rate we would need to determine the ideal number of hash functions required based on the size of the input data [10]. The mathematical equation used to determine these k number of functions is:

$$k = \left(\frac{m}{n}\right) \ln 2$$

Each item from the set S is hashed k times (k being the number of functions) which return k values, resultant bits are set to 1 in the array [10]. If the resultant bit is already set to 1 i.e. a collision occurs, then it is left unchanged.

Now let's say that we have duplicates in set S and we wish to find out what these duplicates in the set. Considering we have very limited memory which is less than the size of set S we can use bloom filter. The crucial point of Bloom Filter is that it takes far less space when compared to other data structures such as Hash Maps, Tries, and Binary Search Trees etc. Let's say that the size of set S to be 25MB then the hash map size would be at least 25MB whereas the bloom filter size would be around 1.1MB. Also Bloom Filters are used to test if a particular item is certainly not present as it guarantees no false negatives.

Now if we take the set S and hash the items using k hash functions then we can determine if an item is duplicate if every bit in the array for resultant bits is 1. Now although bloom filter guarantees no false negatives it does give false positives i.e. it may determine an item as a member of the set S , even if realistically it is not part of the set. This can be minimized by determining the probability of an item being a false positive.

Below figure illustrates bloom filter which uses k hash functions to store n number of items in m bit array.

Fig 3.3 Working of a Bloom Filter

These errors can be reduced by increasing the size of the array. The size of the array is directly proportional to the number of errors that can occur. The false positive rate is approximately $\left(1 - e^{-\frac{kn}{m}}\right)^k$, n being the total number of elements being checked for membership test, various values of m and k are considered to construct an optimal bloom filter depending of the use case. [10]

- If more hash functions are used then it would result in increased time complexity and space complexity as it would result in many values being set to 1 in the bloom filter. [10]
- Similarly if few hash functions are used then it would be quicker but it would result in many false positives. Hence it is important to strike the optimal value using the expression $k = \left(\frac{m}{n}\right) \ln 2$. [10]

Chapter 4

Design Techniques

Sets

Consider n items then the set of n items would be an unordered collection of those distinct items and the size of the set is known as the cardinality

Union, Intersection and Difference are the three main operations which can be performed on sets.

- Set Union is a resultant set which consists of the distinct elements in both set and is denoted by $A \cup B$. The property $A \cup B = B \cup A$ is always true.
- Set Intersection is a resultant set which consists of the elements that are common to both sets and is denoted by $A \cap B$. The property $A \cap B = B \cap A$ is always true.
- Set difference of any two sets say A & B results in a set having all the elements from set A which are not present in set B . The difference of sets does not satisfy the symmetry property as difference of set A with set B is not same as difference of set B with set A .

Since we are dealing with similarity search, two sets A and B have the following properties:

- Small if the distance $D(A, B) < K$; K being the predefined value to determine proximity.

- Large if the distance $D(A, B) > K$; K being the predefined value to determine proximity.
- Is 0 when both A & B are the same.
- It is in range $[0, \infty]$.

Distance Metric

We would be using Jaccard similarity as the default distance function going forward.

Jaccard Similarity [6] of two sets A & B is the ratio of set intersection and set union.

J-SIM (A, B)

$$\text{J-SIM}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

For example, if set A has elements $\{2, 4, 5, 6\}$ and set B has elements $\{1, 3, 4, 5, 6, 7\}$ then $\text{J-SIM}(A, B) = 3/7$.

Shingling

Constructing a document set consisting of words in the document is one of the most efficient ways to determine the lexical similarity of the document. For this purpose shingling is used. This would help determine the similarity between the documents even if the sentences present in the document are not in the same order. Each document is represented as a set of short phrases using shingling. [3]

Choosing shingle size is very important as it helps determine the similarity of the document. Choosing a value too small would result in all documents being identified as similar to each other and choosing a large value to result in no documents being similar to each other. For example, a shingle of size $k = 1$ would result in just the words being used or alphabets being used if the shingling is done on character level. Generally a shingle of size $k = 5$ is preferred for short documents and emails and for web pages and research documents consisting of abundant text a shingle size of $8 < k < 11$ is considered safe.

Using shingles would increase the space complexity, hence instead of using those directly these shingles are hashed using a certain hash function and the resultant bucket number is treated as a shingle itself.

Hashing shingles would reduce the space required but it would still be large and would take n times the size of the document itself. To further reduce the space complexity we need to generate small signatures which preserve the similarity of the sets. This can be done using a technique called min hashing. Before we proceed further first let's take a look at the matrix representation of the sets.

