INF 553(Spring 2016)

Assignment 3: Document Similarity using Shingling, Min-Hashing & LSH

In this assignment you will explore the use of Shingling, Jaccard Similarity, Min Hashing, and LSH in the context of document similarity.

Write your own code!

For this assignment to be an effective learning experience, you must write your own code! I emphasize this point because you will be able to find Python implementations of most or perhaps even all of the required functions on the web. Please do not look for or at any such code! **Also do not share code with other students in the class!!**

Submission Details

You have to find out similarity between text documents. You will be provided with a folder containing the text files among which you have to find out the similarity. All text files only have at most 27 characters: **all lower case letters and space**.

1. You need to turn in a python file named as: **<lastname>_<firstname>_DocSimilarity.py**
2. Your code will be executed using the following command.

python <lastname>_<firstname>_DocSimilarity.py <folder_path_containing_docs> <k> <type_of_shingles> <no_of_hashes_for_minhashing> <threshold_value>

The input parameters supplied are as follows:

- **folder_path_containing_docs**: The path to the folder containing the text files.
- **k**: Value of k used while making k-shingles.
- **type_of_shingles:** The type of shingles you have to consider. The two possible values are **char/word.**
  - If the input parameter is set as **char**, you have to construct **k-shingles** based on characters.
  - If the input parameter is set as **word**, you have to construct **k-shingles** based on words.
- **no_of_hashes_for_minhashing:** No of hash functions to be used for Min-hashing.
- **threshold_value:** The threshold value **s** used in LSH which defines how similar the documents have to be for them to be considered as similar pair.

Task 1: Shingling

You will construct **k-shingles** for all documents. Remember to store each shingle once, duplicates should be ignored. The value of **k** will be provided as an input parameter. Also the type of shingles you have to construct will be specified. It can be either **char** or **word.**

- For input parameters **k = 5** and **type_of_shingles = char**, the shingles for text **"How are you"** will be:

  ['How a', 'ow ar', 'w are', ' are ', 'are y', 're yo', 'e you']

- For input parameters **k = 2** and **type_of_shingles = word**, the shingles for text **"How are you"** will be:

  ['Howare', 'areyou']

Output:

You need to output the **No of shingles** for each document and the **Jaccard similarity** between each possible pair of documents. So if there are 4 different documents the number of possible pairs will be 6.

$$\text{JaccardSimilarity(D1, D2)} = \frac{\text{Shingles(D1)} \cap \text{Shingles(D2)}}{\text{Shingles(D1)} \cup \text{Shingles(D2)}}$$

Please refer to **sample_char_output.txt** to find the output format required.

Task 2: Min Hashing

Implement min-hashing to convert the shingles to short signatures. The number of hash functions to be used will be specified as an input parameter.

Hash function to be used: $\mathbf{h_\alpha(x)= \alpha x+1 \bmod}$ **<no_of_unique_shingles_in_the_corpus>.**

- Here **α = 1, 2, 3……(no_of_hashes_for_minhashing)**
- Here **x** represents the index of each k-shingle in an alphabetically sorted list having all shingles. Note that the index starts form 0.

So if the input parameter **no_of_hashes_for_minhashing = 3,** the hash functions to be used are:

- **x+1 mod <no_of_unique_shingles_in_the_corpus>**
- **2x+1 mod <no_of_unique_shingles_in_the_corpus>**
- **3x+1 mod <no_of_unique_shingles_in_the_corpus>**

Apply the Algorithm taught in the class (slides 21-24 from Finding Similar sets (part 2)) to obtain the min-hash signature.

Now using the min-hashes, compute the Jaccard Similarity between each possible pairs of documents using the below formula.

$$\text{Jaccard\_Similarity(A,B)} = \frac{1}{h} * \sum_{i=1}^{h} \begin{cases} 1 \; if \; A_i = B_i \\ 0 \; if \; A_i \neq B_i \end{cases}$$

;here h is the **<no_of_min-hashes>**, A and B are the min-hash signature for Documents D1 and D2 respectively.

Output:

You need to output the **Min-Hash signature** for each document and the **Jaccard similarity** (obtained by the above formula) between each possible pair of documents. So if there are 4 different documents the number of possible pairs will be 6.

Please refer to **sample_char_output.txt** to find the output format required.

Task 3: Locality Sensitive hashing

LSH is used to identify the candidate pairs which should actually be checked for similarity. We tune the values of **b (no of bands)** and **r (no of rows per band)** so that we have no false negatives and try to reduce the number of false positives.

- Here you will be given the value of threshold **s**, and you have to obtain the optimal value of **b** and **r**. As we know that **b*r = n,** where **n** is the length of minhash signature and the approximate value of threshold **s = (1/b)^(1/r)**, you can find optimal value of b and r.(Refer slide 23-25 from FindingSimilarItemsetsPart3).
- As you have the values for **b** and **r**, consider splitting the min-hash signature matrix (obtained in previous task) into **b bands** having **r rows** each.
- Now for each band, hash its portion of each column to a hash table with **m** buckets (Keep the value of **m** as large as possible).
- Note: As a hint to hash the columns of a band, you should think of making a unique combination from all rows in a band and make it as a key for the hash. You can then assign the corresponding Document_Name as the hash value for the key. Also make sure to maintain a different hash for each band.
- Now the candidate pairs of documents are those that hash to the same bucket for **>= 1** band.

Output:

- You need to output all possible set of candidate pairs which are obtained after performing LSH.
- Please refer to the **sample_char_output.txt** to find out the output format required.

You can execute your code using following commands and verify your outputs against the sample outputs:

- **python <lastname>_<firstname>_DocSimilarity.py Docs 5 char 120 0.35** and compare your output against **sample_char_output.txt**
- **python <lastname>_<firstname>_DocSimilarity.py Docs 2 word 30 0.35** and compare your output against **sample_word_output.txt**