**Design Document**

1. **Introduction-**

The purpose of the project is to shortlist documents from a large dataset which are more likely to be partial or complete duplicates of one another. For faster detection of duplicates LSH is used, however, other methods like jaccard similarity and cosine similarity have also been presented.

1. **Dataset and Pre-processing**–

The dataset used consists of news articles from Inshorts. The dataset contains author, date, headline, read more link and the actual news text. However, for the purpose of this project, only ‘ctext’ section has been utilized. The section contains detailed news content which makes duplication detection easier.

1. **Shingling** –

For the shingling process the k-value chosen is 9. From the book “Mining of Massive Datasets”, it is advisable that while considering news articles, stop words or punctuations should not be removed. Thus, all the documents inside the datasets are divided into 9-shingles. After processing, the following data-structures are created –

* Dictionary ‘d’ – It is used to store all shingles corresponding a particular document. The key contains document ID and value contains list all unique shingles present in the respective document.
* Dictionary ‘shingle’ – The key is a particular shingle and the value contains a list of documents in which the respective shingle is present.
* len\_doc – It is an integer value containing the number of documents in the dataset.

1. **Shingle Matrix** –

The shingle matrix is stores by the name ‘finalArray’. It is a shingle to document Boolean matrix which has entry 1 or true if the shingle is present in corresponding document. Numpy library has been utilized in creating the zero matrix and the actual entries are later filled manually.

1. **Shingle Similarity** –

Jaccard distance measure has been used to calculated jaccard distance between all pairs of documents. The output has been stores in the file named ‘ir\_jacc\_sim.txt’.

1. **Minhashing and Signature Matrix Construction**–

The signature matrix is a reduced version of shingle matrix which is assumed to represent the shingle matrix by the use of 100 hash functions. It is stored by the name ‘sig\_mat’. The signature matrix is a 100 x (No. of documents) matrix. For the purpose of making signature matrix, 100 hash functions are generated. Those hash functions are of the form (ax+b)%c where x represents the row number from the shingle matrix. The ‘a’ and ‘b’ values are randomly generated and stored ‘mul\_list’ and ‘add\_list’ respectively.

1. **Signature Similarity (without bands)** –

For calculating the signature similarity, multiple methods have been implemented.

* Cosine Similarity – cosine similarity between each pair of documents is calculated using the signature matrix and stored in the file named ‘ir\_cosine\_sim.txt’.
* Signature similarity without making bands is also calculated and the results are stored in the ‘candidate\_pairs’ list. The results have also been stored in the file named ‘ir\_sig\_sim.txt’.

1. **Locality Sensitive Hashing**–

The signature matrix is further divided into ‘b’ bands of ‘r’ rows each. Each column of ‘r’ rows is hashed and mapped into a bucket. The documents hashed to the same bucket are more likely to be similar and can be considered for further advanced and rigorous duplication tests. The signatures have been compared in two ways-

* Calculating signature similarity scores between every pair of columns for each band and mapping them to a bucket only if they cross a certain similarity threshold. The following data-structures are created-

1. List ‘band\_can\_pairs’ – It is a list of lists which contains all the candidate pairs for a particular band.
2. Dictionary ‘all\_can\_pairs’ – It is a dictionary which maps all the candidate pairs for a particular band to its respective band number.
3. List ‘s1’ – it contains all the candidate pairs from a particular dataset irrespective of the bands they fall in.

* Mapping documents to a bucket only if the hash functions are exactly similar – The method named ‘hash\_signature2’ is used for this purpose. Further implementation details of the method have been presented in the ‘Documentation’ file

1. **Candidate pairs scores**-

For further shortlisting of documents returned by LSH, jaccard and cosine similarity scores have also been calculated for the candidate pairs. The methods named ‘cal\_jacc\_score\_candidate\_pairs’ and ‘cal\_cosine\_score\_candidate\_pairs’ have been utilized for this purpose. Further implementation details have been presented in the ‘Documentation’ file.

1. **Output Files** –

* ir\_jacc\_sim.txt – used for storing jaccard similarity values by independently calculating jaccard similarity scores for two documents. (Note – It makes use of shingle matrix only and does not require signature matrix for any computation)
* ir\_sig\_sim.txt – used for storing signature similarity for any two documents **without making bands**.
* ir\_cosine\_sim.txt – used for storing cosine similarity values by independently calculating cosine similarity scores for all pairs of documents. It calculates the cosine scores on the signature matrix and not the shingle matrix.
* ir\_row\_band\_sim\_scores.txt – used to store signature similarity for each column of a every band. It utilizes the signature matrix after making bands.
* ir\_sig\_sim\_band.txt– used to store final candidate pairs from each band. It also contains overall candidate pairs from the entire dataset.
* jacc\_and\_cosine\_sim\_using\_string\_hash.txt – This file stores the jaccard and cosine similarity scores of the shortlisted documents after the LSH process.