Characteristic Matrix

Characteristic matrix represents two document sets from which we can determine the similarity using the Jaccard Similarity measure. For documents of size N and resultant

shingles of size M the matrix would be of size $(N \times M)$ wherein M being no. of columns and N being no. of rows. Here documents correspond to columns and shingles correspond to rows.

Consider the below matrix representation of four sets:

Table 4.1 Matrix representation of sets [3]

Shingles	S1	S2	S3	S4
A	1	0	0	1
B	0	0	1	0
C	0	1	0	1
D	1	0	1	1
E	0	0	1	0

Here $\{A, B, C, D, E\}$ represent the shingles in sets $S1, S2, S3, S4$. Bit 1 in the matrix denotes the presence of the shingle in the set and 0 denotes the absence of the shingle in the set. [3]

Here sets are $S1: \{A, D\}, S2: \{C\}, S3: \{B, D, E\}, S4: \{A, C, D\}$. [3]

The matrix is helpful in visualizing the data and is not the way the data is actually stored. It only helps in understanding the data. Generally these matrices have a lot of 0's hence it is efficient to store them as a sparse matrix than the above represented way.

Minhashing

Minhashing is a type of LSH technique wherein independent permutations of a given column are used to find out the similarity of sets. Each minhash signature which we

intend to generate are permutations of rows in a given column in characteristic matrix. In principal, to minhash a set using its characteristic matrix [3] we have to permute the values in the columns and obtain the value of the first row in which the column has 1.

Let's consider a permutation of the table 4.1 as described above.

Table 4.2 Permutation of the table 4.1 [3]

Shingles	S1	S2	S3	S4
B	0	0	1	0
E	0	0	1	0
A	1	0	0	1
D	1	0	1	1
C	0	1	0	1

Here we can determine the values of h (h being the minhash function) by doing a column wise scan of the matrix until we encounter the value 1. [3]

Now some of us must be wondering how this random permutation and the minhash signature preserve the similarity of the sets. It gets interesting here as we see Minhashing and Jaccard Similarity [3] share a common property, the JSIM of any two sets is nothing but the probability of determining its signature [3]. To visualize this lets consider the sets S1 and S2, the rows can be considered as follows:

1. Type A rows have 1 in both columns
2. Type B rows have 1 in one of the columns and 0 in other.
3. Type C rows have 0 in both columns

Now let's compute the Jaccard Similarity for both these sets. The Jaccard Similarity can be determined as the number of type B by number of type A. Consider we have x rows of type A and type B has y rows, then $J-SIM(S1, S2) = x / (x + y)$ [3]. This is nothing but $S1 \cap S2 / S1 \cup S2$. [3]

Now let's look at the probability of $h(S1)$ being equal to $h(S2)$ wherein h is a minhash function. Firstly we do a random permutation on the rows, then we perform a column wise scan. The probability of finding a type A row before we come across a type B is $x / (x + y)$ [3]. Hence we can say that the J-SIM of two sets is similar to the probability that the minhash functions of those sets produce the same value. [3]

Minhash Signatures

As described earlier generation of minhash signatures requires n number of permutations wherein each permutation determines the minhash function h , so for n permutations we have minhash functions $h1, h2, h3, h4 \dots hn$. Signature matrix for the given set is of the form $[h1(s), h2(s), h3(s), h4(s) \dots hn(s)]$. Thus in this way we can form a signature matrix [3] M in which we take the generated minhash signature and replace it with every ith column of the matrix. The only difference between the two matrices is that both would consist of the same number of columns but very less rows, thereby drastically reducing the dimensions. [3]

However it is computationally very expensive to perform a random permutation over a high dimensional characteristic matrix. Hence permuted matrices as described in table 4.2 are hard to obtain.

This is where hashing comes in to the picture, by hashing using a custom defined hash function we can simulate the randomized permutation. A custom function is the one which maps rows to buckets [3]. Thus for the characteristic matrix described in table 4.1 instead of n random permutations we can use a family of n hash functions $\{h_1, h_2, h_3, h_4, \dots, h_n\}$. [3]

Table 4.3 Hash Functions computer for Table 4.1 [3]

Shingles	S1	S2	S3	S4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
A	1	0	0	1	1	1
B	0	0	1	0	2	4
C	0	1	0	1	3	2
D	1	0	1	1	4	0
E	0	0	1	0	0	3

Here $(x + 1 \bmod 5)$ and $(3x + 1 \bmod 5)$ are two hash functions which are used to generate the signature matrix. [3]

Below are the steps to generate the signature matrix for the table 4.1. Fill the matrix with ∞ 's.

