Data mining assignments 1 summary

Benedith Mulongo and Georgios Leventis

October 2019

1 Finding similar items : the algorithm

Sometimes we need to find documents that are near duplicates of each other. We can can decompose each documents in terms of their words and hash each words in order to compare them as a set. Although this comparison method is completely sound , it is computationally expensive. MinHash is a algorithm that can be use to approximate the Jaccard similarity.

1.1 k-shingles

K-shingles is decomposition of a string in k different sub-string. We need to implement k-shingles for finding different shingles of size k for a given text document on the character level. Example: Suppose our document D is the string abcdabd, and we pick k = 2. Then the set of 2-shingles for D is $[ab, bc, cd, da, bd]$

```
document = "abcdabd"
k = 2
_, shingles = k_shingles(document, k)
print(shingles)
```

```
OrderedSet(['ab', 'bc', 'cd', 'da', 'bd'])
```

Figure 1: Example of k shingles

1.2 Shingles representation and hashing

We nee to hash each shingle element to 4 bytes in order to reduce space and computation time. That can be done by using a hash function that maps each shingle to a bucket in the range $[0, 2^{32} - 1]$. example : hash(['ab', 'bc', 'cd', 'da', 'bd']) $\rightarrow$ [195, 197, 199, 198]

```
document = "abcdabd"
k = 2
shingles_encode, shingles = k_shingles(document, k)
print(list(shingles))
print(list(shingles_encode))
```

```
['ab', 'bc', 'cd', 'da', 'bd']
[195, 197, 199, 198]
```

Figure 2: Example of hashing shingles set

1.3 Shingles comparison : jaccard similarity

In order to compare two vector sets, we can use the Jaccard similarity measure, defined as follows : $Jaccard(A, B) = \frac{A \cap B}{A \cup B}$

```
index1 = 0
index2 = 3
comparison = compareByJaccard(documents,k,index1, index2)
print("Comparison :")
print(documents[index1])
print(documents[index2])
print("Similarity : ", comparison)
```

```
Comparison :
ad
acd
Similarity :  0.6666666666666666
```

Figure 3: Example of Jaccard comparison of two strings

1.4 MinHash algorihm

let's have four documents :

- s1 = 'ad'
- s2 = 'c'
- s3 = 'bde'
- s4 = 'acd'

The union of all the characters for k = 1, is [a,b,c,d,e] (no duplicates). That is the **universal set** is [a,b,c,d,e]. We can now represent each document

s_1, s_2, s_3, s_4 in term of the universal set. Let's take document s1 as an example, its representation is s1 = [1,0,0,1,0]. s1 has 1 in the first and fourth element because it shares shingle "a" and "d" with the universal set. We can do the same with other documents.

```
s1 = 'ad'
s2 = 'c'
s3 = 'bde'
s4 = 'acd'
k = 1
documents = [s1,s2,s3,s4]
_, universalSet = universal_set(documents, k)
print(list(universalSet))
```

```
['a', 'b', 'c', 'd', 'e']
```

Figure 4: Example of universal set

The characteristic matrix is the matrix representation of each documents in terms of the universal set. We have found that the universal set is a,b,c,d,e. s1 = 1,0,0,1,0 in term of the universal set and s2 = 0,0,1,0,0 etc. The characteristic matrix is [s1, s2, s3, s4], where each s_i is the transpose of the Boolean vector representation of each document.

```
s1 = 'ad'
s2 = 'c'
s3 = 'bde'
s4 = 'acd'
k = 1
documents = [s1,s2,s3,s4]
characteristic_mat = matrix_set(documents, k)
print("The characteristic matrix : ")
print(characteristic_mat)
```

```
The characteristic matrix :
[[1. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 1. 0. 1.]
 [1. 0. 1. 1.]
 [0. 0. 1. 0.]]
```

Figure 5: Example of characteristic matrix

Given that we have the characteristic matrix we need to find different permutations of the row elements of the matrix. it is computational expensive to $(N!)$ for N rows. Therefore we can generate instead N hash functions and use them to create an illusion of rows permutations. The N hash functions need to be of the forms :

$H(row_i) = ((a * row_i + b) \mod c) \mod M$

Where a and b are two different no repeating integer in the range of $[0, 2^{32} - 1]$. M is the number of row in the characteristic matrix, c is a prime number such as $c > M$.

The minHash signatures is the compressed version of the matrix set in order to speed up the time needed to compare two documents. MinHash signatures

is a matrix where the number of rows is equal to the number of hash functions and the number of columns is equal to the number of documents. The python implementation shows how it can be implemented. Look up at the book Mining of massive datasets for further details

The Jaccard similarity used to compare signatures is just the fraction between the number of times signature A and B have the same element by the total number of element in signature A or B (length(A) = length(B)).

```
def MinHashing(documents,k,Snumb = 100):
    #MinHash implementation
    characteristic_mat = matrix_set(documents, k)
    M, nDocs = np.shape(characteristic_mat)
    hashes = generateMinHashFunctions(M, Snumb)
    signature = np.ones((Snumb, nDocs)) * np.inf

    for row_r in range(M) :
        for col_c in range(nDocs) :
            if characteristic_mat[row_r,col_c] == 1 :
                for i, h_i in enumerate(hashes) :
                    signature[i,col_c]  = min(h_i[row_r],
    signature[i,col_c])


    return signature

def jaccard_sim_minhashing(signature, index1, index2) :
    signature = np.array(signature)
    A = signature[:,index1].tolist()
    B = signature[:,index2].tolist()

    total = len(A)
    commun = 0
    for i in range(len(A)) :
        if A[i] == B[i] :
            commun += 1

    return (commun/total)
```

