ECS240 Project Proposal

Camille Welcher

February 18, 2016

A Drop-in Data Validator for Bioinformatics Applications

Introduction

The advent of high-throughput sequencing has made bioinformatics software increasingly important for basic biology. With so many publications relying on these programs for their results, and most analyses involving complicated pipelines with many different programs, verifying the results of thee programs becomes challenging. Worse, most bioinformatics software is maintained by academic labs with limited resources to spend on engineering, and is often poorly tested and prone to failure.

While biological meaning is difficult to verify and highly application-specific, basic integrity of outputs is very approachable. I myself have encountered numerous cases where programs failed to handle IO errors correctly and simply `exit 0` after producing invalid output, which often is not caught until upstream when other programs crash on the invalid input (also often without proper error codes). A simple utility to perform this validation would be quite useful, and would open the door to some generalized testing methods for bioinformatics software.

Description

I propose to build a simple streaming data validator for genomics applications. This utility would read a data stream from standard input, validate it given some set of assertions, and stream the data back out on standard output. Upon encountering invalid data, it would simply exit with an error code; if used with standard POSIX pipes, this would also exit the source program.

On its face, this is an extremely simple utility. It becomes more interesting when we start to consider conditions. For example, many bioinformatics applications

consume a stream of short sequences, filter some out, and write out the results; to validate these programs, we might wish to specify that the output stream be a subset of the input stream, and crash when this is not the case. Many programs perform machine learning approaches and filter more data as they learn; we might wish to supply a simple function describing the rate of filtration as a function of how many sequences have been observed. The ability to specify these sorts of conditions would allow the utility to be used as both a drop-in validator for pipelines and a testing tool for build systems.

Implementation and Plan

A major component of this utility would be writing valid parsers for major bioinformatics data formats; bad parsers will of course lead to incorrect validation and only introduce additional bugs to already-buggy pipelines. I would use an existing parsing tool such as Ragel ("Ragel" 2014) to specify each format using a Backus-Naur Form (BNF) or other context-free grammar; such a collection of grammars would be highly useful to the community, and would be portable to many languages. One project, biojulia ("Bio.jl" 2016), already uses Ragel for its parsers, but there is not a large collection of grammars available, and julia as a language is still in its infancy. My hypothetical utility would instead be written in C and wrapped in Python. Python has a large collection of existing libraries for BNF's and is widely used in the bioinformatics community, and writing the core in C will ensure that it can be dropped in to existing pipelines without noticeable performance losses.

I will need to do further research on methods for describing the conditions, in hopes of finding a method which is expressive while also being comprehensible. I foresee this component being the most interesting result of the project.

I will aim to implement the utility on some core formats first (FASTA, FASTQ, SAM, GFF3, Newick) and further extend it at a later date.

References

"Bio.jl." 2016. https://github.com/BioJulia/Bio.jl.

"Ragel." 2014. Colm Networks. http://www.colm.net/open-source/ragel/.