Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Application Note

Application Note

Quaff: overlap detection and reference genome alignment to nanopore reads

Ian Holmes[1]*

[1] Department of Bioengineering, University of California, Berkeley, 94703, USA.

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX

Abstract

Nanopore sequencing generates abundant long reads with a relatively high error-rate, motivating the development of an error-tolerant read alignment tool that can be easily parallelized. Quaff is an open source command-line tool for aligning long FASTQ-format reads to a FASTA reference genome using a conditional pair HMM (transducer) that also models quality scores. The software implements pre-filtering based on kmer-matching; parameter-fitting via unsupervised Baum-Welch training; read overlap detection; and multithreaded, cluster- and cloud-based parallelization.

1 Introduction

Nanopore sequencing typically generates long, noisy reads: one recent study reported substitution and indel error rates around 5%, and read lengths around 4kb (Jain *et al.*, 2015). This entails different alignment strategies from short, accurate reads, where compressed data structures have proven highly useful for fast indexing of exact (or close-to-exact) matches (Lindner and Friedel, 2012). For longer, more error-prone reads, parameterization of the error model is of greater importance; indeed accuracy on individual datasets can often be improved by fitting a custom statistical model, rather than using a one-size-fits-all parameterization.

Several signals can be exploited in the pursuit of improved alignment accuracy. For sequencing technologies such as Oxford Nanopore, where the raw signal is derived from a convolution of overlapping k-mers, gap and substitution rates may depend on recent sequence context. An additional, and under-exploited, signal is available in the case of FASTQ data which contains quality scores, as well as base-called nucleotides. According to the usual definition of FASTQ format, these "Q-scores" are nominally to be interpreted as PHRED scores (Cock *et al.*, 2010); i.e., the logarithm of the probability of error. Such an interpretation offers one way to incorporate Q-scores in a statistical analysis. However, the distributions of Q-scores from different sequencing technologies and base-calling pipelines may be very different (Utturkar *et al.*, 2015) and there can be no guarantee that the relationship between Q-score and actual observed error rate is consistent across such different regimes.

Here we describe a way of using quality scores in sequence analysis without making any assumptions that they represent exact error probabilities. Our method exploits only the fact that there is a measurable (and quantifiable) difference in profile between quality scores that correctly align to a reference genome, and those that don't. The quality score distribution, along with the k-mer dependence of error rates, is modeled directly, to enhance alignment performance by giving the alignment program extra clues about which regions will align well.

2 Methods

Pair Hidden Markov models (HMMs), along with transducers (their input-conditioned relatives), are well-suited to the problem of aligning noisy, long reads (Durbin *et al.*, 1998). The emission and transition parameters of an HMM can be learned in an unsupervised way (i.e. without user-supplied training alignments), using the Baum-Welch algorithm, a version of the Expectation Maximization (EM) algorithm. The expected emission and transition counts computed by EM are themselves relevant to some statistics of interest (for example, sequence coverage, percentage identity, or indel frequencies). Furthermore, HMMs (and transducers) are easily extended. Incorporating adjunct data into the HMM emissions (such as quality scores) is straightforward; and transducer theory allows the systematic derivation of elaborated alignment tasks (e.g. finding the overlap of two reads, hypothesizing that they were both derived from a single unknown reference) without the need to re-parameterize the elaborated model (Westesson *et al.*, 2012).

It should be noted that the maximum-likelihood alignment, as returned by the Viterbi algorithm, is not necessarily the most accurate alignment obtainable with a given parameterization: the decision-theoretic "optimal accuracy" alignment is better (Holmes and Durbin, 1998). Nor indeed does the maximum-likelihood parameterization, as given by Baum-Welch, in general yield the most accurate Viterbi alignments of any parameterization

(Drasdo *et al.*, 1998; Frith *et al.*, 2010). Nevertheless, Quaff implements Viterbi alignment (because it is around threefold faster than "optimal accuracy") and Baum-Welch training (because, to our knowledge, no unsupervised training algorithm yields more accurate parameterizations).

Quaff's underlying transducer incorporates the k-mer signal directly by allowing the substitution and gap probabilities at any given position to depend on the last k nucleotides of the read sequence. (It would be a better reflection of reality to allow the weights to depend on the last k nucleotides of the reference sequence; however, since the reference is not directly observed in many applications, such as read-to-read alignment, this would increase the computational expense of those applications.)

Quality scores are modeled using a negative binomial distribution with emission-dependent parameters. They are required for training, but are optional in alignment and overlap modes.

The read-overlap algorithm is a variant of Viterbi that probabilistically sums degenerate paths through gap states (so that biologically unmeaningful permutations of the gap order are marginalized). The underlying pair HMM is a minimal (3-state) approximation to a larger (27-state) pair HMM derived using phylocomposer, which implements algorithms for combining transducers (specifically, the 27-state machine arises from the intersection of two reference-to-read transducers, composed with a single-state generative HMM for the common underlying reference). The 3-state approximation trades meticulous accuracy for pragmatic run times.

