Finding Candidate Pairs Between COVID-19 Genomes Using LSH

1 Introduction

With the current pandemic situation we are experiencing, we have often heard of the many mutations of the Sars-CoV-2 virus. Our job will be analysing the DNA sequences coming from the its RNA . When we are infected, the virus attacks several cells, after 2/3 cell cycles the DNA sequence will be different from person to person. Passing though different people, after hundreds of infections the cycle changes considerably and mutations are born.

Therefore, similar DNA sequences are more likely to be contracted nearby while very different sequences are more likely to come from distant geographical areas. Our work is focussed on finding similarity between the genomes of various COVID-19 strains, specifically we use localtive sensitive hashing to identify the 'candidate pairs' of similar covid-19 genomes, and we expect them to have the same origin. We also aim to demonstrate the efficiency of LSH from the point of view of tine complexity and to demonstrate how it works better than classic deterministic algorithms.

2 Dataset Preparation

To get a more detailed idea of the data-set used for this work we show a transposed plot of the data-set itself, which will help us understand how this is formed.

	count	unique	top	freq
FASTA description	367	367	LR757995.1 Wuhan seafood market pneumonia viru...	1
Genome collection date	367	76	2020-03-13	21
Locality	367	71	USA: WA	106
Sequence	367	367	TTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCTG...	1
YP_009725297.1	367	10	MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHL...	344
YP_009725298.1	367	29	AYTRYVDNNFCGPDGYPLECIKDLLARAGKASCTLSEQLDFIDTKR...	251
YP_009725299.1	367	56	APTKVTFGDDTVIEVQGYKSVNITFELDERIDKVLNEKCSAYTVEL...	290
YP_009725300.1	367	17	KIVNNWLKQLIKVTLVFLFVAAIFYLITPVHVMSKHTDFSSEIIGY...	349
YP_009725301.1	367	9	SGFRKMAFPSGKVEGCMVQVTCGTTTLNGLWLDDVVYCPRHVICTS...	353
YP_009725302.1	367	8	SAVKRTIKGTHHWLLLTILTSLLVLVQSTQWSLFFFLYENAFLPFA...	332
YP_009725303.1	367	4	SKMSDVKCTSVVLLSVLQQLRVESSSKLWAQCVQLHNDILLAKDTT...	364
YP_009725304.1	367	7	AIASEFSSLPSYAAFATAQEAYEQAVANGDSEVVLKKLKKSLNVAK...	359
YP_009725305.1	367	2	NNELSPVALRQMSCAAGTTQTACTDDNALAYYNTTKGGRFVLALLS...	366

Figure 1: Dataset

As we can see in this schema, we have 13 features, 9 of which, the last 9, are referred to the protein. The first 4, that are the features most used in this work, identifies an ID, a date, a locality and a DNA sequence. Then the columns give information related to total count, single values, top element and frequencies of the top element. All those data, are quite useful to understand in which way is possible to interact with data set, starting to the the identification of the main parts. Another important consideration to put in place regard the identification of ”null” and ”na” value. To have the certainty that the considering data-set haven’t this type of element inside, are used functions like ”dnasequence.isnull().sum()” and ”dnasequence.isna().sum()”, where dnasequence is the name of the considering data-set and the sum function made the total amount of na or null value of a particular features.

3 LSH (Locality Sensitive Hashing)

The LSH is a data mining technique that allow analyst to find pairs that are likely to be similar. The general idea behind LSH is that we hash items using many different hash function. Then we can examine the candidate pairs, defined in this way after that them was located in the same bucket. An important point of this function is the identification of the major sub function that is important to take in to account:

- Shingling/K-mers creation
- Min-Hashing
- Locality-Sensitive Hashing

3.1 Shingle/k-mers

The Shingles creation is one of the main function of the LSH. This step allow to made up a series of sub-string of the total frequency, in this way is possible to apply the LSH. In this particular case the shingles generation is called "K-mers" generation, due to a bioinformatics definition. Indeed in bioinformatics, k-mers are sub-strings of length k contained within a biological sequence. Primarily used within the context of computational genomics and sequence analysis, in which k-mers are composed of nucleotides (i.e. A, T, G, and C), k-mers are capitalized upon to assemble DNA sequences, improve heterologous gene expression, identify species in metagenomic samples, and create attenuated vaccines. So it is important to define an appropriate length for what concern the k parameter. Considering the total length of the main DNA sequence, it's possible to use a k = 15 to have an optimal performance on the shingle generation, and then on the application of the Hash Function on them.