Hash Function	S1	S2	S3	S4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

Table 4.4 Signature matrix of table 4.1 [3]

Values for 0th row are 1 for both h_1 and h_2 and the 0th row and 1st row have 1 for S1 and S4. The matrix after making changes is [3]:

Hash Function	S1	S2	S3	S4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

Table 4.5 Signature matrix after scan of first row [3]

Moving forward to next row (1st row) we see that only S3 has value 1 and hash values are 2 and 4. Thus, we set S3 to 2 and 4 respectively [3]. All the other values remain the same, the matrix after making changes is:

Hash Function	S1	S2	S3	S4
h_1	1	∞	2	1
h_2	1	∞	4	1

Table 4.6 Signature matrix after scan of second row [3]

Moving forward to next row (2nd row), we see that S2 and S4 have 1's and 3 and 2 as the corresponding hash values [3]. We cannot make changes to S4 since the values present are already less than the corresponding hash values.

Hash Function	S1	S2	S3	S4
h_1	1	3	2	1
h_2	1	2	4	1

Table 4.7 Signature matrix after scan of second row [3]

Moving forward to row 3, we see that S1, S3 & S4 have value 1 and the corresponding hash values are 4 and 0 [3]. Here we replace the values in signature matrix that are greater than the obtained hash values. The resulting matrix after changes is:

Hash Function	S1	S2	S3	S4
h_1	1	3	2	1
h_2	0	2	0	0

Table 4.8 Signature matrix after scan of third row [3]

Moving forward to row 4 we see that S3 has value 1 and the corresponding hash values are 0 and 3 respectively [3]. We now only modify values to s3 in the signature matrix and that too values which are greater than the hash function values. Hence the final signature matrix would be as follows:

Hash Function	S1	S2	S3	S4
h_1	1	3	0	1
h_2	0	2	0	0

Table 4.9 Signature matrix after scan of fourth row [3]

From the obtained signature matrix we can now computer the Jaccard similarities of the sets. [3]

LSH on Signatures

In the above section we have looked at a dimensionality reduction technique which drastically reduces the space complexity while retaining the similarity between the sets.

Now let's look at the time complexity of comparing documents which is still quite high because each document has to be compared with each other resulting in a time complexity of $O(n^2)$.

For example let's consider we have 1 million documents which need to be processed and the final clusters need to be formed consisting of similar documents. Based on minhash signatures we can generate signatures of 250 and 1000 bytes [3]. The resulting space complexity is around 1gigbyte which can easily fit a memory. However this would require 1 trillion comparisons which would take around six days to complete on a decent machine. [3]

In this section we will look at a technique which is designed to solve the above problem, it is known as banding technique and it is a specific form of the locality sensitive hashing family.

LSH in general is implemented in such a way that similar items are hashed into the same bucket as opposed to general hashing [3]. Then the pairs bucketed together are considered as candidate pairs and are compared thereby drastically reducing the overall comparisons.

Banding Technique

Based on the banding technique, the rows in the signature matrix are divided into $(b \times r)$ where b represents the number of bands. Each band consists of maximum r rows. For each band a hash function is defined. The function takes column of its corresponding

band and hashes them to large number of buckets. We can use the same hash function for all

Fig 4.1 Banding Technique

The banding technique might also have some problems, when similar pairs are not hashed into same bucket then false negatives are born and when dissimilar pairs are hashed into same bucket then false positives are born.

Chapter 5

Implementation and Analysis

Implementation:

We will be implementing similarity search using Minhashing by defining certain number of hash functions. First we shingle the documents, we then calculate the actual Jaccard similarity of those documents (for result comparison purpose only), generate minhash signatures for those documents and finally estimate the Jaccard similarity based on the signatures.

Algorithm:

1. Select all documents
2. Convert the text documents in to shingles. The shingle size maybe 3 – 9.
3. Calculate the actual Jaccard Similarity.
4. Optional: Store shingles in bloom filters to improve performance.
5. Generate minhash signatures.
6. Calculate estimated Jaccard Similarity.
7. Compare estimated vs actual similarities.

I have taken a sample test data of 1000 documents. Each document is a news article and we have articles which are similar to each other in the set.

