Assignment 1: Abstract

Modern Sequencing machines generate a large amount of data that must be processed. However, the existing algorithms and best tools are not perfect and produce archives.

We will use compression algorithm called FQSqueezer that sequences data and processes single- and paired-end reads of variable lengths, and what was mentioned above was based on the dynamic Marco encoder algorithm.

The disadvantage of the proposed method are large memory and time requirements.

Introduction

Illumina's instruments generate the vast majority of available data at a low cost compared to other sequencers.

The sequenced reads are short but with excellent quality, on the other side The PacBio or Oxford Nanopore 3rd generation instruments can produce much longer readings, but with worse quality and lower throughput.

Unfortunately, items like short-distance repetitions of sections of data are rare in read sets, so researchers looked for other options. They initially concentrated on the compression of bases to collect reads from near regions of genomes.

Our algorithm” FQSqueezer” make use of the ideas from the prediction by partial matching (PPM) and dynamic Markov coder (DMC) general-purpose methods.

Some suggested techniques: the organization of the huge dictionaries, design of the PPM-like estimation of probabilities, use of a custom DMC as the stage following the PPM, the technique for prediction and correction of sequencing errors, technique of ordering the reads and making use of shared prefixes.

The main aspect of FQSqueezer is its compression ratio, usually much better than of the state-of-the-art competitors, i.e., FaStore, Spring, and Minicom. however, our tool has some drawbacks in terms of speed and memory usage" it is a few times slower than the mentioned competitors in compression and much slower in decompression".

Assignment 2 : Related Works

Given that the amount of data that needs to be sequenced and stored is large, and this will consume a lot of effort and money, it is not surprising that a large amount of research has been conducted to solve this problem.

The first step was Using gzip, (which is a general purpose compressor). Reducing the file size by nearly 3x was impressive, but the deluge of data demanded more. The biggest drawback to Gzip is that it is mainly built for text data, or more specifically, data with repetitive textual types.

The next step was to create specialized algorithms that consider the different types of redundancy found in FASTQ 3 files.

As the details of the proposed algorithms differ, researchers have begun to look for alternatives. They initially focused on pressing the law. The basic idea was to rearrange the data in order to capture readings from regions close to the genome, this may seem like a loss of data.

In the following years, other researchers explored the concept of using minimizers, i.e., short substrings of sequences, to find reads from close regions.

In 12" https://www.nature.com/articles/s41598-020-57452-6#ref-CR12", it is illustrated how the reads are grouped from slightly larger genome regions. However, three recent research papers describing HARC, Spring and Minicom have had much better results. Attempts differ in detail, but they are all based on the same principles.

Assignment 3 : Results

Tools and datasets : For the evaluation, we used the latest competitors, such as FaStore, Spring, and Minicom, for the test, and decided not to test some of the other successful compressors, such as BEETL, Orcom, AssembleTrie and HARC, because previous research on them showed that they performed worse than the tools we chose. The older utilities are not competitive in terms of compression ratio, (see table1 for experiments with one of datasets"https://www.nature.com/articles/s41598-020-57452-6/tables/1"). Some of them are very fast as DSRC 27. However, in this article we mainly focus on the compression ratio.

Compression of the bases : The compression of bases is the most important issue right now. We evaluated the tools in this scenario in the Previous experiment. The results for the single-end reads were given in Table 2" https://www.nature.com/articles/s41598-020-57452-6/tables/2 ", and the results for the paired-end (PE) reads can be found in (Table 3"https://www.nature.com/articles/s41598-020-57452-6/tables/3").

The disadvantages of the FQSqueezer are the time and place requirements (see Table 4"https://www.nature.com/articles/s41598-020-57452-6/tables/4"). FQSqueezer can be classified as PPM algorithm in decompression.

Full FASTQ compressors : FQSqueezer is the FASTQ compressor, we tested its output on a few datasets. There were two competitors: FaStore, Spring, and Minicom were built only for bases.(Table 7"https://www.nature.com/articles/s41598-020-57452-6/tables/7") shows the results of three different modes , In the lossless mode, all data were preserved. In the reduced mode, the IDs were truncated to just the instrument name and the quality values were down-sampled to 8 levels (i.e., Illumina 8-level binning). In the bases only mode, only the bases are stored. The results of the experiment verified FQSqueezer's compression ratio advantage.

Methods

Basic definition of an algorithm : FQSqueezer is a multi-threaded algorithm, we will start with one version for clarity. Both single-ended (SE) and paired-ended (PE) reading are accepted by our method. Readings can be saved in their original order (OO) or rearranged (REO). The reads are initially sorted according to the DNA sequence (first reading) in the rearrangement mode.

Compression of bases : when compressing a SE read (or first read of a pair) We process the bases from the beginning of a read position by position . We calculate the statistics of k-mers ending at this base in the already processed part of the input data for each base.

For more understanding and details let us follow the example given in Fig.1 " https://www.nature.com/articles/s41598-020-57452-6/figures/1" .The size p prefix of a SE read (or the first read of a pair in the PE mode) is encoded differently in the REO mode. We encode the difference between it and the integer representing the prefix of the previous read by treating it as a 2p-bit unsigned integer. The suffix is compressed similarly to how it is compressed in OO mode. The first read of a pair in the PE modes is compressed in the same way as it is in the SE mode. The second read is a little more complicated to process, but it is the same in both OO and REO modes. Figure 2 illustrates how a pair of minimizers can be used to estimate some of the second read b-mer values." https://www.nature.com/articles/s41598-020-57452-6/figures/2".

Compression of IDs : IDs are compressed in the lossless mode in the same way as they are in state-of-the-art compressors like Spring or FaStore.

FQSqueezer also offers two data loss modes. In the first, only the tool name (the first part of the identifier) is preserved, and if the current tool name is absent from the list, it is explicitly coded. In the last possible situation, identifiers are ignored.

Compression of quality score : Rating scores can be compressed in five different ways, each of which allows for a different number of values: 96, 8, 4, 2, and 0. No conversion needed if the entered FASTQ file did indeed reduce the Quality Score resolution (e.g., 4 for Illumina NovaSeq sequences).

Implementation details : C ++ 14 programming language was used to build the application. For collision handling, most dictionaries are implemented as linear scanning hash tables. We use prefetch software to reduce delays caused by cache misses. The native C ++ threads are used to implement multithreading.