3.2 Min-Hashing

Min-Hashing is the next step in our process, allowing us to convert large sets to short signatures, while preserving similarity. We create a signature matrix by iterating through all columns in the input and applying a hash function on each shingle and finding the minimum hash code across all shingles. We do this $b \ x \ r$ times to get $b \ x \ r$ values in the signature for erach column. Creating the signature matrix has a time complexity of $O(Nbrs)$ because there are N columns whose signature must be computed. To compute each signature, we perform min-Hash by computing the minimum hash of the s shingles $b \ x \ r$ times.

3.3 Locality Sensitive Hashing

The final step in identifying similar sequences is the LSH function itself. We will be taking the banding approach to LSH.The banding method solves this problem by splitting our matrix into sub-parts called bands b. Then, rather

than processing the full vector through our hash function, we pass each band of our matrix through a hash function. The idea is that we identify candidate pairs as pairs of documents that share the same min-Hash signature across at least one band. To determine this, we create hash tables for each band containing all the bands of min-Hash that were obtained for all documents. Items that hash to the same bucket in these hash tables are considered candidate pairs. This works because, as said before, more similar documents are ore likely to produce identical min-Hash bands. For computing the time complexity of this step, we consider that for each band in the signature matrix we hash each segment of each column to a hash table, for a time complexity of *O(Nb)*

4 Data Visualization

4.1 Tuning number of Bands & Rows

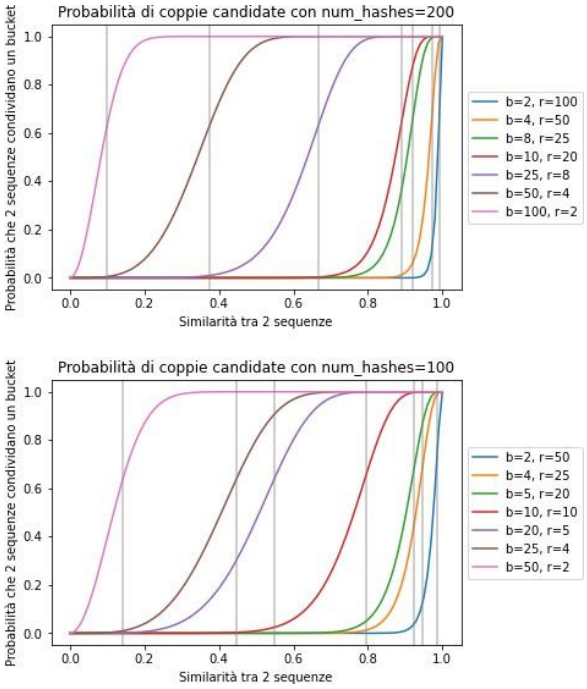

Figure 2: Different B&R matching on different n° of Hashes

The above example referring to the two graph is an explanation of the way in which change the similarity threshold when the hiperparameters B&R changes.As you can guess from the results reported in the first two graphs, there will be a tangible difference when the b and r parameters within the model changes. As b (bands) will decrease the number of r (rows) per band increase. This will lead, from a theoretical point of view, a reduction in the number of possible elements to be inserted in the various buckets for each band. The low number of r rows for each band will result in a much less marked threshold, which will lead to classify as similar even elements that are basically not. Vice versa, as bands decrease and rows increase, there will be more items to address to the bucket for each band, this will result in a more marked threshold. In the referring project case with 2 b and 50 r, for a total of 100 hashes, it is possible to show that most of the candidate pairs are distributed after the 97th percentile.