The shingling of documents is first done to convert the documents into sets consisting of shingles of a certain length. Below is a glimpse of how documents look like after shingling documents into n shingles of size 3.

```
document8660 : set(['star _ is', 'day ahead of', 'force Saturday in', 'nowadays _ especially', 'newest comedy, ``Hollywood', 'of a banned', 'the highe
document8477 : set(['and caused delays', 'to 7 at', 'left 25 civilians', 'the invitation of', 'final-round three-under-par 69', 'at the Stockholm', 'f
document8660 : set(['star _ is', 'day ahead of', 'force Saturday in', 'nowadays _ especially', 'newest comedy, ``Hollywood', 'of a banned', 'the highe
document8661 : set(['is being urged', 'Turkish government warned', 'to make a', 'will "endanger relations"', 'to mend fences', 'multi-million dollar c
document8662 : set(['has retracted it', 'the emergence of', 'time since taking', 'motion against the', 'conducted the premiere', 'over 200 people', 'D
document8664 : set(['heard of the', 'funded by the', 'Friday. The iconic', 'Hugh Hefner stepped', 'after Playboy mogul', 'Angeles has been', 'informat
document8665 : set(['governing body FIFA', 'favorite ballparks and', 'aggressively polarised between', 'of academic-centered thinking', 'in one of',
document9516 : set(['she may have', 'Singapore said Thursday', 'mailing one to', 'right hand, and', 'Abbas said Sunday', 'in which he', 'were supporte
document9530 : set(['its highest level', 'after the Passover', 'said Thursday amid', 'Darfur region. She', 'where crowds cheered', 'Open, where crowds
document9531 : set(['terms in office.', 'its facilities, bolstering', 'Korea, which have', 'America" has suffered', 'its support for', 'strike by Almu
document9532 : set(['Minister Milorad Dodik', 'year with children', 'the Basel Furniture', 'Abbas, part of', '1.5 percent in', 'arriving at the', 'rod
document9533 : set(['Singapore CableVision (SCV)', 'making loans --', 'rules applied to', 'Colin Powell said', 'a flood of', 'CableVision (SCV) Pte.',
document9534 : set(['The results then', 'forcing the evacuation', 'for the previous', 'the Gulf of', 'to break. Frederick', 'and Industry Ministry', '
document9535 : set(['two charges by', 'Could my son', 'Pyongyang as their', 'Engine" on television', 'organisation, a monitoring', 'of kidnapping his
document9536 : set(['computer centers in', 'after the Passover', 'summit of world', '80 police and', 'Williams likes her', 'was prepared to', 'firebom
document9537 : set(['ever single grant', 'to help rehabilitate', 'Korea, which have', 'organisation, a monitoring', 'court on Monday', 'become big exp
```

Fig 5.1 Shingles of documents

Now let's apply min-hash algorithm on the shingles and generate signatures for each document. These signatures hold essence of each document at the same time they are small enough to fit into the main memory. Below sample minhash signatures are for couple of documents using 8 hash functions.

```
[39403359L, 14791228L, 4502706L, 25101529L, 19389490L, 35131354L, 13264699L, 20110126L]
[5910110L, 29106054L, 26723033L, 4463779L, 3355757L, 21862382L, 4481925L, 55508294L]
[43736037L, 12621371L, 15670259L, 18548051L, 33055735L, 29830040L, 32206363L, 29698222L]
[14624534L, 5417362L, 1531336L, 3963659L, 19389490L, 30976236L, 25396856L, 58367973L]
[10821068L, 1257802L, 26136525L, 48121006L, 7947840L, 9899379L, 1016030L, 3911138L]
[22086536L, 2225942L, 11689831L, 30737624L, 19389490L, 17573744L, 48071347L, 77607658L]
[4618322L, 6940146L, 9075346L, 4463779L, 12701521L, 4401477L, 5118035L, 9426059L]
[43402687L, 3848594L, 9167671L, 569738L, 13334575L, 17529628L, 40088463L, 17896672L]
[47879122L, 36202817L, 4687369L, 548026L, 11485375L, 7205817L, 31656362L, 1201552L]
[33821269L, 2086079L, 42563313L, 2533948L, 26125269L, 9726596L, 1016030L, 2247696L]
[2372831L, 18407006L, 5510156L, 10744588L, 24449623L, 29791718L, 1838384L, 1415899L]
[7565070L, 9339513L, 3200539L, 24568286L, 27445701L, 2174565L, 33217843L, 26817128L]
[979007L, 866137L, 25380521L, 548026L, 5367034L, 613187L, 31428055L, 2608304L]
[13356738L, 8002186L, 556783L, 8738581L, 680026L, 10370680L, 16272547L, 3017670L]
[42697950L, 19527814L, 110651466L, 4401216L, 16316584L, 35290049L, 7833854L, 11727916L]
[5865672L, 6553512L, 9124493L, 6043343L, 7858630L, 4401477L, 5118035L, 10887202L]
[23937133L, 15652234L, 48267527L, 1260246L, 17136069L, 351366L, 953766L, 13715943L]
[50493564L, 9339513L, 24342325L, 16775487L, 3340357L, 10193932L, 14511891L, 9540779L]
[24373629L, 9339513L, 22447356L, 48664071L, 47368192L, 48088081L, 22691050L, 2162721L]
[5910110L, 15834679L, 8010583L, 25343941L, 9419942L, 24585408L, 25396856L, 6043533L]
[2867333L, 2668041L, 5082029L, 6874976L, 1426676L, 76511486L, 1417014L, 2135179L]
```

