J. Basic. Appl. Sci. Res., 3(1s)466-472, 2013
© 2013, TextRoad Publication
ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Solution for Calculating the False Positive and False Negative in LSH Method to Find Similar Documents

Hossein Azgomi[1] and Ali Mahjur[2]

[1]Department of Computer Engineering, Islamic Azad University of Shabestar, Hossein.
[2]Malek Ashtar University of Technology,

Received: June 2 2013
Accepted: July 2 2013

ABSTRACT

One of the fundamental issues in data mining is exploring the data to find similar items. For example, one can point at the investigation of a number of different documents for finding items which are close to each other or are repetitive. Finding similar documents can be turned into a problem based on sets and this will be done in the Shingling method. Moreover, the resulting big sets can be compressed in such a way that the similarity of the main sets may be distinguished from their compressed versions and this will be done through the Minhashing method. If we decide to avoid comparing all the documents with each other and only compare the documents with high probability, we should use the LSH method. In this method, we have the ability to create false positive and false negative. In this paper, we have presented relationships for calculating the false positive and false negative in this method. Using these relationships, the parameters of the LSH method can be selected in such a way that false positive and false negative would be controlled. Actually, the main objective of paper is to design a solution for decreasing the false positive or false negative
KEYWORDS: data mining, similar documents, LSH, false positive, false negative.

1. INSTRUCTION

With the growth of Information Technology (IT) and the methods of data production and collection, the volume of databases related to daily exchanges is steadily increasing. As a result of competition in different areas, getting access to the data included in this collection of massive data in the shortest time and without the direct intervention of the man is really important. Therefore, these items lay the foundation for the science of data mining. One of the important issues pertaining to the science of data mining is finding the similar items and documents [1]. The objective of this branch of science of data mining is to design methods for finding the similar items and these items belong to a massive dataset. For instance, this massive collection may include all the pages in a webpage or a great amount of textual documents and news stories. In the present paper, we focus on finding similar documents [2]. It should be noted that the aspect of similarity which we try to discover in our article is the similarity in the level of words and characters of a text, and similarity in the meanings and the concept of the texts is not our concern.

In the present paper, we have firstly touched upon the issue of similarity, and the similarity which is sought for the documents has been explained. Afterwards, the Shingling, Minhashing and LSH methods which are used for finding similar items have been investigated. After these concepts were reviewed, a method for calculating the value of false positive and false negative resulting from the three abovementioned methods are offered and their practical results for different values have been presented subsequently. Finally, the conclusion and the vision of the future assignments have been clarified.

2. The issue of similarity

Firstly, one should concentrate on a certain concept of similarity. By similarity, here we mean a similarity between the datasets which is dependent on their intersection. Having this concept in mind, we are looking to find the closest items to a certain item or element. These elements can include different materials such as documents, webpages, products, customers, etc [3]. To analyze this sort of similarity, we analyze and investigate the Jaccard Similarity.

2.1. Jaccard Similarity

The Jaccard Similarity of two sets is equivalent to the proportion of the value of the intersection of sets to the value of their union. The Jaccard Similarity of two T and S sets will be shown as SIM (S, T) and is calculated through equation 1 [4]:

$$SIM(S,T) = \frac{S \cap T}{S \cup T} \quad (1)$$

One set of the concepts which the Jaccard Similarity covers in a perfect manner is the concept of finding similar textual documents in a big set [5]. As it was noted previously, that aspect of similarity which we are looking for here is similarity in the level of characters in a text not similarity in the meanings of the texts. That's why we should investigate the characters existing in the documents. Finding similarities in the texts has different applications some of which include finding repetitive texts, texts which are similar to each other and detecting plagiarism.

Corresponding Author: Hossein Azgomi, Department of Computer Engineering, Shabestar Branch, Islamic Azad University, Iran. Email address: Hossein.Azgomi@gmail.com

3. Relevant solutions

3.1. Shingling method

The most effective solution for displaying documents in the form of datasets is to select strings from the documents and put them into sets. If we do so, the documents which are divided into small parts of sentences or even phrases will have several common elements which may appear in different sequences in two documents. Using the Jaccard Similarity of these sets, one can calculate the quantity of their similarity [6].

