Consensus Promoter

Prediction software analysis

daniel huffer

2019

Table of Contents

# Introduction

The application that I have optimised is the provided Java application; Bioinformatics - Consensus Promoter Prediction. To start my initial analysis of the application I began by first researching the context of the application. Quickly I was able to learn that "Bioinformatics is a hybrid science that links biological data with techniques for information storage, distribution, and analysis to support multiple areas of scientific research, including biomedicine" [https://www.britannica.com/science/bioinformatics]. I then started looking into Consensus promoter prediction where I discovered that a promoter is a region of DNA that leads to initiation of transcription of particular genes [Analysis of Biological Networks: Transcriptional Networks – Promoter Sequence Analysis] and that promoter prediction is a common element of many gene prediction methods. Based on this preliminary research I deduced that the application was a gene matching tool, meaning I expected it to perform vast amounts of pattern matching, which is typically a computationally time-consuming task.

# Analysis

After having done some preliminary research I began by opening the code within Jet Brains IntelliJ IDE where I observed the project contained an external jar dependency, a couple of sub base packages and an Ecoli directory a bunch of large GBK encoded files; where GBK is an extension of the character set GB2312 used for simplified Chinese [<https://www.iana.org/assignments/character-sets/character-sets.xhtml>]. After a quick look through each of the java files I decided to run the application. I tried briefly to run the application through IntelliJ and by manually running by the terminal to no success. To keep moving forward I started by making some simple modifications I had in mind upon selecting the application to optimise. These changes included

1. Refactoring of the project structure
2. Addition of spring boot framework

The resulting project structure can be seen in Appendix x – Project structure.

After then having run the application and observed the output, I started looking into profiling the application to start understanding why it took so long to run (INSERT TIME IN HERE ). To profile the application, I loaded up JProfiler having used it previously for other tasks.

# Appendix