Now when we run all the steps we see the below result. Below output shows the process and the time taken to complete each process followed by estimated and actual values. Only documents with a threshold of above 0.5 are being displayed.

```

-----
Document Shingling has started
-----
>> Total time taken to shingle 1000 documents is 1.06 sec.
>> Average number of shingles per doc: 251.24

-----
Calculating the actual JSIM has started
-----
>> Documents currently being processed (0 / 1000)
>> Documents currently being processed (250 / 1000)
>> Documents currently being processed (500 / 1000)
>> Documents currently being processed (750 / 1000)
>> Calculating Jaccard Similarities for all documents took 51.36sec

-----
Generating minHash Signatures has started
-----
>> Generating MinHash minHashSignatures for all documents took 31.61sec

-----
Comparing minHash Signatures has started
-----
>> Comparing MinHash minHashSignatures took 18.08sec

Estimated Jaccard Similarity VS Actual Similarity

      Est.      Act.
t980 <--> t2023 1.0000 0.9792
t1088 <--> t5015 0.9888 0.9805
t1297 <--> t4638 1.0000 0.9806
t1768 <--> t5248 0.9663 0.9803
t1952 <--> t3495 0.9775 0.9784
t2535 <--> t8642 1.0000 0.9811
t2839 <--> t9303 0.9888 0.9821
t2957 <--> t7111 0.9775 0.9817
t3268 <--> t7998 1.0000 0.9772

```

Fig 5.2 Estimated JS vs Actual JS

The estimated Jaccard Similarities are quite close to the actual values. We would varying results for each run as the hash functions are being randomly chosen. For example below is the result for a subsequent run.

```

-----
Document Shingling has started
-----
>> Total time taken to shingle 1000 documents is 1.11 sec.
>> Average number of shingles per doc: 251.24

-----
Calculating the actual JSIM has started
-----
>> Documents currently being processed (0 / 1000)
>> Documents currently being processed (250 / 1000)
>> Documents currently being processed (500 / 1000)
>> Documents currently being processed (750 / 1000)
>> Calculating Jaccard Similarities for all documents took 51.71sec

-----
Generating minHash Signatures has started
-----
>> Generating MinHash minHashSignatures for all documents took 24.49sec

-----
Comparing minHash Signatures has started
-----
>> Comparing MinHash minHashSignatures took 14.63sec

Estimated Jaccard Similarity VS Actual Similarity

      Est.      Act.
t980 <--> t2023 0.9706 0.9792
t1088 <--> t5015 0.9706 0.9805
t1297 <--> t4638 0.9853 0.9806
t1768 <--> t5248 0.9559 0.9803
t1952 <--> t3495 0.9559 0.9784
t2535 <--> t8642 0.9706 0.9811
t2839 <--> t9303 0.9706 0.9821
t2957 <--> t7111 0.9853 0.9817
t3268 <--> t7998 0.9853 0.9772

```

Fig 5.3 Estimated JS vs Actual JS for another run

We see that the values are consistent even with varying hash functions.

The whole point of using minhash is to speed up the process. Below is a table which depicts the running time of calculating Jaccard similarity vs minhash signatures. As the number of documents increase, Jaccard similarity increases in quadratic time complexity.

Table 5.1 Time taken for JS vs MinHash

No. of Documents	JSIM	MinHash
100	0.51 Sec	2.41 Sec
200	2.05 Sec	4.80 Sec
400	8.33 Sec	9.65 Sec
800	29.95 Sec	18.35 Sec
1600	130.34 Sec	38.40 Sec

As the number of documents increase we see an exponential increase in the time taken to calculate JS whereas we see a linear increase in time taken to calculate their corresponding minhash signatures.

