# **LSH BASED HUMAN GENE TRANSCRIPT RECOMMENDER**

*OBJECTIVE*

The project aims at making **a LSH Similarity measure based recommender system for human genome transcripts**, where the **most similar documents** corresponding to a given query will be retrieved. The dataset is taken from **Kaggle.** It is a huge dataset containing more than four thousand gene transcripts.

*THEORY*

We are required to find the most similar documents corresponding to a particular query where the similarity index is given by us. Usually this can be done by calculating the Jaccard similarity for all document vectors corresponding to a query vector and then giving those documents as an output which are above the given similarity index, but that is computationally heavy when a large number of documents are to be considered. Hence we are using a technique called **Locality Sensitive Hashing**. This technique significantly **reduces the amount of computation** by using a mapping structure where **similar documents are hashed to the same bucket**. Later only the documents which fall under the same hash as the query are checked for similarity with the query. This can be performed on **a Boolean matrix** but since the number of shingles for large documents may be very large hence that is **not very efficient in terms of RAM** usage. This has been tackled by a method known as **min-shingling**. We have **taken random permutations of the Boolean matrix using hash functions** and **retrieved the first non-zero row indices corresponding to each document vector**. The matrix formed by this method is known as **Signature matrix. LSH has been performed on the signature matrix**.

*DATASET ARCHITECTURE*

The dataset consists of one text file containing 4380 sequences of transcripts and their corresponding classes.

*PRE-PROCESSING*

For pre-processing , the sequences are separated from their classes and appended to a list.

*CODE FLOW*

1. The user is asked for the shingle size in order to perform shingling.
2. A Boolean matrix is created which indicates the presence or absence of every shingle for every document in the corpus. The rows correspond to the shingle indices (the mapping of these indices to shingles is stored in a dictionary) while the columns correspond to the document index. This matrix is then saved in the text file to be used for min-hashing for all the queries which may be asked.
3. The user is then asked to give the query.
4. Taking the query first the query is processed and all shingles corresponding to the query are mapped on a Boolean matrix.
5. Min-hashing is performed on the Boolean matrix and a signature matrix is created using 100 permutations of the shingles. Similar approach is used for the query.
6. The Signature matrix is divided into 5 bands each corresponding of 20 rows. Corresponding to each band the documents are hashed into buckets using a hash function. Finally the hash function is applied on the query vector and the bucket corresponding to the hash key is chosen. For all the documents inside that bucket the query and document similarity is measured using the Jaccard similarity measure.
7. The most similar documents with similarity measure more than 80% are given as the output.

*TIME AND SPACE COMPLEXITY*

*S : Total Shingles*

*D : Total Documents in the dataset*

*H : Number of Hash function used*

*B : Number of bands*

*R : Number of rows*

*X : Average number of documents per bucket*

T1 = TIME COMPLEXITY SHINGLING : O(S\*D)

T2 = TIME COMPLEXITY MIN-HASHING : O(H\*D)

T3 = TIME COMPLEXITY LOCALITY SENSITIVE HASHING: O(B\*D)

T4 = TIME COMPLEXITY FOR CALUCULATING JACCARD SIMILARITY : O(H)

T5 = TIME COMPLEXITY FOR QUERY SEARCH: O(X\*T4)

S1 = SPACE COMPLEXITY FOR SHINGLING : O(S\*D)

S2 = SPACE COMPLEXITY FOR MINHASHING : O(H\*D)

S3 = SPACE COMPLEXITY FOR LOCALITY SENSITIVE HASHING : O(B\*(D/X))

S4 = SPACE COMPLEXITY FOR CALUCULATING JACCARD SIMILARITY : O(1)

*AUTHORS*

*ABHISHEK BAPNA : 2018A7PS0184H*

*ASHNA SWAIKA : 2018A7PS0027H*

*HARDIK PARNAMI : 2018A7PS0062H*