Biocomputing 2 Reflective Essay

Tiina Talts April 2022

Bioinformatics MSc
Birkbeck University of London

Biocomputing 2 coursework was handed out to the groups on 8th of March 2022. Our group was Group #8 consisting of 4 members in total.

1) Approach to the project

a. Interaction with the team:
The group proceeded to create a WhatsApp group and schedule regular meetings to discuss project outline and API code. The individual contribution for the project layers was identified quite quickly. WKC created the WhatsApp group and adapted the role as the group leader. First meeting was held on 14th of March 9:00am as an online call. Participants were WKC, TT and AO. During which the general outline of the webpage was created using interactive Google Drawings. General agreement was reached on how the overall webpage should work. The APIs and preparation for the course session presentation on 15th of March was discussed. Second meeting was held on 20th of March 9:00am as an online call with all four members WKC, TT, AO and KK present. APIs were discussed and working code input and output lists between DBAPI and BLAPI were discussed as main points.

b. Overall project requirements:
Overall, the specification documentation and material for the project requirements were very extensive. Group members were at different levels of familiarising themselves with the requirements fully. This resulted in unwelcomed surprises even in the middle stages of the project. Generally, such problems were handled, but would have been avoided if there was a better knowledge of the requirements from the start.

c. Requirements for my contribution
As I was responsible for the development of the business logic layer, I started by thoroughly familiarising myself with the requirements. Form there outlined a general structure of the code that would need to be written to fulfil the various business layer (BL) tasks to help identifying the input data types that I would require from the database layer (DB), and output data types that I would pass on to the front end (FE).

2) Performance of the development cycle

The group did not spend long time in design stage. It was moved on fairly quickly into the development stage as it seemed that everyone was on the same page when it came to design. The group leader was very responsive on the group chat whenever there was a problem or question, which was very helpful and reassuring. The dummy API code for the BL was laid out at the first stages of the development. The DB layer moved on to creating real API code early on, skipping the dummy APIs. Which consequently meant there was abit of uncertainty around that at the start, especially for the BL.

I would say we went through iterations during the development as the data parsing from the chromosome file proved to be more difficult than expected. Lot of the data was non-eligible due to coding regions spanning into another accession entry or protein translation not being available for the entry. Total of 861 entries of the human chromosome 10 file, 310 were eligible to use for the project. That is 36% only. There were some changes to the APIs during the development for both BL and DB layers, but this was expected. Overall, the code and data flow between the DB and BL was flowing relatively well. I did not see FE webpage development at all, therefore it is difficult to say how it all worked out for the FE and if the return data that was provided for the FE was unproblematic. There was hardly any feedback on that from the FE unfortunately.

3) Code testing

I adapted a simple 'test.py' executable file where return data was either saved to a file if very large in size; or printed to a shell for smaller sized return data. I also tried to run functions from the terminal in Pandora but this often returned error messages for syntax that was working in Python version 3.6.8 in IDLE3 as opposed to Python version of 2.7.5 when ran from the Terminal.

Before the DB was created I tested my code with a simulated input file where the data was an entry from the actual chromosome 10 data file. I created many simulation data files in the process of development to try many possible scenarios.

4) Known issues

There should not be any issues or bugs currently in the BL code. Everything has been tested out as described above, and everything should work as intended. However, its not known how the output from the BL code is working for the FE and if any issues arising there. It has not been communicated back to me if anything could have been done better for the FE. Without the feedback it is difficult to assess how easily the output from the BL code is handled in the FE layer.

Additionally, there was in the plans to store the once extracted coding regions back in the database. The coding regions were extracted for all entries and saved in a text file meanwhile. The BL code at this stage re-extracts the coding region each time the query is run instead of taking it straight from the DB.

5) What worked and what didn't - problems and solutions

Pandora, although very reliable, is a bit clunky and cumbersome to use. Therefore the bulk of the code was tried out on JupyterHub first before pasted over to Pandora and tested as part of the general workflow between the layers. The code that I have written for the BL is very 'organic' as no Biopython was used, which I personally find very enjoyable to write code that way.

The code and functions layout or structure could perhaps be more modular for the purpose of least redundancy as possible and make the code as sleek and quick as possible. But this perhaps comes with experience and as a beginner programmer something to be aspired to. Additionally, there seems to be redundant code in the DBAPI that BL didn't use in the end. But it is appreciated that there were many options for the BL. If this project would be continued the next steps would be eliminating redundancy and making the data flow more efficient overall. And as mentioned above, the coding region stored in DB instead of re-extracting each time the query is ran.

However, I am quite pleased with the enzyme code. The code works universally to any enzyme it is given i.e. added to the enzyme list text file. The current enzyme list is derived from NEB HF sticky end enzyme list that has 30 enzymes listed in total. The High Fidelity enzymes include most well known and regularly used restriction enzymes like BamHI, EcoRI, SpeI and XhoI that is a prototype for BsuMI. HF enzymes provide rapid digestion in 5 to 15 minutes, high activity in universal buffers. The code accommodates the degeneracies in the enzyme sequences. However, not all degeneracies are added into the code, only the ones that feature in the current enzyme list. If more enzymes added to the list the code need revisiting and more degeneracies added if needed.

From the group interaction side as it was already mentioned above - would have been exciting to try out the actual FE webpage.

6) Alternative strategies

a. I would have strongly recommended the FE webpage to be designed on the provided demo version from the beginning, rather than separately outside of the project area.
b. The group meetings should have continued regularly beyond the first two meetings
c. Would have been good if all group members contributed equally well
d. Finally, the coursework requirements and specification documentation was very lengthy with duplications and repeats that was sometimes contradicting. It would be helpful if one concise and precise requirements document from one source only was published.

7) Personal insights

Parsing biological data is a non-trivial task. But correctly parsed data and well set up database and data sets a strong base for the subsequent code operations and calculations.
I would have wished to learn more about the web page development – the challenges and difficulties when developing a biological analysis webpage.

Overall, this course work was quite challenging and a very useful experience to go through.