Below is a graph plotted against above mentioned results.

Fig 5.4 Time Taken JSIM vs Min-Hash

Analysis:

We have seen the implementation of Min hash algorithm in the above section, let's now understand the working of the algorithm and specify the areas in which the algorithm would be useful and successful.

The algorithm is somewhat dependent on the order in which the text appears hence it can be used for specific cases and will not be successful universally. Let's look at the below example:

A man was shot dead and fifteen others injured when Zambian policemen clashed with citizens rioting in protest against alleged ritual murders by a local businessman, police said Monday. U.S. President George W. Bush expressed confidence on Monday about passing an immigration bill and said a Senate vote of no-confidence in Alberto Gonzales would have no bearing on his service as attorney general. French President Nicolas Sarkozy announced Tuesday in Washington that he would visit China later this month, joined by his wife, Carla Bruni-Sarkozy. These columns for release Tuesday, April 2, 2002 are moving today to clients of the New York Times News Service. Media and entertainment giant Viacom Inc. said Wednesday it may split into two divisions with one focussing on "growth" and the other, more traditional arm, aiming for "value". I don't know if ``Harry Potter and the Order of the Phoenix'' is a good movie -- I haven't seen it. But I'm pretty certain that it shouldn't be judged as a movie at all. It is a visual representation of a book in which millions of people are Two car bombs blew up Monday in the working class town of Yehud, just hours after three Palestinian militants were killed by missiles fired by an Israeli helicopter in a pinpointed attack. Australia's farmers should remember rising fuel prices were hitting farmers worldwide and not just them, Deputy Prime Minister and National Party Leader John Anderson said Tuesday.

A man was shot dead and fifteen others injured when policemen clashed with pedestrians rioting in protest against alleged murders by a local businessman, police said Monday. U.S. President expressed confidence on Monday about passing an immigration bill and said a Senate vote of no-confidence in Alberto Gonzales would have no bearing on his service as attorney general. French President announced that in Washington that he would visit China later this month, joined by his wife, Carla Bruni-Sarkozy. These columns for release Tuesday, April 2, 2002 are moving today to clients of the New York Times News Service. Media and entertainment giant Viacom Inc. said that it may split into two divisions with one focussing on "growth" and the other, more traditional arm, aiming for "value". I don't know if ``Harry Potter and the Order of the Phoenix'' is a good movie -- I haven't seen it. But I'm pretty certain that it shouldn't be judged as a movie at all. It is a visual representation of a book in which of people are Two car bombs blew up Monday in the working class town of Yehud, just hours after three Palestinian militants were killed by missiles fired by an Israeli helicopter in a pinpointed attack.

The above documents are almost similar, when we run these against our program, we see the following result.

	Est.	Act.
t0980 <--> t1190	0.7206	0.7237

Now let's permute the sentences of the 2nd document and run them against our algorithm. Permuting sentences would disrupt the order of the document. Since we are considering shingles of size 3 this would impact the similarity of the documents.

2002 are moving today to clients of one focussing on "growth" and the other, more traditional arm, aiming for "value". A man was shot dead and fifteen others injured when policemen clashed with pedestrians rioting in protest against a movie at all. It is a visual representation of a book in which of people are Two car bombs blew up Monday in the alleged murders by a local businessman, police said Monday. U.S. President expressed confidence on Monday about passing an immigration bill and said a Senate vote of no-confidence in Alberto Gonzales would have no bearing on his service as attorney general. French President announced that in Washington that he would visit China later this month, I don't know if ``Harry Potter and the New York Times News Service. Media and entertainment giant Viacom Inc. said that it may spl in a pinpointed attack. joined by his wife, Carla Bruni-Sarkozy. These columns for release Tuesday, April 2, the Order of the Phoenix'' is a good movie -- I haven't seen it. But I'm pretty certain that it shouldn't be judged as working class town of Yehud, just hours after three Palestinian militants were killed by missiles fired by an Israeli helicopter

The result obtained is:

	Est.	Act.
t0980 <--> t1190	0.6765	0.6014

Now let's sort the whole document and see the result, this would disrupt the order of the document completely. Below document is sorted based on words in the document

