Information Retrieval (CS F469)

Design Document

Assignment-2

Topic: Locality Sensitive Hashing

No. of Group Members: 1

Name: Navdeep Singh Narsinghia

ID No.: 2017B5A71675H

Problem Overview:

In this assignment, we are expected to code LSH algorithm on a dataset. We have to implement all the three sub-procedures for Shingling, Min-hashing and LSH step. We can use any similarity metric (Jaccard, Cosine or Euclidean) and its respective hashing technique to find Signature matrix.

Proposed solution:

We aim to build a program to detect similar documents and rank them in order of similarity using the algorithm of Locality Sensitive Hashing (LSH). Locality sensitive hashing (LSH) involves 3 major steps as described below:

1. Shingling – In the standard literature there is a concept of shingle size, k. A substring of length k is known as a shingle. All the documents are used to extract all possible shingles and using this, a shingle-document matrix is built. A shingle document matrix consists of 0's and 1's where 1 represents that a particular shingle is present in the document and 0 represents the absence of shingle in the document.

2. Minhashing – On the shingle-document matrix, minhashing is performed to build signature matrix using different hash functions. Minhashing is performed to compress the document vectors and give signatures that have less number of rows. Signature matrix is produced using multiple (random) hash functions.

3. Locality Sensitive Hashing – In this step, the signature matrix is divided into several bands and each band consists of a number of rows. Now, each of these bands is hashed onto a bucket so that similar documents end up in the same bucket. Now using a similarity metric (Jaccard, Cosine or Euclidean), the similarity between different documents can be calculated in a band.

After performing the above 3 steps of the LSH algorithm, we get the similarity between test documents and our query document.

Architecture

Dataset: Dataset from the website http://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html was used as corpus for testing my LSH algorithm program. The dataset consists of 95 documents containing short answers that were plagiarised from various sources. The rest 5 documents are original. One of the original document

("orig_taska.txt") was taken as query and similarity between this query document and the 95 training documents was calculated.

Programming Language: Python language was used for building this project.

Data Structure used:

1. Python lists have been used for storing names and contents of files, similarity percentages and other final results

2. Numpy module was used to create shingle document matrix and signature matrix.

3. NLTK module was used to calculate cosine similarity.

4. Python lists were used for creating and storing buckets.

Libraries Used:

Numpy, NLTK

Working:

Shingling: First, some processing of the text is done, for example changing the case of the text to lowercase, removing spaces, unnecessary words. Then using the contents from all the documents, different shingles are constructed. After this, a shingle-document matrix is built. In a shingle document matrix 1 represents that a particular shingle is present in the document and 0 represents the absence of shingle in the document.

Minhashing: In this step, the signature matrix is generated from the shingle-document matrix using multiple hash functions. In this project, we have used 50 hash functions which were randomly generated. For each shingle in the shingle document matrix, if a document has 1 in that row, then the corresponding column in signature matrix is filled with hash values of the indices only if they are less than the values already present. In this way number of rows is decreased from number of shingles to number of hash functions.

Locality Sensitive Hashing: In this step, the signature matrix generated in previous step is divided into bands and each band consists of a certain number of

rows. The relation between hash function, rows and bands is h=br where h is the number of hash function, b is the number of bands and r is the number of rows. Now, each of these bands is hashed onto a bucket so that similar documents end up in the same bucket. Documents with same signatures in a band end up in the same bucket. These documents are considered as candidate pairs and similarity between them has been calculated using Cosine Similarity.

Results

Shingle size, k = 6

Hash functions used = 50

No. of bands = 5

No. of rows = 10

Run Time: ~ 11.56 seconds

Output: The given output shows the most similar documents for the query "orig_taska.txt" along with similarity percentage of each training document with respect to query in decreasing order.

```
D:\Users\navdeep singh\Documents>python code.py
Welcome to the Locality Sensitive Hashing Program
Please wait.....Processing

Similarity report for  orig_taska.txt

DOCUMENT NAME                SIMILARITY
g0pE_taska.txt              98.85599944318659 %

g4pC_taska.txt              95.7295625424696 %

g0pD_taska.txt              84.7277697690971 %

g2pE_taska.txt              83.9139684876231 %

g4pE_taska.txt              74.63561736384106 %

g2pC_taska.txt              69.82803040419182 %

g0pB_taska.txt              69.6562557934116 %

g3pC_taskd.txt              68.33167673803882 %

g3pC_taska.txt              66.76641919946348 %

Time =  11.576477766036987 s
```