As we know, a document is a string consisted of characters. For a document, a *k*-shingle will be any sub-string with the length of *k* which has appeared in the document. In this method, we select a set of k-shingles from every document which have been repeated once or several times inside it and allocate it to the document [5].

For instance, suppose that the document *D* contains the string abcdabd and the value of *k* is equivalent to 2. Then the 2-shingles collection for the document *D* will be in line with the {ab, bc, cd, da, bd}. Please note that the sub-string ab has appeared in the document *D* twice, but has been considered as shingle only one time. We can select any constant value for *k*. If the value which we select for *k* is too small, then several strings selected with the length of *k* will exist in the majority of the documents. In this case, we will have a set of shingles which have a high Shingle Similarity, while perhaps they may have no similar sentence or phrase. For example, if we take the value of *k* as 1 (k = 1) then the majority of webpages will have numerous joint characters and a small number of uncommon characters. In this case, all the webpages will have a high similarity [7]. The fact that to what extent our *k* is large depends on the length of the documents and the largeness of the collection of the characters. At any rate, the important point which we should bear in mind is that *k* should be selected large enough so that the possibility that every specific shingle appears in every document may be less [5].

3.2. Minhashing

The datasets containing shingles have a high volume. If we have millions of documents, it might be possible that restoring all the shingles in the main memory is not plausible. The principal objective of this method is to replace the big datasets with small representations which are literally called signature. The main feature which we need for the signatures is being able to compare the signatures of two collections and estimate the Jaccard Similarity of the collections associated with the signatures [8].

To create a signature, we first need to display shingle collections using a matrix. This matrix which can show all the collections is called “characteristic matrix.” The columns of this matrix correspond to the sets and its rows correspond to the items of universal set which are among the constituents of all the sets. In the row *r* and column *c* of this matrix, the value “1” will be put if the element of the row *r* is a member of the dataset of column *c*. Otherwise, a “0” value will be put in (r, c) set [8]. For instance, the Fig. 1 shows a characteristic matrix for the sets S_1 = {a, d}, S_2 = {c}, S_3 = {b, d, e} and S_4 = {a, c, d} and the universal set {a, b, c, d, e}.

Element	S_1	S_2	S_3	S_4
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

Fig. 1. A characteristic matrix for showing four sets

The signatures which we want to create for the sets are consisted of the results of several calculations and each of these calculations are a minhash of the characteristic matrix. To calculate the value of minhash for one set which is shown in one column of the characteristic matrix, we should select a permutation of the rows. In this case, the value of minhash in each column is equal to the number of the first row (in the permutation sequence) where the column has the value of “1.” For instance, suppose that the sequence of rows in the Fig. 1 is beadc. This permutation defines a minhash function named “h” which inscribes the sets on the rows. It means that the minhash function replaces the lines of characteristic matrix [5]. The characteristic matrix will look like the Fig. 2 once the permutation has been applied to the rows.

Element	S_1	S_2	S_3	S_4
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

Fig. 2. A permutation of the rows of the matrix shown in Fig. 1

Now, to calculate the minhash of the S_1 set, we investigate the column related to it and consider the first row which has the value of “1” as the minhash value. Therefore, $h(S_1)$ = a. Similarly, we have $h(S_2)$ = c, $h(S_3)$ = b and $h(S_4)$ = a.

The important feature of the minhashing method is that there’s a remarkable relationship between this method and the Jaccard similarity of the sets which have been subject to minhash. In this relationship, the possibility that a minhash for two sets in a random permutation of the rows may produce identical values is equal to the value of Jaccard Similarity in these two sets. It means that for the two S_1 and S_2 sets, the possibility that $h(S_1) = h(S_2)$ equals $SIM(S_1, S_2)$ [8].

