Biocomputing II – Reflective Essay

3rd May 2022

Denzel Eggerue

Requirements for my contribution

As the lead backend developer, it was paramount that I fully understood the purpose of the database and had an extensive understanding of both Python and SQL. The database forms the foundation of the entire project, and without it the other layers are rendered useless; the database must communicate with the business layer, which in turn communicates with the presentation layer.

I needed to create a relational database that would be used to store relevant data from the genbank file; this would be done by writing a parser in Python, and then convert the extracted data into an SQL table. A data access tier also needed to be made, by wrapping the SQL data in python to create an abstraction of the database that can be interacted upon by the business logic layer.

Before getting to writing any code, I spent some time reading the genbank file so I could understand its format and how best to parse this information into a database. I looked for patterns and regularities in the file that would make my job easier; it is this initial assessment that made my job easier, rather than trying to solve problems as they come up.

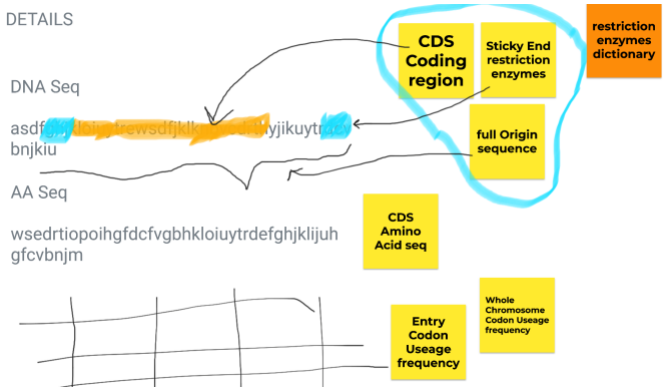
As a software engineer by trade, I had prior Python and SQL experience which was a massive boost for my role; I was able to use my subject matter expertise to suggest and implement the best possible code for our database and advise my teammates on whether it would be viable to implement their ideas or not. My depth of knowledge made me a natural fit for the role, and also meant my teammates felt comfortable trusting me with arguably the most important layer.

The Development Process

As mentioned earlier, I spent the first day or two reading the actual genbank file itself, to understand its format and how best to possibly extract the necessary information. Below is a screenshot of the very first iteration of my code; its purpose was to insert a demarcation after each entry in the genbank file, so that each entry would be easier to extract as a whole. After some careful deliberatino with the team, we found that my initial design wasn't as effective as I'd hoped, hence it was redesigned.

```
from distutils.file_util import write_file


filename = '/Users/Denzel/Documents/Birkbeck/chrom_CDS_1.txt'
with open(filename,'r') as f:
    file = f.read()
    new_file = file.split('//\n')
    with open('organised_file2.txt','w')as nf:
        for line in new_file:
            nf.write(line + "\n" + "--------------------------------------------------------------------------------" + "\n")
```



The team and I then generated a rough mockup of what needed to be extracted (see image above) and put into the database; this formed a guideline of sorts for me. In the beginning, I wanted to create the database locally using MySQL Workbench on my Mac, as it was an environment, I was comfortable with; however, it became apparent that it would be too difficult. Consequently, I decided it was best to connect to my MariaDB instance associated with my student account and go from there instead.

Rather than the somewhat outdated methodology of working in silos, although I was the lead developer for the backend it was very much a collaborative effort. We would meet every Tuesday and Thursday using Google Meet, and they would watch me as I wrote the code via screen-sharing. This allowed me to have an extra pair of eyes for spotting any errors, but also meant that when I hit a brick wall, my teammates were immediately available to provide suggestions and reinvigorate me. We repeated this process until I eventually completed the database design, and we applied the same principle to the business logic layer and the presentation layer.

Performance of the development cycle

Our unorthodox ‘all-hands-on-deck’ approach worked particularly well for this project. Not all members of the team were as tehcnically proficient when it came to writing code, so this approached allowed me to support those who had less experience writing such expansive code. It also meant that we could pick each others’ slack up whenever the lead developer for a particular layer was unavailable for whatever reason.

Initially, I wanted to implement a similar strategy as to what I’ve been used to as a software engineer, using scrum techniques and progerssion-tracking software like Jira; this would have theoretically made us more efficient and accountable for our code. Given the time constraints and the fact that my teammates were not from a technical background like myself, I concluded that it would not be in their best interests to implement such strategy.

As a whole, the strategy worked given the makeup of our team, but also meant that we spent more time than required in some instances.

Testing

When I finally finished the database, I ran multiple tests both in Jupyter and the front end itself. With Jupyter, I tested the database’s ability to recall large amounts of information incrementally. Starting off by first calling 10 entries, then 25, 50 and eventually 100. The limiting factor at this point became I/O threading of my own machine, rather than the database.

For the entire application itself, we ran several unit tests followed by some integrated tests. Our unit tests consisted of testing small portions of code by itself and then scaling up. For example, I tested every code block I wrote individually, before even testing how well it communicated with the business layer through the data access tier. Our integration tests consisted of using different search parameters on the front end; this meant that we could test both the presentation layer, business logic and peristence layer simultaneously.

Whenever we encountered an error during an integration test, we would systematically change different things to see where the problem was; starting from the database layer up to the front end, we would make small changes until the website worked again, thus telling us where the bug was.

Known Issues

Due to the way the cgi scripts were written, we were unable to get the pages past the landing page, to match the same background colours. Although a minor issue, it would've been desirable. For the database, sometimes it may not parse a single entry (out of tens of thousands) and will return it without any tangible values; the entry may read "accession number, protein product, gene" instead of containing the actual data inside those columns.

Problems and Solutions

My decision to use Regex as the main means of parsing information from the genbank file was largely successful; Regex is highly configurable and applicable in almost any scenario that requires parsing. However, there was an occasion where I realised Regex may not be the best solution, and that was when it came to passing the entire Origin of every entry. Firstly, it was far too difficult to create an appropriate matching pattern due to the extreme variation in the lengths of each sequence; secondly, it would be far too laborious on my computer to run such a command for over 10,000 entries, even running the regex commands at times would freeze my computer. Hence, I opted for BioPython only when it came to extracting the origin of each entry. This was a much easier process and used significantly less computing power.

Alternative Approaches

Knowing what I know now, a potentially better way of parsing the origin for each entry would've been to use a state machine of sorts. I would have set a flag to 'True', and when it hits a certain character(s) the flag will revert to false, at which point it would read over the entry character by character. This too, however, would've been incredibly time-consuming given the sheer number of entries in the genbank file.

Personal Insights

This was a very enjoyable project for me, and probably the most enjoyable piece of coursework I've done throughout the degree. It had a more practical focus on the side of bioinformatics I enjoy most: writing code. This task stretched my capabilities both as a bioinformatician and programmer, for which I am particularly pleased with. I am proud of the work my teammates and I produced.