ASSIGNMENT - 2 DESIGN DOCUMENT

On

LSH Based Retrieval System

By

Triyasha Ghosh Dastidar - 2017B2A70829H

Chatrik Singh Mangat - 2017B5A70822H

Digvijay Singh - 2017AAPS0317H

Shreyam Kumar - 2017AAPS0346H

Under the supervision of

PROF. ARUNA MALAPATI

Submitted in partial fulfilment of the requirements of

CS F469: INFORMATION RETRIEVAL

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(1st Semester 2020-21)

INTRODUCTION

We have built a LSH-based Retrieval system that takes in DNA sequences as queries and returns the DNA sequences with similarity above a certain threshold. The LSH based storage of DNA data allows us to resolve queries in very less time compared to Brute Force search.

PREPROCESSING

DNA sequences often have the structure ‘N’ in addition to A, G, C, and T. This N denotes that the structure at that location has not been identified properly, and is unknown. Hence, these N values are omitted from the sequences while generating the corpus and k-shingles upon which the later steps are based.

ARCHITECTURE

The system consists of 5 Python modules:

1) Parameters: This module stores all the global parameters required in the LSH retrieval process, namely: k = Shingle Length, b = Number of bands in which Signature Matrix is divided, r = Rows per band, t = Threshold for similarity, n = Number of Hash Functions used.
2) Shingling: This file reads the sequences from the given text file, cleans them of 'N' values and lists out all the possible k-shingles present in each sequence.
3) Minhashing: This module generates the Signature Matrix from the Document Matrix, using a suitable minhashing function.
4) LSH: This module divides the Signature Matrix into b bands with r rows each and performs Locality-Sensitive Hashing. For each sequence, the signatures corresponding to each band are dumped into a bucket, with similar signatures having a higher probability of going to the same bucket.
5) Main_Retrieval: This module is responsible for interacting with the user, processing the inputs fed by him, and fetching the results for the same, using the LSH based corpus generated in the previous modules.

DATA STRUCTURES USED

In places where O(1) search is required, Dictionaries are used with the search object as keys. For iterable structures, Lists are used. The exact structures of various database components are as follows-

1) Doc_Matrix: This is the Document Matrix. It is a nested dictionary (python hash table). The outer dictionary contains the sequence number as key, mapped to another dictionary. This inner dictionary contains all the k-shingles in the particular sequence as keys, and allows us to check whether a particular sequence has a particular k-shingle in O(1) time.

2) Shingle_Dict: This dictionary stores all the Shingles occurring in the corpus, mapped to their ‘order’. This order is used as input to the MinHash function, to generate random permutations for generating Signature Matrix.
3) Seq_List: This is a list of all the Sequences occurring in the corpus, and can be used to retrieve the Sequence from its ID.
4) Signature_Matrix: This is a nested dictionary (hash table). For each sequence number, the 100 Signature values of the corresponding sequence are stored. This allows retrieval of column wise signature values in constant time.
5) Hash: This is a list of lists, and stores the value of MinHash inputs for all the 100 permutations of MinHash, so that the same values can be used while processing the query.
6) Doc_Buckets: This is a dictionary of lists, in which the Bucket values are mapped to the list of Sequences hashed into that bucket.

PERFORMANCE

- One of the major advantages of our system is that the processing of sequences takes place only once. Hence, for resolving a query, very less time is required, since all the mentioned steps are performed only on 1 sequence.
- The run-time for our DNA retrieval system is less than 0.3 seconds even in the worst case scenario. Hence, in the long run when multiple queries need to be resolved, our system will perform very well.