www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 11 November, 2013 Page No. 3097-3100

Identification of level of resemblance between web based documents

Surbhi Kakar[1]

Assistant Professor, GGSIPU

Abstract—One of the biggest challenges today on web is to deal with the "Big data" problem. Finding documents which are near duplicates of each other is another challenge which is in turn brought up by Big data. In this paper the author focuses on finding out the near duplicate documents using a technique called shingling. This paper also presents the different types of shingling that can be used. Further, a measure called the Jaccard coefficient is discussed which can be used to judge the degree of similarity between the documents.

***Index Terms*—Big data, shingling, Jaccard Coefficent**

INTRODUCTION

"Big Data" as the name suggests refers to a large amount of data. Every industry today is enjoying the benefits as well as dealing with the challenges brought up by the ever increasing data of their own companies. Data mining is a process where in it is possible to mine huge amounts of data in order to expose hidden patterns. For instance, with the help of data mining techniques today, it is possible for a business organization to get insights into what their customer wants or how he will react in future. Therefore, mining this Big Data can help discover user's hidden behavioral patterns and their intentions too. [1] The major challenges with "big data" are the increasing size, heterogeneity and velocity of data being added to the space every moment [2].

The increasing size of data is resulting in storage overheads. A serious issue is what to do with this data. What data should to be discarded and what should be stored. The heterogeneity of data on the other hand refers to the data coming from several sources which might be unstructured or even incomplete. For instance, data can be in various formats ranging from simple records to geo spatial data.

The velocity of data refers to the rate at which data is continuously being added on the web. Such data is becoming difficult to analyze and interpret which is also taking a large processing time.

Duplicate or near duplicate documents are the documents which might be exact replicas of each other or might be similar to one another. Finding out the near duplicates of web documents in a scenario like today where the web is facing the challenges of Big data becomes a serious issue to handle. This is mainly because the users today are just not interested in responses that are mere duplicates of each other. Also, indexing such documents affects the storage and processing time of the search engines. The resemblance between documents can vary between 0 to 1. A "1" indicates that the two documents compared for similarity are almost the same. Whereas a "0" indicates a higher level of dissimilarity between the two documents [4]

Shingling is a technique which can be used to find out the near duplicate documents. It is based on creating contiguous subsequences of a document of length q. These

shingles can be created by either using subsequences of characters or words.

Jaccard Coefficient on the other hand, is a measure which determines the similarity between two documents that are represented as sets. This measure ranges from 0 to 1.

This paper defines the problem definition in the first section which is to judge the level of resemblance between two documents.

The second section describes about shingling as a technique used to solve the problem. The section thereby discusses about the Jaccard coefficient measure used to judge the similarity between documents.

1. PROBLEM DEFINITION

One of the problems associated with "Big Data" is finding duplicate or near duplicate documents as discussed previously. As per [3] it has been estimated that as many as 40% of web pages are duplicates of each other. The duplicity here refers to documents which are textually similar and not semantically

similar. Some of the common examples of duplicate/near duplicate web documents are mirror sites, news articles from a common source, plagiarized documents. Mirror sites are sites which are duplicated at a number of hosts to facilitate load sharing. Plagiarized documents are the ones that may not be exact copies of one another but it is a possibility that they share large amounts of text which may not be in the same order.

The documents which are exact replicas of each other are still easier to find as compared to documents that are similar. Finding such documents require the use of hash functions [8]. The items which are duplicates will have the same hash value as compared to the dissimilar items. Therefore detecting near duplicate pages is very important because it affects the search engines time and space complexity when it has to index and store duplicate web documents.

2. SHINGLING: A SOLUTION

Shingling as discussed above is a technique of creating consecutive subsequences of tokens of a document. The tokens here can refer to characters, words or lines. A document can be represented as a set of string of characters. Given a document d, we can create 't' tokens of d which are contiguous sequences of the text of d. Creating a q-shingle/q-gram means creating a set of substrings of d which are of length q[4][5]

Shingling can be done by creating tokens of characters as well as words.

A. Shingling by characters

Let $S(d_1)$ be a set of document d_1 which is defined as "acdacef". Then q-shingles based on letters for the above document where q=2 will be {ac, cd, da, ce, ef}.

