Introduction to Focus Areas in Bioinformatics – WS19/20

Prof. Dr. Tim Conrad

13 Project 13

- Deadline: 02.02.2020, 23:59
- All files need to be available through your GIT repository, in the directory “Project 13”.
- You can work in teams up to 3 people. Please state in the report the group member names.
- If your code is in python, I must be able to run your code within a Google Colab notebook. If your code is not in Python or R, you must provide a manual how to compile and run it on a Linux machine.

(This project is based on a lab project by Prof. Goldman, U Washington)

A DNA sequence is a string of characters, from the alphabet {a, c, g, t}. Genomic DNA encodes a large collection of features, e.g. genes or regulatory sites. The DNA (sequence) is subject to mutations, however, functional sequence features are quite resistant to mutation. A common approach to find these features is to compare DNA sequences from two different organisms and see which parts of the sequences have remained similar to each other.

Abstractly, we are given two long strings s1 and s2, and we want to find short substrings common to s1 and s2 (we assume that these are parts of some features). Since even functional sequences mutate over time, we know that the common substrings might not be exactly the same. However, in practice one finds that s1 and s2 often exhibit exactly matching substrings of at least 10 to 15 characters in a neighborhood of shared features. By detecting these exact substring matches, we can find the most likely locations of the shared features in s1 and s2 and can then apply more sensitive but more expensive similarity search algorithms only at those locations.

Naively, we could find k-mers (substrings of length k) common to s1 and s2 by comparing every k-mer from one sequence to every k-mer from the other. Such an approach would take worst-case time Θ(|s1| * |s2|), which is unacceptable because interesting DNA sequences range from thousands to billions of characters in length.

13.1 Get Data

Find two genomic datasets of reasonable size that can be compared. (e.g. two full smaller genomes or only one chromosome of each). You can find genome data here:
https://www.ncbi.nlm.nih.gov/guide/howto/dwn-genome/

13.2 BWT based Search

Call s1 the corpus string and s2 the pattern string, and assume we are searching for common substrings of length k. We first build an index for the pattern string. Then, for each k-mer in the corpus string, we check whether it occurs in the pattern string.

- Implement this strategy (in C++) based on the BWT and the FM index.
- Hint: We are not interested at which position the k-mers have been found, but how many have been found.
- Test your implementation for k=10 and k=15.

13.3 Bloom Filter based Search

Call s1 the corpus string and s2 the pattern string, and assume we are searching for common substrings of length k. We first construct a set S of every k-mer in the pattern string. Then, for each k-mer in the corpus string, we check whether it occurs in the set S; if so, we have found a match between pattern and corpus.

- Implement this strategy (in C++) based on a Bloom Filter.
- Hint: We are not interested at which position the k-mers have been found, but how many have been found.
- Test your implementation for k-mer sizes of k=10 and k=15.
- Test your implementation for 3 different false positive rates (e.g. 0,1%, 0,001% and 0,00001%). Report the type and number of hash functions used and the size of the resulting bit-vector array.

13.3 Deliverables

Your need to upload all source codes and a report to your GIT repository.

- The report should be about 300-600 words in length (this is roughly 1 pages, depending on your layout).
- The report must be delivered in PDF format using the BMC template (including the abstract as defined in project 5).
- The following sections must be present (you can add more if needed):
    - Data
      Briefly describe which data you have used (including raw data size).
    - Implementation
      Describe briefly your implementations.
    - Benchmarks
      Show the benchmark results comparing the wall-clock run-time of your tests.
      Do not forget to report how long it takes to build the index, where applicable.
    - Discussion
      Briefly discuss your results. This should include a brief comparison of the algorithms with respect to the task of comparing genomes.