"growth" "value".A -- 2, 2002 ``Harry a a a a a a about after against aiming Alberto all. alleged an an and and and and and announced April are are arm, as as at attack. attorney be bearing bill blew bombs book Bruni-Sarkozy. businessman, But by by by by car Carla certain China clashed class clients columns confidence dead divisions don't entertainment expressed fifteen fired focussing for for French general. giant Gonzales good have haven't he helicopter his his hours I I I'm if immigration in in in in in in Inc. injured into is is Israeli it It it it. joined judged just killed know later local man may Media militants missiles Monday Monday Monday. month, more movie movie moving murders New News no no-confidence of of of of of on on on on one Order other, others Palestinian passing pedestrians people Phoenix'' pinpointed police policemen Potter President President pretty protest release representation rioting said said said seen Senate service Service. shot shouldn't split that that that that the the the the the These this three Times to today town traditional Tuesday, Two two U.S. up Viacom visit visual vote was Washington were when which wife, with with working would would Yehud, York

The result obtained is:

	Est.	Act.
t0980 <--> t1190	0.0441	0.0168

We see that the accuracy takes a big hit and both the documents are considered not similar at all.

To summarize accuracy decreases if the content of the documents are different (although the meaning may be same) and if the document is jumbled or permuted. Hence there are limited use cases for which the algorithm would be successful. They are:

1. Near De-dup Detection
2. Plagiarism Detector
3. Similar Web Documents
4. Image Clustering

Use cases for which the mentioned approach is not suitable are:

1. Pattern Matching
2. DNA Analysis
3. Author Paper Identification

The major advantage of using this approach is due to its speed and its close relationship with Jaccard Similarity.

Analysis Based on Number Hash Functions

Number of hash functions plays a crucial role in estimating the Jaccard similarity, too few hash functions would result in poor estimation and too many hash functions would increase the time of computation. Total number of hash functions should be chosen based on the document size, documents of small size would require less number of hash functions and as the document size increases the hash functions should be increased. For documents consisting of thousands of shingles, about 100 - 200 hash functions should suffice.

Below is the change in accuracy based on the number of hash functions:

Table 5.2 Similarity est. by no. of hash functions

No. of Hash Functions	Estimated Similarity	Actual Similarity
1	0	0.67
5	1	0.67
10	0.6	0.67
15	0.66	0.67
30	0.73	0.67
45	0.72	0.67
70	0.71	0.67
90	0.72	0.67
150	0.71	0.67

As the number of hash functions increases the estimation improves but after a point the increase in number of hash functions has no effect on the estimation.

Comparison with Alergia:

We can compare our approach with Alergia, let's take the above two documents and input them to alergia. Here only the function words are considered in the sentences and are used to determine the similarity.

Table 5.3 Alergia results t0980 vs t1190

Results:		Alpha	Doc 1	Doc 2
Doc 1	Training	0.1	100	100
Doc 2	Test	0.2	100	100
		0.3	100	100
		0.4	100	100
		0.5	100	100
		0.6	100	90
		0.7	100	90
		0.8	100	80
		0.9	100	80

We see that this gives us the expected similarity.

Now when we permute the document and sort it alphabetically, we see the below result:

Table 5.4 Alergia results t0980 vs permuted t1190

Alpha	Doc 1	Doc 2
0.1	100	20
0.2	100	20
0.3	100	20
0.4	100	20
0.5	100	20
0.6	100	0
0.7	100	0
0.8	100	0
0.9	100	0

The result determines the importance of order in the sentences. However Alergia better fits cases such as author paper identification problem wherein each author maintains a pattern subconsciously in his writing which is captured by Alergia.

Applying DNA data to Min-Hash:

For this we have taken sample DNA for human, mouse, chimpanzee, monkey & rat.

The DNA sequence of these species is chopped using start and stop codons and is input to the algorithm. The processed DNA would look like something as below:

```
01HUMAN ATGGTGTTCGGGAAGAAAATAA ATGGCAG ATAA ATAA ATAA ATAA ATAA ATGCAGGATCTTTTAA ATGG ATGGTG ATGTTGCTAA ATGTCACGTAA ATAA ;
02CHIMP ATGGTATTCGGGAAGAAAATAA ATGGCAG ATAA ATAA ATAA ATAA ATAA ATGCAGGATCTTTTAA ATGG ATGGTG ATGTTGCTAA ATGTCACGTAA ATAA ;
03MONKEY ATGGTGTTCGGGAAGAAAATAA ATGGCAG ATAA ATAA ATAA ATAA ATAA ATAA ATGCAGGATCTTTTAA ATGCCAGTACAACAGATACCCAGAACTTCGCAG;
04MOUSE ATGGTTTTCGGGAAGAAAATAA ATGGCGG ATAA ATAA ATAA ATGGCCAATCAGAACTGCCACACCAGCGCTGGCACAGCCACAACCAGCAAAGCCGGTGGAGCGGAAG;
05RAT ATGGTTTTCAGAAAGAAAATAA ATGGCGG ATAA ATAA ATAA ATAA ATAA ATGCAGGGTCTTTTCAGCCCAAATAA ATGG ATGGAGACGTTGCTAA ATGTGGACGAA;
```

Fig 5.5 Processed DNA input data

The gaps are present to divide the DNA into shingles and determine the similarity. In DNA the sequence matters.