Listing 1: MinHash

1.5 Finding similar documents with minHash

In order to find similar document in a set of documents, we need to decide on a similarity threshold to be used say $s = 0.6$, then all the documents have similarity $s \geq 0.6$ are considered as similar. In python, we get :

```
s1 = 'ad'
s2 = 'c'
s3 = 'bde'
s4 = 'acd'
k = 1
s = 0.8
documents = [s1,s2,s3,s4]
similar_document(documents, k, s)
```

```
similar docs :  [[0, 3]]
```

Figure 6: MinHash example

2 Locality Sensitive Hashing

3 Data

We have used the Quora dataset `https://www.kaggle.com/c/quora-question-pairs/data` for questions duplicates and we will use minHash for finding simalar questions on the character level (nothing about semantic similarity, linguistic similarity). The data is slightly modified for our purpose, we have filtered out everything in order to quip only text questions.

```
number = 5
data = read_data()
documents = data[0:number]
print(documents)
```

```
['What is the step by step guide to invest in share market in india?', 'What is the step by step guide to invest
in share market?', 'What is the story of Kohinoor (Koh-i-Noor) Diamond?', 'What would happen if the Indian gover
nment stole the Kohinoor (Koh-i-Noor) diamond back?', 'How can I increase the speed of my internet connection wh
ile using a VPN?']
```

Figure 7: Quora data snippet

4 Results

4.1 Data size vs execution time

The plot shows that there is almost a linear relation between the problem size and the execution time. So we have achieve some scalability.

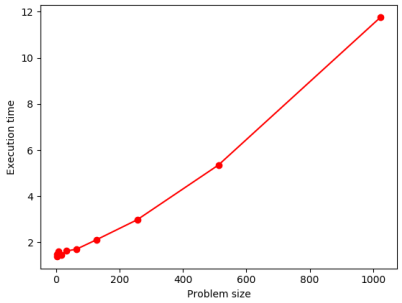

Figure 8: Plot of data size vs execution time

4.2 Shingle size vs relative accuracy

Given the data we are using there is no way of calculating the accucary in the proper sense, however we are comparing the true similarity with jaccard ($true_{sim}$) and the approximation with minHash ($approx_{sim}$). The similarity or accuracy is defined as follows :

$acc = \frac{true_{sim} \in approx_{sim}}{max(true_{sim}, approx_{sim})}$

The graph shows that single size between 4 and 7 is quite good options. This kind of method can also be used for parameter tuning.

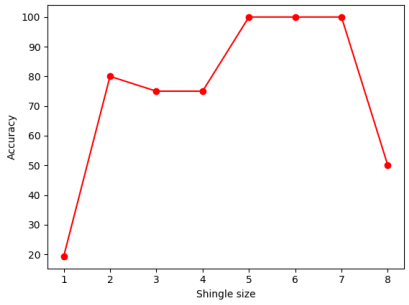

Figure 9: Plot shingle size vs relative accuracy

4.3 Number of hash functions vs relative accuracy

Here we want to test the effect of the number of hash functions to the performance of the minHash similarity approximation. The graph shows that an increasing number of hash functions increase the certainty and the performance of the minHash.

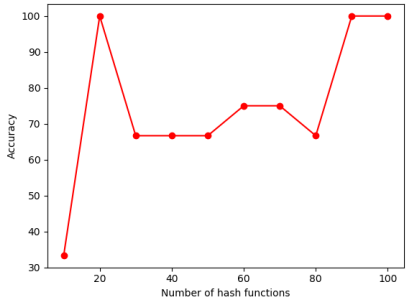

Figure 10: Plot number of hash functions vs relative accuracy

4.4 Test with real data

```
# Take 20 questions
number = 100
data = read_data()
documents = data[0:number]
#Parameters
k = 5
s = 0.8
# Find similar questions :
similar_documents(documents,k,s)
```

```
similar docs indices :  [[0, 1], [26, 27], [32, 33], [50, 51], [82, 83], [84, 85]]
Similar docs :  0
What is the step by step guide to invest in share market in india?
What is the step by step guide to invest in share market?
Similar docs :  1
What was your first sexual experience like?
What was your first sexual experience?
Similar docs :  2
What does manipulation mean?
What does manipulation means?
Similar docs :  3
What are some tips on making it through the job interview process at Medicines?
What are some tips on making it through the job interview process at Foundation Medicine?
Similar docs :  4
When can I expect my Cognizant confirmation mail?
When can I expect Cognizant confirmation mail?
Similar docs :  5
Can I make 50,000 a month by day trading?
Can I make 30,000 a month by day trading?
```

Figure 11: Test of minhash with the Quora dataset

References