After reviewing the method of calculating minhash, we now want to create a characteristic matrix using several minhash functions. For this purpose, we first select n random permutations of the rows of the characteristic matrix. The minhash functions are determined using h_1, h_2, ... h_n permutations. Now for each column of the characteristic matrix like S, we calculate the $[h_1(S), h_2(S), ..., h_n(S)]$ vector and this vector is a signature of the minhash of the S column. Therefore, we can replace the minhash signature of the column "i" with the "i" column in the characteristic matrix and turn it into a characteristic matrix. It should be noted that the number of columns in the signature matrix is only equal to "*n.*" Therefore, the volume of a signature matrix is immensely less than the volume of characteristic matrix. Moreover, given the relationship which the minhashing method has with Jaccard Similarity, the similarity of the documents will be preserved to some extent and will not disappear after this transformation. One should also pay attention to the fact that practically, it's not possible to apply permutation to a big characteristic matrix. Even selecting a random permutation from among a great number of rows and bringing the lines into order based on permutation will be pretty time consuming. As a result, we apply permutation to the matrix using hash functions [9].

Given the descriptions provided, it's possible to provide a general algorithm regarding the calculation of minhash signatures. So, instead of selecting *n* random permutations from the rows, we randomly select *n* hashing functions of h_1, h_2, …, h_n from the rows. We consider SIG (*i*,*c*) an element of the signature matrix for the hashing function *i* and column *c*. Firstly, we put SIG (*i*,*c*) equal to infinity for all the "i"s and "c"s. To calculate the signature matrix, we browse the rows of characteristic matrix one by one. We calculate each r line by going through the following steps [5]:

1) We calculate the $h_1(r)$, $h_2(r)$, …, $h_n(r)$
2) For each *c* column, we do the following steps
   a) If the column *c* in the row *r* has a value of "0", we won't take any action
   b) if the column *c* in the row *r* has a value of 1, then for each $i = 1, 2, …, n$, if the value of $h_i(r)$ is smaller than SIG (*i*,*c*), then we will take the value of SIG (*i*,*c*) equal to $h_i(r)$.

Now we can write the pseudo-code of the algorithm explained above in the form of the Fig. 3.

```
For each row r
  For each column c
    If c has 1 in row r
      For each hash function h_i do
        If h_i(r ) is a smaller value than M(i, c ) then
          M(i, c ) := h_i(r);
```

Fig. 3. The pseudo-code of the algorithm of producing the signatures of minhash

Now having this abovementioned algorithm in mind, we can calculate the minhash matrix of the Fig. 1. We first need to select *n* minhash functions. For this example, we put *n* equal to 2 and select two hashing functions of $h_1(x) = x+1 \bmod 5$ and $h_2(x) = 3x+1 \bmod 5$. Then name the rows using the numbers 0 to 4 instead of letters. In the Fig. 4, you can find the characteristic matrix along with the values calculated for the hashing functions for each row.

Row	S_1	S_2	S_3	S_4	x+1 mod 5	3x+1 mod 5
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Fig. 4. The values of the minhash function selected for the matrix shown in the Fig. 1

Now, to calculate the signature matrix which has four rows and two columns, we first put all of its items equal to ∞ and then implement the algorithm on the characteristic matrix given the values calculated before. After the implementation of the algorithm and investigation of all the lines of the characteristic matrix, the resulting minhash matrix will look like what is shown in the Fig. 5.

Functions	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Fig. 5. Minhash signature matrix for the matrix shown in the Fig. 1

As it was said earlier, the similarity of the sets can be estimated by looking at the signature matrix, but this estimation will be precise when the matrix is too big; the estimation won't be accurate enough for very small samples.

3.3. LSH method

An important issue in the methods of finding similar items is that when we look for similar items, we may not be simply trying to calculate the similarity of a pair of documents. It means that generally, we may want to compare all the documents with each other and identify the similar documents from among them. As it's evident, investigating the whole pairs of the documents is pretty much time-consuming [10]. To solve this problem, we should focus our attention only on the pairs which are most probably similar, and avoid investigating every pair. There's a general solution for this task which is named LSH (Locality Sensitive Hashing) [11]. A general theory for the LSH method is that we should hash the

items into each other several times in such a way that the identical items which are closely similar to each other be hashed into a common bucket. Then in each of the hashings, each pair which is connected to one bucket will be considered as a candidate pair. To find the similar documents also we only investigate the candidate pairs. In this case, all the pair documents won't be investigated [5].