Table 5.5 Min Has Results on DNA

Shingle Size	Comparision	Est.	Act.
1	01HUMAN <--> 02CHIMP	0.75	0.7561
1	01HUMAN <--> 03MONKEY	0.325	0.364
1	01HUMAN <--> 04MOUSE	0.105	0.102
1	01HUMAN <--> 05RAT	0.08	0.09478
2	01HUMAN <--> 02CHIMP	0.67	0.711
2	01HUMAN <--> 03MONKEY	0.25	0.275
2	01HUMAN <--> 04MOUSE	0.065	0.067
2	01HUMAN <--> 05RAT	0.035	0.06
3	01HUMAN <--> 02CHIMP	0.64	0.66
3	01HUMAN <--> 03MONKEY	0.215	0.204
3	01HUMAN <--> 04MOUSE	0.055	0.033
3	01HUMAN <--> 05RAT	0.045	0.028

Surprisingly we see a good match between human and chimpanzee DNA while there is little to no match between the other species. Although the algorithm has been designed and best fits text data with similar content, inputting DNA sequence data to the algorithm has given a decent result especially for human and chimpanzee DNA. On closer observation we see that the processed sequence for human and chimpanzee is pretty much similar.

Chapter 6

Conclusion

In this project, we have explored similarity search problem using techniques such as Minhashing, Locality Sensitive Hashing and Bloom Filters. Hashing has been implemented upon signature matrix and banding technique where signature matrix is created using characteristic matrix and a family of hash functions.

Characteristic matrix is built using the process of shingling which is then converted into minhash signatures using a family of minhash functions. These minhash signatures and the signature matrix have a direct correlation with the Jaccard similarity i.e. the probability of two minhash signatures being the same is equal to the Jaccard Similarity of the two sets.

Chapter 7

Future Scope

In this project we have explored various techniques of improving the similarity search using probabilistic algorithms such as Bloom Filters, Linear time hashing algorithms such as Minhashing and Locality Sensitive Hashing. If we carefully consider the implementation then we see that couple of areas can use improvement such as the issue with false positives with Bloom Filters and Locality Sensitive Hashing and similarly defining an optimal hash functions for these algorithms is also an important part of the process.

Selecting better hash functions would minimize the false positives and negatives issue and also one can determine the bloom filter size and number of hash function by using some statistical and probabilistic methods.

The algorithm can be utilized for parallel / distributed processing environments for tasks such as shingling of documents and signature generation. Tools like Apache Spark or Map Reduce programs can be used to achieve high level of parallel processing / distributed processing.

Chapter 8

References

- [1] J. R. Smith, “Integrated Spatial and Feature Image Systems: Retrieval, Analysis, and compression”, Columbia University, 1997.
- [2] “Shingles and near-duplicates” [online] available at <http://nlp.stanford.edu/IRbook/html/htmledition/near-duplicates-and-shingling-1.html>
- [3] “Finding Similar Items” [online] available at <http://infolab.stanford.edu/~ullman/mmds/ch3.pdf>
- [4] U. Manber, “Finding similar files in a large file system”, Proc. USENIX Conference, 1994.
- [5] A. Awekar and N. F. Samatova, “Fast Matching for All Pairs Similarity Search”, ISept. 2009, Milan, Italy.
- [6] A. Rajaraman, J. Ullman, “Mining of Massive Datasets”, Cambridge University Press, December 30, 2011.
- [7] S.S. Chauhan and S. Batra, “Finding Similar Items using Locality Sensitive Hashing and Bloom Filter” Thapar University, Patiala, June, 2014.
- [8] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions”, Comm. ACM, pp. 117-122, 2008.
- [9] “LSH and General hashing” [online] available at http://cybertron.cg.tuberlin.de/pdci08/imageflight/nn_search.html
- [10] Ripeanu, Matei & Adriana Iamnitchi, "Bloom Filters Tutorial," 2001, <http://people.cs.uchicago.edu/~matei/PAPERS>