Database Tier Implementation to Chromosome Browser

Jenny Su

Birkbeck, University of London

May 5, 2018

My approach to the project

After our team agreed with each of our tasks, I started exploring the data file and the genBank flat file format description from NCBI website to get a general sense of how genbank file is formatted and to look for hidden patterns. I used project requirements as my guideline and discussion with frontend person to decide which information are relevant for extraction. I then conduct some research online for information related to genbank parser and text mining, studying codes from other people and Biopython to understand the structure of a working script and what functions can be used to achieve the goal.

Interaction with the team

Communications with team members were mostly through the Slack channel of our team (group 12) and code reviews in GitHub repository. As I am from a biological science background, sometimes I find programming terms used in our online group discussion confusing. For a programming amateur like me, I believe I would benefit more from face-to-face clarification. Nevertheless, some of the issues were still able to resolve through remote communications.

Requirements for my contribution

Python scripts for database and data access tier were developed through test-driven development approach. I was able to write a Python application that extracts relevant data from GenBank and constructs a chromosome database with those data. I also created a database API so that middle layer can use functions in the API to retrieve required information from the database and further deliver that information to the frontend layer.

Performance of the Development Cycle

At the beginning of this project, our group decided on what genome information we would want to display on the web browser and the agreed APIs for layers to communicate. I then came up with a draft physical schema of the chromosome database, so that each member could start working individually to construct layers with the knowledge of what information to expect from the database end. Unfortunately, we did not have enough time for finalising the test module and full implementation of the chromosome browser as we each had own technical hurdles to overcome. However, individually each layer was able to execute completer their task with dummy data.

Development Process

I started with developing a script to extract relevant information from chromosomal 18 genBank file which has the least gene information entries. I went through the cycle of writing code that would fail and then revising and modifying it to pass until I was able to get a working script that would parse all the required information and calculate codon usage for each entry and for the whole chromosome. I then tested the script with the actual file, chromosomal 12 genBank file, and some modifications had to be made for the script to work. For easy viewing and a better understanding of what information has been extracted and what information is missing from genBank file, I put a write to CSV file function code which can be disabled if it is not needed. The next stage was to create database and tables in MySQL Server using Python with PyMySQL. While going through another cycle of test-driven development to establish the MySQL database, I noticed there was a flaw in my initial database design which I then quickly rectified. After database establishment, I then reviewed and finalised the agreed APIs for the middle layer.

Coding testing

As mentioned above, code testing was initially done on chromosomal 18 genBank file, and all the modules worked successfully. The python application was able to parse required information and create chromosome database on localhost server from start to finish. I also tested the application on chromosome 12 genBank but had to terminate half way because the large amount of data in the file was causing the script to run over more than an hour.

Known issues

There were troubles getting the database to work on hope server and the issue was mainly due to access permission. Another problem was each layer failed to integrate, as a result, we don’t have a working chromosome browser.

Problems and Solutions

Individual elements of each layer might have worked successfully in its environment but due to different versions of Python/packages used could cause compatibility issues. This problem can be solved by setting up different environments. Another problem was getting the database to work on Hope server. I have tested that the database works effectively on the localhost server, so I believe the python application is working properly. But, if we had more time to get it working on a free server and each of the team members invests some effort in integration, then we could have a fully working browser. Lastly, I think issues also was arisen from unclearly defined APIs and members not implementing the all the compatibility issues of greed API function. With a database website project like this, an effective API is the most crucial element for backend and frontend to collaborate with each other.

Alternative Strategies

It would be more convenient and efficient to develop an effective REST API and to use backend framework such as Python powered Django and already developed libraries. It would enhance collaboration and help keep code organised.

Personal insights

Coming from a non-computer science background, I find this project extremely challenging and daunting. There are numerous computer science terms I don’t understand, and I have never written such a complicated application that needs to consider so many different aspects. And on top of all these, I have no actual knowledge or experience of how backend and frontend collaborate. Therefore, at the beginning of this project, I spent a substantial amount of time researching information relevant to complete this project. I learnt from examples online about what functions and techniques can be used for text mining, how to write clean codes, how to document code with docstrings format, and how to break a big task into small achievable parts. Also, when my script failed or was not performing as what expected, I searched for solutions on forums like 'stackoverflow', modified the codes with the advice and tested the codes again. And this is the most important thing I have learnt from this project is to keep trying. Another tool I find it useful is code review on Github. When you are so close to your work, mistakes can be so easily overlooked. A second pair of eyes from a team member is always an effective way to catch bugs.