If we have a minhash signature of the items, one pertinent way to implement the LSH method is to divide the signature matrix into *b* bands in such a way that each band includes *r* rows (n = br). Then for each band, we will select a hashing function and this vector function will give numerous buckets using *r* integers along with a great number of hashing for them. We can use the same hashing function for all bands, but we use a separate array for hashing in each band so that columns with similar vectors are not connected to the same buckets in different bands [10]. For instance, the Fig. 6 shows parts of a signature matrix consisted of 12 lines which are divided into 4 bands and each band has three rows [5].

band 1	**...**	**1 0 0 0 2** **3 2 1 2 2** **0 1 3 1 1**	**...**
band 2			
band 2			
band 4			

Fig. 6. Dividing a signature matrix into 4 bands and 3 rows in each band

As it can be seen, the second and fourth column in the first band show the [0, 2, 1] vector and they are definitely connected to the same bucket in the hashing of the first band. Therefore, regardless of the fact that these columns are similar in three other bands or not, this pair of columns can be considered a candidate pair. Moreover, the two columns which do not conform in the first band have this chance to become a candidate pair in the three other bands. Actually, it's possible that they are similar in each of the other bands. At any rate, two extremely similar columns are most probably similar in some of the bands. Therefore, this method directly turns the similar columns, with a higher probability as compared to the dissimilar pairs, into a candidate pair [5].

3.3.1. Analyzing the LSH method

Suppose that in this method, we use *b* bands each of which contains *r* rows. Moreover, suppose that a certain pair of the documents have a Jaccard Similarity with a value of *s*. As it was noted, we know that the possibility that the minhash signatures for these documents in each certain line of the signature matrix are in line with each other equals *s*. The possibility that these documents (or preferably their signatures) are a candidate pair can be calculated as follows [11]:

1) The possibility that the signatures in all lines from a certain band are in line with each other equals s^r.
2) The possibility that the signatures in at least one line from a certain band are not in line with each other equals $1 - s^r$.
3) The possibility that the signatures in all lines from each band are not in line with each other equals $(1 - s^r)^b$.
4) The possibility that the signatures in at least one band and all lines of that band are in line with each other and consequently a candidate pair equals $1 - (1 - s^r)^b$.

Regardless of the *b* and *r* constants, this function of $1 - (1 - s^r)^b$ looks like a curve which is shown in the Fig. 7 [11].

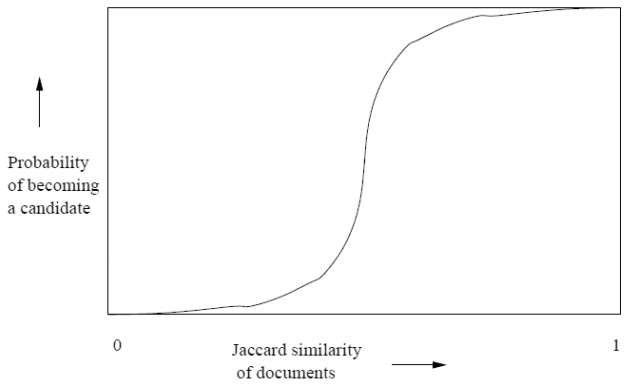

Fig. 7. The curve of $1 - (1 - s^r)^b$ function

Threshold limit is the quantity of similarity for *s* in which the ascending course of the graph reaches its highest limit. The quantity of threshold limit actually determines the quality of similarity. The quality of similarity is dependent on the false positive and false negative. This threshold limit is a function of *b* and *r* and its quantity is approximately equal to $t = (1/b)^{1/r}$ [11].

