Additional Tools

A few tools remain to be explained before the algorithm can be described. These tools have a variety of reasons for being used that will be explained individually.

k-mer

In order to create the sets that $\mathbf{Mm}\frac{1}{2}$ and $\mathbf{Mm}$ sketches are built with, k-mer will be used to partition each sequence into subsets. The k-mer of a sequence string are defined as follows:

The k-mer of a sequence string s is the set of all the substrings of size k of s.

The 1-mer of a sequence string, will therefore be the set of all characters in the sequence string. It is therefore sensible to consider the size of k when partitioning the sequence string.

k-mer transformation

Given that the k-mers will be used as hash function input, it is reasonable to transform them to an input that is easy to map. As sequence strings only comprimise of 4 characters (A,C,G,T), each character is assigned a 2 bit value so that $A = \mathbf{00}, C = \mathbf{01}, G = \mathbf{10}, T/U = \mathbf{11}$, and then put them in sequence according to their position in the substring, eg.

A G T A C
0010110001

which also greatly reduces the memory usage.

MapReduce

Given a sizeable amount of sequences per file, it was quite essential to have a parallel and distributed programming model. For this purpose, MapReduce is a popular programming model. It works by distributing its task to a multitude of workers. Worker can be computers or cores. It expresses its computation as two functions:

1. Map: Runs a function over each element of a list and returns an intermediate value.
2. Reduce: Merges the intermediate values to form a potentially smaller set of values.

As explained in [**?**], MapReduce processes input by the following steps

1. Map step: Splits the input into M splits. Each split is then distributed to a worker who will perform a Map function on the given split and saves the result into a temporary storage.

2. Shuffle step: The results from the Map calls are then written to a local disk, partitioned into R regions.
3. Reduce step: For each region, a worker is set to run a Reduce job on it, in parallel.

MapReduce has been shown to scaler better than other parallel programming tools for input sizes that surpass 100 Mb, which is why it chosen.[?] Apache Hadoop MapReduce was used, as it is free source. However, MapReduce has been shown to be very slow at small input files[?], because of startup time, and a difficulty of partitioning small files.

Pig

Pig Latin is a high level language for compiling and executing MapReduce jobs over Hadoop. Advantageous when performing a pipeline of MapReduce jobs[?], it also demands very few lines of code compared to Hadoop MapReduce code. Additionally, users may write User Defined Functions (abbr. UDF), completely eliminating the need to write any map or reduce function, as they are on the lower level.

Levenshtein Distance

For a precise distance metric between two string, the levenshtein distance is quite applicable. Given two strings, s_1 and s_2, the **Levenshtein distance** is defined as the minimum number of single character edits necessary to change s_1 into s_2. The **Levenshtein similarity** could then be calculated as

$$sim = \frac{\max(s_1, s_2) - \text{LevenshteinDistance}(s_1, s_2)}{\max(s_1, s_2)} \tag{1}$$

where $\max(s_1, s_2)$ is the maximal length of s_1 and s_2.