Long Read Mapping Algorithms

Poojitha Ponakala

Department of Computer Science, Stony Brook University, NY 11790

Abstract

Motivation: Min Hash is used to estimate overlap of two long reads by estimating their Jaccard Index, which is the ratio of number of elements in the intersection of two long reads and the number of elements in their union. However, Min Hash doesn't work well for data where one data set is much smaller than the other. Also, the probability of deviating from true Jaccard Index increases as Jaccard Index decreases. To overcome these limitations of Min Hash, a new approach called Containment Min Hash is implemented which uses Bloom filters for fast membership queries. Both of these methods are useful for Taxonomic classification. They are used to detect presence or absence of a genome in metagenomics sample. Containment Min Hash, in particular, is used to detect the presence of very small, low abundance microorganisms in a metagenomics sample.

Results: Jaccard Index estimate from Containment Min Hash is closer to the true Jaccard Index, with less relative error, compared to the estimates from Min Hash for increasing kmer size and number of hash functions. As difference in the size of both sets increases, Jaccard estimate from Containment Min Hash is closer to True Jaccard and less very less relative error compared to the estimate from Min Hash. It is also faster to find Jaccard Index estimate through Containment Min Hash approach.

Availability: www.github.com/pponakala/longreadsmapping
Contact: poojitha.ponakala@stonybrook.edu

1 Introduction

Long reads are the results of third-generation sequencing of genomes and transcriptomes which, unlike the short read sequencing, (i.e. second or "next"-generation sequencing) methodologies, generate a very long substrings of the reference, but have a higher error rate compared to short reads (~10-15% vs <1% for short reads) and hence should be treated using algorithms, data structures and approaches other than those commonly used in short read analysis.

Specifically, approaches that use the idea of Min Hashing or related algorithms have recently drawn a lot of attention in the computational biology community, and have been shown to be useful in assembling and mapping long reads. In this approach, which is borrowed from NLP and estimating document similarity, we use a Min Hash to approximate overlap of two long reads or approximately map a long read to a reference.

In addition to this, another approach named containment hashing has been proposed that as an advantage to Min Hashing is addressing potential length differences between the two strings being compared.

2 Methods

Calculating jaccard index is time consuming for all genomic data sets. Estimating jaccard index is faster and better. Min Hash approach is traditionally used for estimating jaccard index.

2.1 Min Hash Approach

Given two sets A and B, we randomly sample from A U B and find number of points in the random sample that fall into A ∩ B. This is used to estimate jaccard index.

Jaccard Index: (A ∩ B / A U B)

However, Min Hash approach has limitations:

- Min Hash doesn't work well for data where one genome or data set is much smaller than the other.
- The probability of deviating from true jaccard index increases as jaccard index decreases to zero
- Works best for sets of similar sizes and significant overlap

To overcome the limitations of Min Hash, another approach called Containment Hash is used.

2.2 Containment Min Hash Approach

Containment Min Hash overcomes the limitations of Traditional Min Hash approach:

- It is appropriate when sets are of different sizes

- It gives a Jaccard estimate that is closer to the true Jaccard index compared to Min Hash approach.

Instead of randomly sampling from A U B, we randomly sample elements only from smaller set A. We then test if these elements are present in set B using bloom filter.
We use the above to estimate containment index and estimate jaccard index from containment index.

Containment Index = intersection / # hash functions

Estimated Jaccard Index =
(size_A * containment_index)
/ (size_A + size_B_est - size_A * containment_index)

2.3 Algorithms

The following are the steps for calculating similarity with Min Hash approach and Containment Min Hash approach.

2.3.1 Min Hash Algorithm:

(1) Generate kmers of A and B
(2) Populate bloom filter with kmers of larger set B
(3) Get the kmers which give min hash values for the hash functions
(4) Calculate intersection by checking how many of the min hash kmers of A from Step 3 are in the bloom filter
(5) Adjust for false positive rate
(6) Estimate containment index
(7) Containment Index = intersection / (float)hash_count
(8) Estimate Jaccard index from Containment Index
(9) Estimated Jaccard Index =
(size_A * containment_index)
/ (size_A + size_B_est - size_A * containment_index)

2.3.2 Containment Min Hash Algorithm:

(1) Generate kmers of A and B
(2) Populate bloom filter with kmers of larger set B
(3) Get the kmers which give min hash values for the hash functions
(4) Calculate intersection by checking how many of the min hash kmers of A from Step 3 are in the bloom filter
(5) Adjust for false positive rate
(6) Estimate containment index
(7) Containment Index = intersection / (float)hash_count
(8) Estimate Jaccard index from Containment Index
(9) Estimated Jaccard Index =
(size_A * containment_index)
/ (size_A + size_B_est - size_A * containment_index)

2.4 Implementation

Both Min Hash and Containment Min Hash are implemented in C++.