Model parameters are saved in a JSON format, as are expected emission and transition counts (computed by the Forward-Backward algorithm); the latter can be used as pseudocounts during subsequent training runs (effectively specifying a Dirichlet prior distribution on parameters). Alignments can be output in a variety of formats, including Stockholm, gapped FASTA, and SAM.

To accelerate the dynamic programming (DP) algorithms, we use a strategy of pre-filtering by word-matching, as used by BLAST (Altschul *et al.*, 1990) and DALIGNER (Myers, 2014). Specifically, the DP is constrained to fixed-size diagonal bands around a small number of seed diagonals that contain more than a certain threshold of matching k-mers between the sequences being aligned. These thresholds can be specified directly by the user; alternatively, Quaff can be configured to select the largest threshold that will fit either within a user-specified memory limit, or within available system memory.

For additional speed, Quaff offers several multiprocessor modes to parallelize computations on large read datasets: **(1)** It can utilize multi-core architectures on a single machine by distributing DP jobs over a thread pool. **(2)** It can launch server jobs on remote machines, effectively increasing the size of the thread pool by using an existing cluster. These server jobs are long-running, and are controlled by the master node over sockets. If the Amazon Web Services (AWS) command-line tools are installed (and the user has an AWS account), Quaff can instantiate a temporary cluster on the AWS Elastic Compute Cloud (EC2), download and build itself on the temporarily created EC2 instances, and use them as servers for the duration of the run (automatically terminating the instances after the run). In order to distribute data to server nodes, Quaff can either transfer files using the `rsync` command, or can alternatively use 'buckets' of the AWS Simple Storage Service (S3). None of these parallel functions require any queueing or other cluster management software, except `ssh`, `rsync` and (in the case of EC2/S3) the AWS command-line tools. **(3)** If a job queueing system (such as Portable Batch System or Sun Grid Engine) is installed, together with NFS, then Quaff can use this system to distribute the workload over the cluster.

Quaff shares some similarities with marginAlign (Jain *et al.*, 2015), which is also a transducer-based aligner for Oxford Nanopore reads which can run on a cluster; the primary differences with the Quaff model are in the parallelization and pre-filtering options, the k-mer dependencies of model parameters, and the explicit modeling of quality scores. Quaff also shares some similarities with DALIGNER (Myers, 2014), in that it uses pre-filtering based on k-mer word matching to accelerate the DP.

3 Availability

Quaff is available at `https://github.com/ihh/quaff` under the Creative Commons Attribution 3.0 US license. It is written in C++11, and compiles on a POSIX system either with Clang (v.6.1.0) or gcc (v4.8.2). It requires the GNU Scientific Software library (libgsl), the ZLib compression library (libz), and (if compiling with gcc) the Boost library.

Acknowledgements

Quaff includes code from Ivan Vashchaev (gason), Jeff Donahoo (PracticalSocket), Heng Li (kseq), Rafael Baptista (stacktrace), David Robert Nadeau (memsize), René Nyffenegger (base64), Roger Pate (string escaping), Konrad Rudolph (index sort), and the StackOverflow community. The code to fit the negative binomial distribution uses exact solution of the log-likelihood maximum for the probability parameter followed by Brent's minimization to find the optimal dispersion parameter, after Stephen Crowley (2012, `http://vixra.org/pdf/1211.0113v1.pdf`).

Funding

This work has been supported by NHGRI grant R01-HG004483.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.*, **215**(3), 403–410.

Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. (2010). The sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.*, **38**(6), 1767–1771.

Drasdo, D., Hwa, T., and Lässig, M. (1998). A statistical theory of sequence alignment with gaps. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **6**, 52–58.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

Frith, M. C., Hamada, M., and Horton, P. (2010). Parameters for accurate genome alignment. *BMC Bioinformatics*, **11**, 80.

Holmes, I. and Durbin, R. (1998). Dynamic programming alignment accuracy. *J. Comput. Biol.*, **5**(3), 493–504.

Jain, M., Fiddes, I. T., Miga, K. H., Olsen, H. E., Paten, B., and Akeson, M. (2015). Improved data analysis for the MinION nanopore sequencer. *Nat. Methods*, **12**(4), 351–356.

Lindner, R. and Friedel, C. C. (2012). A comprehensive evaluation of alignment algorithms in the context of RNA-seq. *PLoS One*, **7**(12), e52403.

Myers, G. (2014). Efficient local alignment discovery amongst noisy long reads. In *Algorithms in Bioinformatics*, Lecture Notes in Computer Science, pages 52–67. Springer Berlin Heidelberg.

Utturkar, S. M., Klingeman, D. M., Bruno-Barcena, J. M., Chinn, M. S., Grunden, A. M., Köpke, M., and Brown, S. D. (2015). Sequence data for clostridium autoethanogenum using three generations of sequencing technologies. *Sci Data*, **2**, 150014.

Westesson, O., Lunter, G., Paten, B., and Holmes, I. (2012). Accurate reconstruction of insertion-deletion histories by statistical phylogenetics. *PLoS One*, **7**(4), e34572.