The elements appearing multiple times in the document are only considered once while creating a shingle set.

B. Shingling by words

Shingling by words is done by using words as substrings of a document. If $S(d_1)$ contains a string:

"I did my work today"

Then q-shingles based on words where q=3 will be:

" I did my", "did my work" "my work today"

In such a case, space used to store these shingles would be O(q*w), where q is the length of the shingle and w are the number of words in the document

C. Size of Shingles

While creating the shingle sets, the point to be focused upon is what should be the size of q so that the probability for the similarity between the similar documents is maximized as compared to dissimilar documents. If q is kept too small, there is a probability that most of the text in one document also appears in the other document in spite of them being dissimilar textually. As per [6], the size of q should be picked large enough so that the probability of one shingle set appearing in the other document is low. For comparing large documents like research papers the size of q can be taken around 9 whereas for documents like comparing emails for similarity, the size q can be picked as 5.

3. JACCARD COEFFICIENT: A MEASURE FOR RESEMBLANCE BETWEEN DOCUMENTS

Jaccard coefficient is used to measure resemblance between two documents. For two documents represented as sets A and B, the Jaccard coefficient is the ratio of intersection of two sets to the union of them[5][7]. It can be calculated as:

$$JC(A,B)=|A \cap B| / |A \cup B|$$

Where JC(A,B) denotes the Jaccard Coefficient between two sets A and B. Jaccard coefficients are therefore in a range of [0,1].

For instance if A={3,4,5,6} and B={5,6,7,8} then:

JC(A,B)=2/4

This measure is used to judge the similarity between only "textually" similar documents and not "semantically" similar document. Two documents with a high Jaccard coefficient denote a higher degree of similarity between them. But at the same time keeping the size of shingle too small may bring about a higher Jaccard coefficient which may not imply that the two documents are similar.

5. IMPLEMENTATION

```
compare(n)
{
  for i=0 to n
    for j=0 to n
      Set S1=getShingle(readFile(d1, q)
                        Set S2=getShingle(readFile(d2,q)
      findJaccardSimilarity(s1,s2)
}
//finds Jaccard Similarity between two sets
 findJaccardSimilarity(S1,S2)
   return (S1 intersection S2) /(S1 union S2)
```

Here n is the no of documents to be compared for similarity. getShingle method first reads the document d_1 and d_2 amongst the list of documents being iterated and then makes shingles by characters and words of length q. It does it by finding consecutive substrings of length q for each document. It then finds the Jaccard similarity between each set being iterated. While making the shingle sets, the blank spaces can be split up to form one blank space.

This algorithm makes $O(n^2)$ comparisons to compare each document against the other. For instance, if there are 10 documents, each document will be compared with all the other documents resulting in overall 45 comparisons. The space

complexity for the same would be O(q*w) as discussed previously.

6. EXPERIMENT

Shingling as a technique was used by the author on a dataset of 10 text files. The length of shingle was taken as 3, 4 and 5. A java program was written to implement the same. Out of the 10 text files the Jaccard coefficients calculated for the first four files are summarized in Table1, 2 and 3.

TABLE1

q=3	
Similarity between i: 2 j: 1	0.11809045226130653
Similarity between i: 3 j: 1	0.023178807947019868
Similarity between i: 3 j: 2	0.03070175438596491
Similarity between i: 4 j: 1	0.01559792027729636
Similarity between i: 4 j: 2	0.15834767641996558
Similarity between i: 4 j: 3	0.022974607013301087

TABLE 2

q=4	
Similarity between i: 2 j: 1	0.09535452322738386
Similarity between i: 3 j: 1	0.01461038961038961
Similarity between i: 3 j: 2	0.018571428571428572
Similarity between i: 4 j: 1	0.003372681281618887
Similarity between i: 4 j: 2	0.13079470198675497
Similarity between i: 4 j: 3	0.009389671361502348

TABLE 3

q=5	
Similarity between i: 2 j: 1	0.07971014492753623
Similarity between i: 3 j: 1	0.00967741935483871
Similarity between i: 3 j: 2	0.011315417256011316
Similarity between i: 4 j: 1	0.0016806722689075631
Similarity between i: 4 j: 2	0.11201298701298701
Similarity between i: 4 j: 3	0.003484320557491289

Note: Here i and j refer to the documents that are being compared for similarity.