The approaches are compared with varying the kmer size and number of hash functions.

Python2.7 is used for plotting graphs from the generated output data.

Estimated Jaccard Indices vs True Jaccard Indices are plotted for both Min Hash and Containment Min Hash approach. Relative Errors for increasing kmer size and number of hash functions are also plotted for Min Hash and Containment Min Hash approach.

Simulated long reads data and reference genome data is generated using Python2.7 code before using the mapper to map the long reads to the reference genome.

Threshold for determining if a long read maps to a reference genome can be provided as a command line argument.
If it is not given at command line, then threshold will be calculated as

Threshold = $(1/b)^{1/r}$,
where b is number of bands with each band of r rows.

2.5 Limitations

The code reproduces the results but it is not of "industrial strength."

With this implementation, the estimates for Jaccard Index are most accurate results when kmer sizes are >15.

As a consequence, the graphs plotted are not as smooth as those in the original Containment Min Hash paper.

3 Results

- Jaccard Index estimate from Containment Min Hash is closer to the true Jaccard Index compared to the estimates from Min Hash.

To estimate Jaccard Index by running Min Hash and Containment Min Hash, run the following commands:

```
./run.sh <file1> <file2> <kmer_size> <hash_functions>
<false_positive_rate> <number_appended_to_results_files>
```

The above script generates simulated data and runs both Min Hash and Containment Hash algorithms.

Example:

```
./run.sh data/file3.txt data/file4.txt 20 200 0.01 2 150 1000

./run.sh data/file1.txt data/file2.txt 18 1000 0.02 3 15 1000
```

Sample output from above commands:

```
c++    -c -o main.o main.cpp
g++ main.cpp hash.cpp include/MurmurHash3.cpp -o main
[+] Generating Data
[+] Reference Genome file
[+] Long Read file
[+] Estimate Jaccard Index with both approaches
EXAMPLE 3: using data/file1.txt and data/file2.txt
./main -g <smaller_genome_file> -r <reference_genome_file> -k <kmer_size> -h <nu
mber of hash_functions> -f <false_positive_rate>


file1 = data/file1.txt
file2 = data/file2.txt
hash count = 1000
false positive rate = 0.02
A size = 77
B size = 1023
kmer size = 18

Calcuting Similarity:
True Jaccard = 0.00566038
Estimated Jaccard from Min Hash = 0.00300903
Estimated Jaccard from Containment Min Hash = 0.00579611

Calcuting Relative Error:
Relative Error with Min Hash = 0.468405
Relative Error with Containment Min Hash = 0.02398
```

- As difference in the size of both sets increases, Jaccard estimate from Containment Min Hash is closer to True Jaccard and low relative error compared to the estimate from Min Hash.

Sample output generated when A and B are of significantly different sizes:

A size = 99
B size = 1022
True Jaccard = 0.00462107
Estimated Jaccard from Min Hash = 0.0010015
Estimated Jaccard from Containment Min Hash = 0.00417403
Relative Error with Min Hash = 0.783275
Relative Error with Containment Min Hash = 0.0967395

- With increasing kmer sizes, Jaccard estimates from Containment Min Hash have very low relative error compared to the estimates from Min Hash.

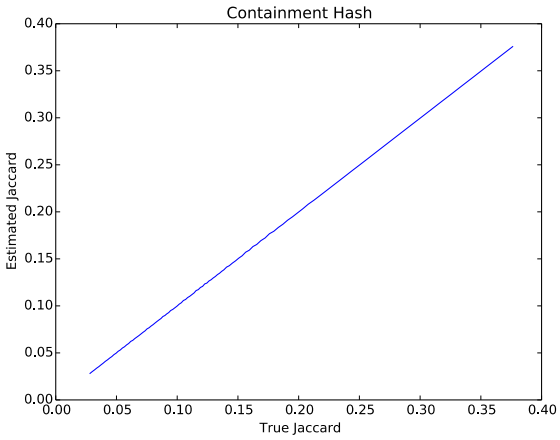
To compare both approaches and reproduce the graphs, run the following commands:

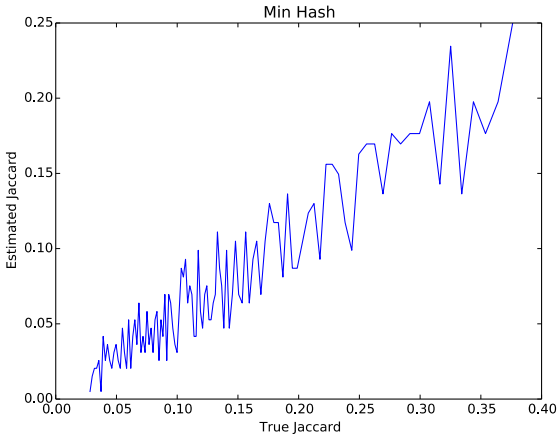
```
./run.sh data/filex3.txt data/filex4.txt 20 150 0.04 3

./run.sh data/filex1.txt data/filex2.txt 25 1000 0.01 4
```