4.The method suggested for calculating the false positive and false negative

In the LSH method, there's a possibility for the emergence of false positive and false negative. False positives are dissimilar pairs which are hashed to the same bucket, and false negatives are similar pairs which are not dispatched to the same bucket. It means that the false positives are pairs which are mistakenly considered as a candidate pair and the false negatives are pairs which are mistakenly not considered as a candidate pair.

As it was demonstrated, in the LSH method, the possibility that two pairs with the Jaccard Similarity of *s* may become a candidate pair equals $1 - (1 - s^r)^b$. The surface below the graph in this function from zero to the threshold limit (t) is equal to the quantity of false positive, because we expect that the pairs whose Jaccard Similarity is less than t will not be considered a candidate pair. In the Fig. 8, the graph of this function along with the areas related to the quantity of false positive and false negative are shown.

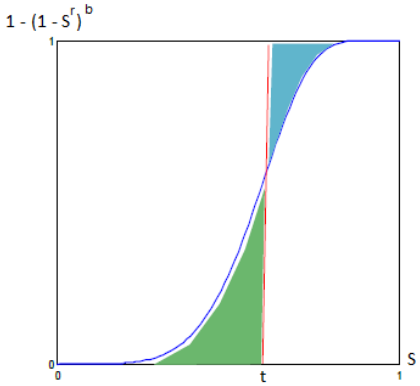

Fig. 8. The quantity of false positive and false negative

In the Fig. 8, the area of the green region equals the quantity of false positive and the area of blue region equals the quantity of false negative. Having that said, we need to compute the integral of this function. The way to compute the integral for this function is as follows:

$$\int 1 - (1 - s^r)^b \, ds$$

To compute this integral, we expand the expression $(1 - s^r)^b$ using the Newton binomial. The Newton binomial expansion is as follows:

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$$

$$(1 - s^r)^b = \sum_{p=0}^{b} (-1)^p \binom{b}{p} 1^{b-p} s^{rp}$$

As a result, the integral can be written as follows:

$$\int 1 - (1 - s^r)^b \, ds = \int ds - \int \sum_{p=0}^{b} (-1)^p \binom{b}{p} s^{rp} \, ds = s - \sum_{p=0}^{b} (-1)^p \binom{b}{p} \int s^{rp} \, ds$$

Therefore, the results of this integral computation will look like the relationship 2:

$$F(s) = s - \sum_{p=0}^{b} (-1)^p \binom{b}{p} \frac{1}{rp+1} s^{rp+1} \quad (2)$$

Now in order to calculate the quantity of false positive, one should calculate the integral of this function from the point zero to t and then extract a relationship for this purpose. This relationship will be calculated like this:

$$\int_0^t 1 - (1 - s^r)^b \, ds = F(t) - F(0)$$

As it was mentioned, the value of t will approximately equal $(1/b)^{1/r}$, so the quantity of false positive (FP) will be calculated using the relationship 3:

$$FP = F((1/b)^{1/r}) \quad (3)$$

In order to calculate the quantity of false negative, one can compute the integral of the function from the point t to 1 and then reduce this amount from the area of the rectangle which has been created in the right side of the Fig. 8 so that the area of the surface above the graph may be calculated. It's possible to compute the integral of the function from point t to 1 as follows:

$$\int_{t}^{1} 1-(1-s^{r})^{b}\,ds = F(1)-F(t)$$

The area of the rectangle which has been created at the right side of the graph for this function equals:

$$1 \times (1-(1/b)^{1/r})$$

Therefore, the value of false negative (FN) will be calculated using the relationship 4:

$$\mathrm{FN} = 1-(1/b)^{1/r} - F(1) + F((1/b)^{1/r}) \quad (4)$$

4.1. Practical results

With the relationships which we produced in the previous part, we can calculate the quantity of false positive and false negative for different *r* and *b* values. Due to the complexity of the relationships, a program was compiled using the VB.Net language to calculate the quantities of false positive and false negative. Using this program, the quantity of false positive and false negative was calculated for several different *n*, *r* and *b* values and the results can be seen in the Table 1.

