Detection of/between similarity of documents with hashing

Roger Vilaseca Darné, Xavier Lacasa Curto and Xavier Martín Ballesteros
Algorithms

1st December 2018

1 Introduction

2 Jaccard Index

The Jaccard Index, also known as Intersection Over Union (IOU), calculates the percentage of similarity between two sets.

For any pair of sets S and T, the Jaccard Index is defined as:

$$J(S,T) = \frac{|S \cap T|}{|S \cup T|} \tag{1}$$

We can easily deduce that the more common words, the bigger the Jaccard Index, which means that it is more probable that one set is a duplicate of the other.

Example 2.1. *In Figure 2.1 we see two sets S and T. There are 3 elements in their intersection ("I", "love", "chocolate") and 6 in their union ("I", "love", "chocolate", "and", "pizza", "white"). Thus, J(S, T) = 3/6.*

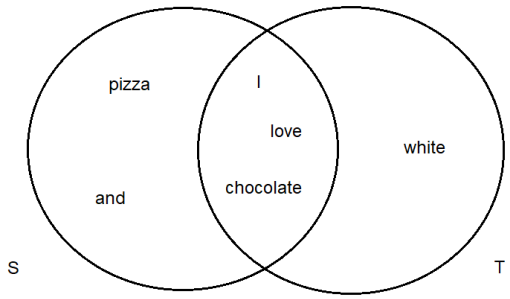

Figure 2.1: Two sets with Jaccard Index 3/6.

3 Shingling of Documents

Any pair of documents can be compared by watching the number of repeated strings they have. The more common strings, the more probable is that one is a duplicate from the other.

One way to represent a document as a set is to insert in the set each string that appears in it. If we do so, then duplicated documents that have reorganized the sentences or even the entire text will have plenty of common strings, and will be detected as duplicated.

3.1 k-Shingles

The idea is not to insert in the set all the words, but a set of characters of size k. Thus, each element of the set will have the same size as the others.

The question now is how big k should be? If we take a small value of k, this will result in many shingles that are present in all documents. Suppose we choose the extreme case ($k = 1$). Then, all documents would result to be similar, as the most used characters are present in all documents. However, if we take a big value of k, then any pair of documents would not share a shingle.

The value of k depends on the size of the documents. A poem will not have the same k value than an article. Otherwise, we could have the problems mentioned before. According to Anand Rajaraman, Jure Leskovec, and Jeffrey D. Ullman (2011), "k should be picked large enough that the probability of any given shingle appearing in any given document is low." (p. 78).

4 Compairing Similarity Using The Sets

If we succeed in shingling the documents by using the k-Shingling technique, we will only have to compare all pairs of documents using the Jaccard Index and say if there is similarity between them or not. In order to do this, we have to store all the information in a data structure, for instance (e.g.), a matrix. By doing this, we have two problems: time and space complexity. <= **Correcte que vagi aquí la úlima frase?**

4.1 Matrix Representation

To represent the matrix, we will put the documents' sets in the columns and the union of all the documents' sets in the rows. The values of the matrix will be the following:

$$\begin{cases} 1, & \text{if column } c \text{ contains row } r \\ 0, & \text{otherwise} \end{cases}$$

¿For any pair of row r and column c, if the set in the colum c has the element in the row r, the matrix will have a 1 in the cell (r, c). Otherwise, the cell will have a 0.?

Example 4.1. *For this example we will use two sets representing the words "Nadal" and "Nadia". Let $k = 2$ to form the k-Shingles.*

Element	S 1	S 2
na	1	1
ad	1	1
da	1	0
al	1	0
di	0	1
ia	0	1

Figure 4.1: Representation of the matrix with two sets S and T.

In this type of matrices, for any pair of columns we can have 4 types of results, which are the permutations of 0s and 1s of size 2:

Element	S 1	S 2
a	0	0
b	0	1
c	1	0
d	1	1

Figure 4.1: Representation of the matrix with all the possible permutations.

Note that as the matrix is sparse, most of the rows will be of type *a*. If we try to calculate the similarity between two sets S_1 and S_2 using the matrix and the Jaccard Index, we will have the following result:

$$J(S_1, S_2) = \frac{Q(d)}{Q(b) + Q(c) + Q(d)} \tag{2}$$

Where Q(x) is the number of rows of type *x*. Q(d) is the intersection of the sets and Q(b) + Q(c) + Q(d) is the union of the sets.