The statistical analysis of the above result can be stated as below:

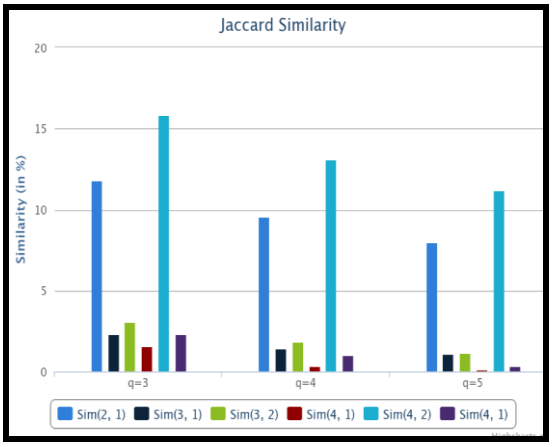

Fig. 1. Jaccard coefficients for q=3, 4 and 5

On experimenting the dataset with the algorithm, it was found that on varying the values of q the Jaccard coefficient also varied. For smaller values of q, Jaccard coefficient came out to be larger which indicated a higher degree of similarity between the documents. Whereas as the value of q was increased the Jaccard coefficient decreased.

7. CONCLUSIONS

The paper discussed about the problem of "Big Data" emerging at a speedy rate. It also focused on problems like near duplicate detection of documents for which it described a technique called shingling. In the later sections the author brought about two approaches by which shingling can be done.

As per the experiment conducted by the author, it was found that for smaller values of the length of the shingle (q) the jaccard coefficient between the documents was larger as compared to when the value of q was increased.

$$q \alpha 1/JC(A,B)$$

This implied that a higher Jaccard coefficient does not necessarily indicate that the documents are similar. It can vary depending upon what value of q is chosen. Therefore, the value of q should be appropriately chosen in order to have an accurate estimate of degree of similarity between the documents.

8. LIMITATIONS

For very large values of n where n is the number of documents to be compared for similarity, shingling alone as a technique used can result in large processing and storage overheads since it takes $O(n^2)$ comparisons to compare each document. As a solution shingling can be clubbed with Minhashing or Locality sensitive hashing(LSH) to improve the results. Locality sensitive hashing does not compare all the documents for similarity. It only focuses on the candidate pairs that are likely to be similar for comparison. This reduces the comparisons to be done while checking documents for similarity. Also LSH is a better technique while searching documents in high dimensional spaces.

9. FUTURE SCOPE

The author will extend the work discussed in this paper and implement Minhashing and Locality sensitive hashing techniques to improve the results in order for efficient near duplicate detection of web pages.

REFERENCES

[1] Michael, Katina, and Keith W. Miller. "Big Data: New Opportunities and New Challenges." *Editorial: IEEE Computer* 46.6, pp.23, 2013

[2] Einav, Liran, and Jonathan D. Levin. "*The Data Revolution and Economic Analysis"*.No. w19035. National Bureau of Economic Research, pp.3-5, 2013.

[3] Manning D. Christopher , Raghavan Prabhakar , Schütze Hinrich , "Introduction to Information Retrieval", *Cambridge University Press*, April 2009

[4] Broder, Andrei Z. "Identifying and filtering near-duplicate documents." In *Combinatorial Pattern Matching*, Springer Berlin Heidelberg, pp. 2-6, 2000.

[5] Broder, Andrei Z. "On the resemblance and containment of documents." In Proceedings of *Compression and Complexity of Sequences*, IEEE, pp. 3-6, 1997.

[6] Rajaraman, Anand, and Jeffrey David Ullman, "Mining of massive datasets" *Cambridge University Press*, pp. 6, 2012.

[7] Andoni, Alexandr, and Piotr Indyk. "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions." In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, pp.120, 2006.

[8] Ye, Shaozhi, Ji-Rong Wen, and Wei-Ying Ma. "A systematic study on parameter correlations in large-scale duplicate document detection." *Knowledge and Information Systems* 14.2, pp. 2-6, 2008