Table 1. The false positive and false negative quantities for several different n values

Table 1(b). n = 20

r	b	t	FP	FN
4	5	0.6687	0.1078	0.0275
5	4	0.7579	0.1033	0.0218
2	10	0.3162	0.0817	0.0357
10	2	0.933	0.0737	0.0065

Table 1(a). n = 10

r	b	T	FP	FN
2	5	0.4472	0.118	0.0402
5	2	0.8706	0.1253	0.0123
1	10	0.1	0.0376	0.0285
10	1	1	0.0909	0

Table 1(d). n = 100

r	b	t	FP	FN
50	2	0.9862	0.0169	0.0013
10	10	0.7943	0.0579	0.0152
20	5	0.9227	0.0361	0.0071
25	4	0.9461	0.0302	0.0052

Table 1(c). n = 50

r	b	T	FP	FN
2	25	0.2	0.051	0.0257
25	2	0.9727	0.0326	0.0027
5	10	0.631	0.0835	0.0251
10	5	0.8513	0.0632	0.0133

In the Table 1, the quantities of false positive, false negative and threshold limit for n = 10, 20, 50, 100 with different samples of *r* and *b* were calculated. Generally, if evading the false negative is important, *r* and *b* should be selected in such a way that they produce a threshold limit smaller than *t*. If speed is important and we want to limit the false positive, we should select *b* and *r* in such a way that a greater threshold limit is produced. However, as it's clear through the results, this rule is not always dominant and has some exceptions.

5. Conclusion

In the present article, the methods of finding similar documents from among the massive dataset were investigated. The first method which was reviewed was the shingling method and it was shown that how it is possible to turn the issue of similarity to an issue based on sets. Then the minhashing method was mentioned the most important objective of which is to reduce the volume of information in such a way that after reducing the volume of information, estimating the similarity of the documents might be possible. After that, the LSH method was investigated and its main purpose is only to review the pairs which are similar to each other with higher probabilities. After naming these methods, a method for calculating the false positive and false negative was presented and relationships were extracted for putting into effect this idea. Ultimately, this method was reviewed using different parameters.

Aside from the fact that one can use the *r* and *b* parameters in order to reduce the false positive and false negative quantities, AND and OR structures can be similarly used to decrease the quantities of false positive and false negative. These two constructions can be applied on minhash functions and amplify them. It means that each minhash function can be replaced with a number of other functions in the mold of one of these two structures. The present project can be an appropriate research basis in this regard.

REFERENCES

1. Mitra, S., Pal, S. K. and Mitra, P, Data mining in soft computing framework: A survey, IEEE transactions on neural networks, 2002 13(1), 3-14.

2. Manber, U, Finding similar files in a large file system, In Proceedings of the USENIX winter 1994 technical conference (Vol. 1), 1994.

3. Brin, S, Near neighbor search in large metric spaces, Department of Computer Science Stanford University, 1995.

4. Bank, J. and Cole, B, Calculating the jaccard similarity coefficient with map reduce for entity pairs in Wikipedia, Wikipedia Similarity Team, 2008.

5. Rajaraman, A. and Ullman, J. D, Mining of massive datasets, Cambridge University Press, 2011.

6. Broder, A. Z, Identifying and filtering near-duplicate documents, In Combinatorial Pattern Matching, 2000, Springer Berlin Heidelberg, p. 1-10.

7. Broder, A. Z, On the resemblance and containment of documents, In Compression and Complexity of Sequences 1997, p. 21-29.

8. Broder, A. Z., Charikar, M., Frieze, A. M. and Mitzenmacher, Min-wise independent permutations, Journal of Computer and System Sciences, 2000, 60(3), 630-659.

9. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., ... and Yang, C, Finding interesting associations without support pruning, Knowledge and Data Engineering, IEEE Transactions, 2001, 13(1), 64-78.

10. Andoni, A. and Indyk, P, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, In Foundations of Computer Science, 2006, FOCS'06, 47th Annual IEEE Symposium, p. 459-468.

11. Gionis, A., Indyk, P. and Motwani, R, Similarity search in high dimensions via hashing, In Proceedings of the international conference on very large data bases, 1999, p. 518-529.