CS 5683: Big Data Analytics

Project-2: Locality Sensitive Hashing

Total Points: 100

Due date: Sept 29, 2021 at 11:59pm

In this project, we study Locality Sensitive Hashing with an application of finding plagiarized documents. We will use the steps discussed in the class – Shingling, Min-Hashing, and LSH to identify similar (plagiarized) documents. Students will use this project to analyze several algorithmic choices (for example, choosing 'k'-shingles and 'L'-hash functions, etc.) and generate a brief report about their work with results.

Dataset Description:

The given data corpus was a part of the research study that created ground truth dataset for plagiarism detection algorithms[1]. The given data corpus is made of short answers (200-300 words) to 5 basic Computer Science questions. The corpus contains 100 answers/documents - 95 documents (file names prefixed with 'g') represent answers given by 19 students for all 5 questions, and 5 documents (file names prefixed with 'orig') collected directly from Wikipedia articles. The corpus has been designed to represent varying degree of plagiarism:

1. **Cut:** Answers were directly copied from Wikipedia
2. **Light revision:** Copied with little alterations on words and phrases
3. **Heavy revision:** Each sentence is rephrased heavily into sentences with similar meaning
4. **Non-plagiarism:** Participants read Wikipedia articles and write answers based on their understanding

We will use original datasets (original Wikipedia articles) to query nearly matching documents to find plagiarized student answers. The answers-to-plagiarism_degree mapping is given in the **corpus-final09.xls** for your reference.

To-Do:

Please use the **Project-2.ipynb** given along with this instruction to get a template for the final submission.

[1] Clough, P. and Stevenson, M., 2011. Developing a corpus of plagiarised short answers. *Language resources and evaluation*, *45*(1), pp.5-24.

Students are required to complete the following modules for this project:

1. **Shingling: (20 POINTS)**
    a. **Count k-shingles from the given corpus:** For k={3,4,5}, count the number of unique shingles you get from your sample data corpus. We use **word-based shingles** instead of character based. In other words, a sequence of k-words in a document is considered as a shingle. Report the count of k-shingles for your data sample
    b. Example: 2-shingles of "I love programming" is ["i love", "love programming"]. We simply need to convert upper cases to lower cases for text processing
    c. Create an index of shingles for **k=5**

2. **Min-Hashing: (40 POINTS)**
    a. Generate Hash functions: Generate ***L*** hash functions, where ***L* = {50, 100, 200, 500, 1000}** – meaning that students should experiment min-hashing with 5 different range of hash functions. Each hash function is simply a permutation of **5-shingle index** created in Step-1
    b. You can generate a hash function L_1 using $L_1 = (ax + b) \, mod \, N$, where **x** is shingle index, **a,b** are random numbers and **N** is the count of 4-shingles. Note that you have to generate random **a** and **b** for each individual hash function
    c. Generate the signature matrix: ***Generate all hash functions before this step.*** Use the algorithm discussed in the class to generate signature matrix based on ***L*** hash functions and 5-shingles obtained from each document. ***This step should not create the input matrix, as given in lecture slides – which will blow the memory***. Instead, the signature vectors should be created while reading each document using shingles index generated at Step-1
    d. Fact checks: Pick any one of the original Wikipedia articles and construct a signature vector (S_o) of it. Calculate Jaccard similarity $J_S(S_o,S_i)$, where S_i is the signature vector of document **i**. Report all documents that have the Jaccard similarity score **greater than *t* (*t=0.85*)**

3. **Candidate Generation (LSH): (30 POINTS)**
    a. Hash Signature Matrix: Split the signature matrix constructed from ***L*=1000** into **50 bands (*b*)** of **20 rows (*r*)**. Hash each document in each band into **199 buckets (*B*)**. You can hash a band of **r** values into one of the 199 buckets using the following formula:

$$h = \left(\sum_{i=1}^{r} r_i a^i \right) \, mod \, 199$$

    where a^i is the random number of a row in a band. a^i cannot be same for multiple rows in a single band. However, you can use same a^i in multiple bands. For example, $a^1(b_1) = a^1(b_2) = a^1(b_3) = \ldots$. But $a^1(b_1) \neq a^2(b_1) \neq a^3(b_1)$
    b. Fact checks: For each of the original Wikipedia articles, generate its signature vector, split it into 50 bands of 20 rows, and hash each band of the vector. Get all

documents that are hashed to same buckets as the original article. These are called **candidate documents.** Document ***d*** is a candidate if ***d*** and the original document hash to a same bucket at least once. Calculate Jaccard similarity $\mathbf{J_S(S_o,S_c)}$, where $\mathbf{S_c}$ is the signature vector of candidate document **c**. Report the number of false positives and the number of false negatives

i. **False positives –** candidate documents that have Jaccard similarity <= **t**
ii. **False negatives –** documents whose Jaccard similarity is actually > **t**, but are not in the candidate set
(Maintain t = 0.85 in all cases)

(Note: A document is plagiarized if it is a '**Cut**' or '**Light Revision**' of the original document)

HOW TO COMPLETE THIS PROJECT?

Students are ***not required*** to use ***PySpark*** for this project.

Grading Rubric:

1. Shingling – Total: 20 points
    a. Count total number of unique shingles in your data sample: 15 points
    b. Create 4-shingle index: 5 points
2. Min-Hashing: 40 points
    a. Hash function generation: 10 points
    b. Signature matrix generation: 15 points
    c. Fact checks: 5 points
    d. Report results of each ***L*** value: 10 points
3. LSH: (30 points)
    a. Hashing: 20 points
    b. Fact checks: 10 points
4. Program documentation, efficiency, and readability: 10 points

What to submit in Canvas?

Submit only your complete python notebook (.ipynb) file.

Simple reminder: You can discuss the logic and implementation details with your friends. But you should write your own program and documentation. *We will not grade your work if we suspect cheating.* Give references wherever necessary if you are using a logic from any internet sources. Start your work early as this project involve many sub-tasks which require some logical thinking.