Birla Institute of Technology and Science-Pilani, Hyderabad Campus

First Semester 2020-2021

Information Retrieval (CS F469)

Design Document

Assignment 2

Locality Sensitive Hashing

Faishal Hussain Siddiqui	2019H1030012H
Rojan Sudev	2019H1030008H
Arpit Roy	2019H1030118H
Suraj Abhiman Shinde	2019H1030507H

Under the guidance of
Dr. Aruna Malapati

Abstract:

In this project, we created a plagiarism detector using Locality Sensitive Hashing. LSH groups similar items together. Items belonging to the same bucket have a high probability of being similar. The LSH algorithm contains the following three steps:

1. Creating shingles
2. Minhashing
3. Performing Locality Sensitive Hashing

k-shingle is a substring of length k. The document is used to extract shingles of length k. The shingles in the above step are used to create the incidence matrix, after that signature matrix is obtained by applying minhashing. Document vectors are compressed using minhashing to create signatures that have a lesser number of rows. The signature matrix is separated into bands in the LSH step, with some rows in each band. Every band is hashed into the buckets such similar documents have a high probability of ending into the same bucket. An appropriate similarity measure is used to find similarity between the documents.

Architecture:

Programming Language: python

Dataset: https://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html

The dataset of 100 documents, in which 95 documents are answers given by some participants to 5 questions.

Data Structures used are:

1. pandas.DataFrame for storing incidence matrix and signature matrix
2. python list for storing similar files to be returned to the user
3. python dictionary for buckets
4. python set for finding Jaccard similarity

Pre-processing:

The shingling module takes the text documents and performs pre-processing like case normalization, removal of \n, \r, and other unimportant characters. Then shingles from each document are created and added to the incidence matrix. The incidence matrix is a data frame with a shape of the number of shingles x no of documents. It has 1 when a shingle is present in a document and 0 if not at a respective position of the dataframe.

Minhashing technique:

Hashes[k] is the kth hash function. Each hash function will return the value as $(ax + b)\%c$ where a and b are greater than 0 and c is equal to the number of shingles.

The signature matrix of shape (no_hash_functions x no of documents) is initialized to infinity.

For each shingle (row) in the incidence matrix, if a document has 1 in that row then the corresponding column in the signature matrix is filled with hash values of the indices only if they are less than the values already present. In this way, the number of rows is decreased from the number of shingles to h (no of hash functions).

Locality Sensitive Hashing:

The signature matrix generated by minhashing is divided into b bands with r rows in each band. If m is the number of hash functions used for a signature matrix.

$$m = br$$

A hash function is used to hash each band. While hashing a particular band, parts of each document are hashed into a set of buckets. For different bands, a different set of buckets are used. Each set of buckets is a dictionary having hash values as the keys and a list of document id as values. So, documents with the exact same signature in a band end up in the same bucket of that band. These documents are considered as candidate pairs and similarity between them is calculated using an appropriate metric.

Results:

```
Reading corpus: corpus
100%|                                                  | 100/100 [04:55<00:00,
2.95s/it]
saving generated incidence index to file...
saved to corpus_inc_mat.pickle
(11640, 100)
shingling time: 295.44113516807556
100%|                                                  | 11640/11640 [02:45<00:00,
70.36it/s]
signature_matrix is being saved to pickle file.......
saved to sig_mat.pickle
minhashing time: 165.48939204216003
lsh time: 0.14264392852783203
Path of the test file: corpus/orig_taskd.txt
Threshold value: 0.2
Given file: corpus/orig_taskd.txt
corpus/g4pE_taskc.txt  0.22649698683970407
corpus/g0pC_taske.txt  0.206847938475928
corpus/g4pE_taske.txt  0.20220148611907196
corpus/g3pC_taskb.txt  0.16689540550892162
Precision: 0.75
Recall: 0.03571428571428571
```