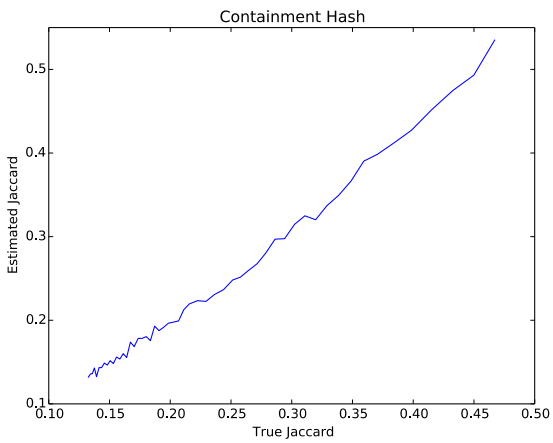
Following graphs below show how estimated Jaccard index varies with True Jaccard Index for increasing kmer sizes:

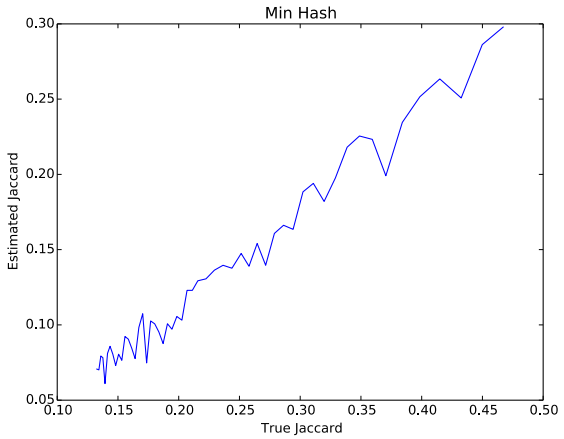
Example 1:





Example 2:





- With increasing hash functions, Jaccard estimates from Containment Min Hash have very low relative error compared to the estimates from Min Hash.

- It is also faster to find Jaccard Index estimate through Containment Min Hash approach.

- Mapping long reads to reference genome: Long reads and Reference genome data is generated, which is the used to map the long reads to the reference genome

To map long reads to a reference genome, run the following script:

```
./query.sh <long_reads_file> <reference_genome_file> <threshold> <kmer_size> <hash_functions> <false_positive_rate> <number_of_long_reads_to_generate>
```

The above script generates the data (long reads file and reference genome file) and maps the long reads to the reference genome if the similarity is higher than the threshold

Example:

```
./query.sh data/reference.txt data/longreads.txt 0.05 18 100 0.01 5

./query.sh data/reference.txt data/longreads.txt 0.05 20 200 0.02 10
```

Sample output:

```
long read 4:-
TAGGCCTCCGGATCCTGATAATTCGCATATCTGGCGTTCTCCACCCTGACCGAGTTGTCGGCGAGATTAGCCCGGAAAGT
GGCGCGGACGGGAATACGTCCGCTATCAGC
(Containment) Est. jaccard_index = 0.000784856
Below Threshold. This Long Read DOES NOT map to the Reference Genome
Similarity = 0.0784856

long read 5:-
CACTATTCTGAAACCGCTCCAGAGAGATTCGCCGTCCTGGCCTTTCAAATAGGCGCGACCCGTGCAGAACGACTTTTGAT
GATACTATGTTAAGAGGGCATTGCTCTTATC
(Containment) Est. jaccard_index = 0.0852245
Above Threshold. This Long Read maps to the Reference Genome
Similarity = 8.52245
```

References

- https://www.biorxiv.org/content/biorxiv/early/2017/09/04/184150.full.pdf
- https://github.com/dkoslicki/MinHashMetagenomics
- http://infolab.stanford.edu/~ullman/mmds/ch3.pdf
- Hash Functions generation https://www.eecs.harvard.edu/~michaelm/postscripts/rsa2008.pdf
- Bloom filter http://www.partow.net/programming/bloomfilter/index.html
- Mapping reads http://sfg.stanford.edu/mapping.html
- Bloom filter https://archive.codeplex.com/?p=libbloom
- Murmur Hash3 https://github.com/aappleby/smhasher
- https://www.uow.edu.au/~lukes/TEXTBOOK/notes-cpp/