**Detecting Duplication documents using LSH**

1. **Introduction**
   1. Problems to be solved

Giving a query, which is also a set, you want to find sets in your collection that have Jaccard similarities above a certain threshold, and you want to do it with many other queries.

To do this efficiently, you can create a MinHash for every set, and when a query comes, you compute the Jaccard similarities between the query MinHash and all the MinHash of your collection and return the sets that satisfy your threshold.

1.2 Challenge

In our experiment, we implement one more precise algorithm and another more approximate algorithm, and compare these two algorithms efficiency and the accuracy.

Challenge1: Implement one precise algorithm on all the documents and get the documents which similarities between each other is higher than the threshold.

Challenge2: Implement another approximation algorithm on all the documents and get the documents which similarities between each other is higher than the threshold.

Challenge3: Compute F-score of the approximation algorithm and the precise algorithm and compare the approximation algorithm efficiency with the precise algorithm.

1. **Data description**

We crawler the news from BBC, New York Times, Guardian, and Yahoo. We collect the articles from 6 different categories, including politics, entertainment, sports, style, technology, business. We have 1171 documents in total. And in the following experiment, we will detect top-50 most similar documents for each document.

1. **Linear scan method**

3.1 Background

The linear scan method is comparatively appraised using a brute force approach to discern the closest similarities in document queries. We first generate minhashes and then do linear action to compare each minhash.

It will compute all the Jaccard distance between all the documents, that means it will be more precise and slower.

3.2 Experiment

To implement Linear scan on all the documents and do query and return the documents which similarity between each other is higher than the threshold. And also return the query time for this method.

In this experiment, we tried different number of permutations 32, 64, 96, 128, 160, 192, 224,256 to get corresponding results.

1. **LSH method**

4.1 Background

The idea behind locality sensitive hashing is to take the document fingerprints and chop them up into pieces, each piece being some number of minhashes. Since a single minhash (single entry in the fingerprint) has a probability equal to the Jaccard similarity of producing a collision, each chopped up portion of the fingerprint should as well. This chopped up portion is the locality in locality sensitive hashing, the hashing is just a hash function (any hash function) which produces a bin ID from the fingerprint locality being hashed. Each bin holds the entire fingerprint (with optional meta information) of the document and that of other documents that hash to the same bin.

Let's say our fingerprint has 100 minhashes in it and we chop the fingerprints into 10 pieces. Each piece of each fingerprint therefore contains 10 minhashes, we hash those again (not using minhash this time) to get a bin ID and store the whole fingerprint in every bin each of the pieces happens to land in.

When we want to know which documents are similar to a query document, we look in all the bins the query document lands in, any document in any of the bins is a potential duplicate. Comparing the full fingerprint of all documents in the bin or computing the actual Jaccard similarity between the shingle sets yields the final similarity of documents. Crucially since not all documents will land in the same bins we've reduced the number of comparisons needed to find similar or near duplicate documents.

The number of pieces to chop each fingerprint into and the size of each piece are parameters that need to be set. These should be set such that num\_pieces \* size\_of\_piece = num\_minhashes, this makes sense since having computed all the N minhashes we want to use all of them in the locality sensitive hashing part. There is however a further issue that needs to be considered when setting the parameters; the relation between the number and size of the pieces and the probability of LSH "finding" a pair of similar documents.

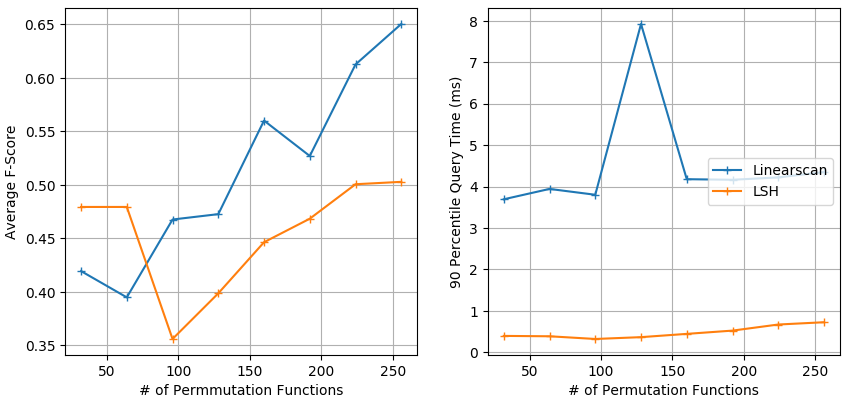
LSH is a probabilistic model which means that it won't always do the "right thing". Using LSH one needs to consider the similarity of a pair of documents (in this case the Jaccard similarity) and the probability that LSH will find that pair to be similar (a true positive, i.e. a correctly discovered duplicate pair). The pair of documents LSH finds to be similar should be thought of as candidate duplicates. The higher the probability, or guarantee, that LSH will find a pair of documents to be similar the falser positives the model will also produce, that is candidate duplicates that are not in fact duplicates.

* 1. Experiment

To implement LSH method on all the documents and do query and return the documents whose similarities between each other is higher than the threshold. And also return the query time for this method.

In this experiment, we tried different number of permutations 32, 64, 96, 128, 160, 192, 224,256 to get corresponding results.

1. **Result comparison**



From the above plot, we could conclude two conclusions.

For Average F-score, we know that normally the score of LSH is lower than the score of Linear scan, and with more permutation, the F-score of LSH will be higher. The LSH method could achieve at least 75% performance of Linear scan.

For query time, the time of LSH is around 0.5ms, and the time of Linear scan is around 4ms, it’s almost 8 times faster than Linear scan.

**Reference**

<http://cs.brown.edu/courses/cs253/papers/nearduplicate.pdf>

<http://infolab.stanford.edu/~ullman/mmds/ch3.pdf>