Midway Report

Problem Statement

I will attempt to create a clustering algorithm, that can compete with, or even surpass, uClust in terms of speed, without losing precision, when working with a dataset of 500000 RNA/DNA sequences of length 500-1500.

Analysis

The very high number of sequences that have to be compared means that the speed of the algorithm that i develop is extraordinary. uClust is at the moment number one contender in terms of speed without loss of much precision.

Studies have shown that the implementation of Minhash in [**?**] have approached both the speed and precision of uClust. This method is quite different from the centroid based approach of uClust, and therefore spurred my interest.

My attempt will be to apply some improvements to the Minhash implementation above, and see whether i can achieve the same speed and precision as that of uClust. This implementation i will call Max-Minhash.

Status

Currently i have finished the following tasks:

1. Finished a prototype of my Max-Minhash algorithm. It has the following functionalities
   (a) Singlethreaded implementation of a Max-Minhash algorithm
   (b) A hashfunction based on the one in [**?**]
   (c) Reads sequences from .seq files.
2. Finished a draft of the following two sections: Universal Hashing and Minhash

What is left to be done are the following tasks

1. A comprehensive test of several universal hashing schemes to find a method that might be faster than the currently used one.
2. Developing the prototype so that it has the following improvements
   (a) The newfound quickest hashfunction.
   (b) Programmed to work in a Mapreduce framework, so that it can be run in Hadoop
   (c) Able to read .fasta files. Has to be able to handle sequences that contain the character N.
3. Comprehensive tests of my algorithm in comparison to the uSearch algorithm, to see which is faster and most precise

4. As my algorithm uses K-mers as the basis for its Minhash input, a comprehensive test of which k-mer size is the most optimal for precision and speed.
5. Documenting all the tests that i perform above.
6. Writing a conclusion.
7. Improving the sections i wrote about universal hashing, and the Minhash section

It should be noted that quite a few of these tasks can be completed without having 2.b ready, read. 1, 2.c, and 4. These will therefore be done parallel to implementing the Mapreduce framework, some using the current prototype.

Time plan

I will estimate each of the above remaining tasks so that an implementation/test task likewise includes the time it takes to document this implementation/test.

1. Test of universal hashing schemes.
   (a) **Product:** Imlpementation of, and a comprehensive test of several hashing schemes. Includes graphs and data. Includes conclusivee findings.
   (b) **Resources:** Java, .seq test data
   (c) **Dependencies:** Section on Universal hashing
   (d) **Workload:** 2 days
2. Implement IO handler that can read Fasta files
   (a) **Product:** Implementing IO handler that can read fasta files.
   (b) Resources: Java, fasta files
   (c) Dependencies: Defining how to handle sequences containing the character N, Java
   (d) **Workload:** 1 day
3. Mapreduce Framework
   (a) **Product:** Improving the prototype to be runnable in a Mapreduce Framework.
   (b) **Resources:** Java, Hadoop
   (c) **Dependencies:** Knowledge of Hadoop, Researching Mapreduce
   (d) **Workload:** 10 days
4. Test of K-mer sizes
   (a) **Product:** Documented tests of k-mer sizes to find the appropriate k-mer size for RNA/DNA sequences. Includes data and graphs of results.

(b) **Resources:** Java, .seq files

(c) **Dependencies:** A Gold standard, Java, .seq test data

(d) **Workload:** 2 days

5. Final comparative test

   (a) **Product:** Documented final comparative test of Max-min implementation alongside uClust. Includes data and graphs of results.

   (b) **Resources:** Java, Hadoop, usearch

   (c) **Dependencies:** A finished implementation of Max-min, A sorted .fasta file for smallmem usearch.

   (d) **Workload:** 3 days.

6. Section about my algorithm

   (a) **Product:** Writing a section describing my algorithm. Includes examples from code, etc.

   (b) **Resources:** Latex

   (c) **Dependencies:** Final implementation of Max-Min hash algorithm.

   (d) **Workload:** 1 day.

7. Finetuning the report.

   (a) **Product:** Writing the conclusion and finetuning the other sections. Making a red thread between all sections. Assuring the scientific correctness of introductory sections about minhash and universal hashing.

   (b) **Resources:** Latex, Articles about previous sections, feedback

   (c) **Dependencies:** Section about algorithm, Section about Minhash, Section about universal hashing, The documented tests described above.

   (d) **Workload:** 10 days.

8. **Section about Minhash:**

   (a) **Product:** Writing section about Minwise independent permutations and MinHash.

   (b) **Resources:** Latex, Articles about Minhash

   (c) **Dependencies:** None.

   (d) **Workload:** 2 days

The timeplan will be describe so that the number above will be completed within the week it is set into:

Week 17: (1), (2)
Week 18: (3),(4)
Week 19 - 20: (3),(6)
Week 21: (5), (8)
Week 21-23: (7)