4.1.1 Time Complexity

Imagine we have n documents. Then, we have to compare each document with all the rest. Thus, the number of comparisons we have to do is $n * (n-1)/2$ which is equal to $O(n^2)$ ¿$omega(n * log(n))$? O potser és Theta de n^2?.

Example 4.2. *Suppose we have 1 million documents. The number of comparisons would be* $5 * 10^{11}$ *which is a huge number.*

$$\frac{(1 * 10^6) * 999.999}{2} = 499.999, 5 * 10^6 \approx 5 * 10^{11} \tag{3}$$

4.1.2 Space Complexity

In typical applications the matrix is sparse, which means that there are more 0s than 1s. ¿We can demonstrate this by calculating the probability of an element of the set to belong to a document D.?

If we take k shingles, then the document have relatively few of the possible shingles. Another way to think about this is with the toys in Christmas Day. Kids would be the columns of the matrix and toys, the rows. Usually, kids would like to have a specific toy, which is very popular at that moment. Then, lots of toys would not be buyed for any kid.

4.2 Minhashing

The main goal using minhashing is to reduce a lot the space complexity. We can achieve this by subsituting the matrix shown before by another matrix called "signature matrix".

Signatures are smaller representations of the sets, but they still preserve the similarity of the sets they represent. We will demonstrate this in the next section.

To minhash a set, first pick a random permutation of the rows. Then, the minhash value is the value of the first row that has a 1, preserving the permuted order.

Example 4.3. *In this example we will reuse the two sets of Example 4.1. Suppose that the random permutation has given the following order: "di", "da", "ia", "na", "ad", "al". Let h be the minhash function.*

Element	S_1	S_2
di	0	1
da	1	0
ia	0	1
na	1	1
ad	1	1
al	1	0

Figure 4.3: Permutation of the matrix of Example 4.1.

In the first column, we can see that $h(S_1) =$ "*da*" *and in the second one we see that* $h(S_2) =$ "*di*".

With this technique, we can see that every time we do a permutation, we only occupy one new row of the "signature matrix", which is reducing a lot the space.

4.2.1 Preserving the Jaccard Index

As we mentioned before, the Jaccard Index in the matrix is equal to the number of rows of type d divided by the number of rows of type $b + c + d$. And that index is preserved in the "signature matrix".

Proof. Look down through the permuted columns C_1 and C_2 until we see a 1 in any of the two columns. Then, we can have a row of type b or c, where only one of the columns have a 1, or a row of type d, where both columns have a 1 in it. If we find a row of type d, then the minhash function will take the same row. Thus, $h(C_1) = h(C_2)$. Otherwise, we must have a row of type b or c and $h(C_1) \neq h(C_2)$.

We can see that this is exactly the Jaccard Index. The probability of two columns have the same minhash value is equal to the number of rows of type d divided by the number of rows of type $b + c + d$.

$$P[h(C_1) = h(C_2)] = J(C_1, C_2) \tag{4}$$

□

4.2.2 Optimizing the Time for Permutations

Encara falta posar cosetes NENG!

for each row r **do**
  for each has function h_i **do**
    compute $h_i(r)$;
  end for
end for
for each column c **do**
  if c has 1 in row r **then**
    for each has function h_i **do**
      if $h_i(r)$ is smaller than $M(i, c)$ **then**
        $M(i, c) := h_I(r)$;
      end if
    end for
  end if
end for

5 Locality Sensitive Hashing (LSH)

Mas texto.

5.1 Referencies

https://towardsdatascience.com/understanding-locality-sensitive-hashing-49f6d1f6134
https://santhoshhari.github.io/Locality-Sensitive-Hashing/
https://www.youtube.com/watch?v=96WOGPUgMfw
https://www.youtube.com/watch?v=_1D35bN95Go
https://medium.com/engineering-brainly/locality-sensitive-hashing-explained-304e
http://www.mit.edu/~andoni/LSH/
http://infolab.stanford.edu/~ullman/mmds/ch3.pdf
https://aerodatablog.wordpress.com/2017/11/29/locality-sensitive-hashing-lsh/

References

[1] Author, *Title*, Editor, (year)