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 have used project requirements as my guideline and discussion with frontend person to decide which information are relevant for extraction. I then conducted 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 our 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.

Since the deadline of project submission was extended for one extra week, the frontend person and I decided that we wanted to use the extra week to integrate all layers, further refine our scripts and implement a working chromosome browser which we had trouble reaching this stage previously. Unfortunately, a conflict of opinions arose over the goal of implementing a working website, and we were unable to get the middle layer person on board with this matter. Therefore, we had to create a new middle layer that communicates information from database to the frontend.

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 that information. 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.

Performance of the Development Cycle

At the beginning of this project, our group decided on what chromosome 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 be expected from the database end. Middle layer and frontend were able to complete requirements of their task with dummy data while I was working on a script to parse relevant information from genBank and to build the database on my local server. Due to access privilege issue, we had troubles getting the database to work on Hope server which prevented each member to test and modify their scripts for integration. Initially, we were unable to finalise a fully functional chromosome browser before the original project submission date. However, the hurdle of creating a database in MySQL on Hope server was later overcome by creating tables in existing student user database on Hope. As mentioned previously, because the project submission date got extended, frontend person and I were dedicated to using that extra time given to unify all layers for establishing a working website, but we were unable to get the person responsible for middle layer to agree on this matter. Therefore, by referring to his original scripts, we created a new middle layer. During the integration stage, we went through the cycle of testing and modifying our scripts multiple times. The product of this integration was a fully functional chromosome browser.

Development Process of Database and Access Tier

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 series 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 frequencies for each entry and for the whole chromosome. I then tested the script with our actual file, chromosomal 12 genBank, and some modifications had to be made for the application 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 local 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 with regards to using the wrong attribute as the primary key in my initial design and I had to quickly rectify it. After establishing the database, I then reviewed again and finalised the agreed APIs for the middle layer.

Coding testing

As mentioned above, codes testing was initially done on chromosomal 18 genBank file, and then on chromosome 12 file. The python application was able to parse required information and create a database on localhost server from start to finish. Since the final goal was to have a chromosome website accessing information from the database in MySQL on Hope server, connection script had to be modified to suit this use.

Known issues

Although we were able to get the chromosome website up and running, we failed to display restriction enzymes cutting sites on DNA sequence. This was partly due to inability to create a working restriction enzyme search function in the business logic layer. Another issue was webpages design. An extensive amount of information was both shown on Gene and Protein summary pages. Such a way of showing information on the website may not be user-friendly.

Problems and Solutions

Individual elements of each layer might have worked successfully in its environment, but different versions of Python/packages used could cause compatibility issues. This problem can be solved by setting up different environments for testing. As mentioned above in the known issues section, we failed to deliver the search function for restriction enzymes cutting sites on DNA sequence. But such a letdown can be fixed properly in the business logic layer if we were able to further refine the script. Another issue stated previously was the huge amount of data displayed on a few web pages. Again, this problem can be resolved easily through modifying website features in the future. Lastly, problems also arose from unclearly defined APIs at the start of this projects and later members not implementing all the agreed API functions. 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 expected, I searched for solutions on the internet and on forums like 'stackoverflow' and modified the codes with advice provided and tested the codes again, and this is the most important lesson I have learnt from this project. Another tool I find it useful is code review on Github. Sometimes, when one is so close to their own work, mistakes can be so easily overlooked. A second pair of eyes from a team member is always an effective way to catch bugs.