4.2 Distribution of candidate pairs

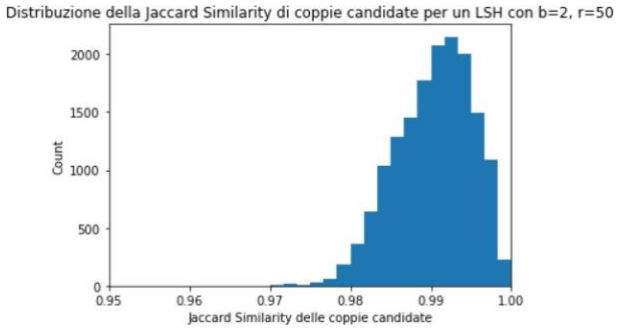

Figure 3: Histogram of Distribution of Candidate Pairs

The histogram above is the result of choosing the appropriate number of b&r to achieve the most efficient result, that is 2&50 respectively for a total of 100 Shingles. These data allow to underline the observation reported in the previous paragraph, that is that with a very low number of bands usually the threshold will be high. As can be guessed from a first look, the Threshold taken into account will be placed on the 97th percentile, as already defined above and all pairs defined by the LSH function will be distributed after this percentile. To be able to generate this histogram in such a way as to show the distribution we used a number of bins = 600, so it is easier to see because made the distribution smoother. The bins define the area of each bar in the histogram.

Another important and most intuitive way used to visualize the candidate pairs, consist of the plot of top 30 candidate pairs.

```
(('MN994468', 'USA: CA'), ('MN996528', 'China: Wuhan'))
(('MT281577', 'China: Anhui - Fuyang'), ('MT334547', 'USA: UT'))
(('MN985325', 'USA'), ('MT322401', 'USA: VA'))
(('MT345872', 'USA: ID'), ('MT350241', 'USA: VA'))
(('MT322408', 'USA: VA'), ('MT326119', 'USA'))
```

Figure 4: Candidate pairs and region of provenience

The schema above is just an overview of the total pairs plotted, but are enough to catch the meaning. In this way is possible to have a visual point of view of the way in which candidate pairs are located considering the geographic region. A count was implemented to have a proof of the difference between the total number of pairs and the number of pairs coming from the same region, and as a result it's possible to see that on 15947 total pairs, 9416 coming from the same region.

4.3 Analysis of time complexity

As it is possible to see, the entire algorithm is based on the probability that element with a kind of similarity get hashed together and for this reason the best result is not a guarantee. But why LSH must be used? The answer is the time complexity. Indeed the LSH function in faster than the normal deterministic algorithm. Let's see how it works. Suppose we are identifying similar pairs of N sequences, using a total of s k-mers. The deterministic algorithm requires comparison of all possible pairs of sequences for a total of *(N-1)N* possibilities.) Hence, the time complexity of the deterministic algorithm is $O(sN^2)$. Instead, what about the LSH algorithm, as we said before,the signature matrix has a time complexity of *O(Nbrs)* and LSH has a time complexity of *O(Nb)*, hence, the overall time complexity is *O(Nbrs)* . Ideally, though, this additional time would be much less than $O(N^2)$. Hence, with appropriately tuned b and r, LSH has a better time complexity. In this case is possible to observe that on the same number of permutations defined by *(b x r)*, the LSH function work with the same time. Rather, reducing the number of permutation , also the the time to create the signature matrix is reduced. The time to find the candidate pairs is not relevant. The time taken by the deterministic algorithm is, as expected, bigger.

	Runtime for Generating Signature Matrix	Runtime for Getting Candidate Pairs	Number of Candidate Pairs	Average Jaccard Similarity of Candidate Pairs
b=2, r=7	44.510330	0.473187	26019	0.988903
b=2, r=50	348.555498	0.092245	15947	0.989015
b=5, r=20	338.637326	0.186012	25686	0.989013
ALGORITMO DETERMINISTICO	/	751.183012	26131	0.988897

Figure 5: Time Complexity Table

5 Conclusions

This work have the aim to find similarity between genomes of various COVID-19 strains on Dataset composed by 367 DNA sequences. In particular, using LSH, can be identify candidate pairs of similar Covid-19 genomes and it's possible to expect that these candidate pairs correspond to where the strains originated from (e.g. CHINA strains are identified as candidate pairs). The output of candidate pairs coming from the same geographical area is the 60% on the total number of pairs. But some consideration must be made:

- The whole algorithm(LSH) is based on probabilities that elements of certain similarity get hashed together, and due to this probabilistic nature, there is no guarantee that the correct set of candidate pairs will be produces.
- Is not a certainty that genomes coming from different regions have different DNA strains, considering the high mobility of this historic period and the increasing relationship between Est and West.
- Dataset put in place is quite small to have a great accuracy for this type of analysis.