CSE549

Project Report

Topic: Implementing Long Read Mapping Algorithms

Min Hash is a fast probabilistic method to approximate the Jaccard Similarity(J) of two sets.The performance of Min hash degrades when the sets are of different size. Containment Hash improves on this by the use of Bloom Filters for fast membership queries.

In our approach, we begin by setting the k-mer size, the probability error rate for bloom filter, maximum number of hashes to retain for each sequence, the number of k-mers formed from each sequence and the number of k-mers inserted into the bloom filter without collision.

Upon completion of this step we generate k-mers from the sequences using the **generateKmers()** function that is responsible for generating k-mers from the input sequence and storing it in a vector. It takes into account the number of k-mers to be generated and the size of the k-mer. The min hash sketch is populated with the smaller set using Count Estimator that includes the functions **add\_sequence()** which calls the helper function **add()** in which the size of the substring(k-mer) is used to compute k-mers of a sequence. It also has the functionality to send the reverse complement but presently this has been defaulted to False. The bloom filter is then created and populated with the larger set after which the k-mers in the min hash sketch are tested for presence in the bloom filter.

After this the results of the previous step are used to find the True Jaccard Index. **trueJaccards()** is responsible for this and it returns the true Jaccard similarity of the generated Kmers of the given sequences and it is used for comparison.

The following functions have been used to implement MinHash and Containment MinHash:-

**add\_sequence():**

The function add\_sequence() calls the helper function add() in which the size of the substring(k-mer) is used to compute k-mers of a sequence. It also has the functionality to send the reverse complement but presently this has been defaulted to False.

**add():**

The computed kmers are then added to data structure sketch and ensures that the sketch is sorted It also updates the count accordingly. Adding the kmer to the sketch occurs only if the kmer is not present in the sketch whereas the count is updated in case the kmer is present in the sketch.

Apart from the above mentioned functionalities, it will also have provision for calculating and comparing the minhash functions of a specific kmer with its complimentary hash and choose the lower of the two to proceed.

**MurmurHash64A():**

Murmur Hash is a family of hashing functions suitable for general hash based look-up and is non-cryptographic. The MumurHash64A has been optimized for 64-bit processors.

**get\_prime\_lt\_x():**

It is used for generating a prime for hashing. The functions computes the largest prime number less than the target passed as argument. It calls helper function is\_prime().

**generateKmers():**

This function is responsible for generating k-mers from the input sequence and storing it in a vector. It takes into account the number of k-mers to be generated and the size of the k-mer.

**trueJaccards():**

This function returns the true Jaccard similarity of the generated Kmers of the given sequences.