Tuan Nguyen
tnguy256@jhu.edu
AS.410.712.82 - Adv. Practical Comp. Concept for Bioinformatics
Final Project – Narrative

Web Interface for Searching an Anti-Cancer Drug Database

Background

With the wealth of information that is available on the internet it can be difficult to access the data you want. One instance I see this being a hinderance in is drug development/research. In drug development, you test drugs on cells prior to moving to human subjects, but it can be cumbersome if you must search for whether a certain drug has been tested on a certain cell line already. In addition, results such as IC50 may not be readily available. With my web interface, I hope to make this data easily accessible, so that users can streamline their research. Specifically, this data will be tailored towards cancer drug research, where users can enter a drug into the search bar and get back cell lines the drug has been tested on, target enzymes/proteins, and more.

Tool Development

Starting with the SQL database (db), I had to revise the structure significantly from my proposal (Figure 1). After reading the dataset documentation, it was clear that the original structure wasn't adequate for the relationships between variables. The code will be included in the "load_data_final.py" file on the class server and GitHub, but I made 7 tables and used the 'keys' table to go between them.

With the db made, I moved on to writing the HTML code. I have an 'index.html' file for a landing page, and then a 'template.html' file for the rendering of results. After that, I wrote the CGI script. The script involved taking input from the 'index.html' file, running a SQL query with the input, and then parsing the query output to render 'template.html.' All the code will be available on GitHub and the class server (`/var/www/html/tnguy256/final`).

Validation and Challenges

My first task was to clean the data and load it onto a SQL db. To do this, I wrote a python script that took the data from the CSV/XLSX files that was provided on the Genomics of Drug Sensitivity in Cancer (GDSC) website (Yang et al., 2013). I quickly figured out that the initial

data structure I made was not sufficient, as there are many "n:n" relationships in the data. For example, for one drug, there could be one alternate name, no alternate name, or even multiple alternate names. So, I revised my data structure, created the tables and added the variables as such in Figure 1. To ensure that I added the data correctly into the database file, I used DB Browser for SQLite, which is a program that allows you to quickly browse data that is in databases in a user-friendly interface (*DB Browser for SQLite*, n.d.).

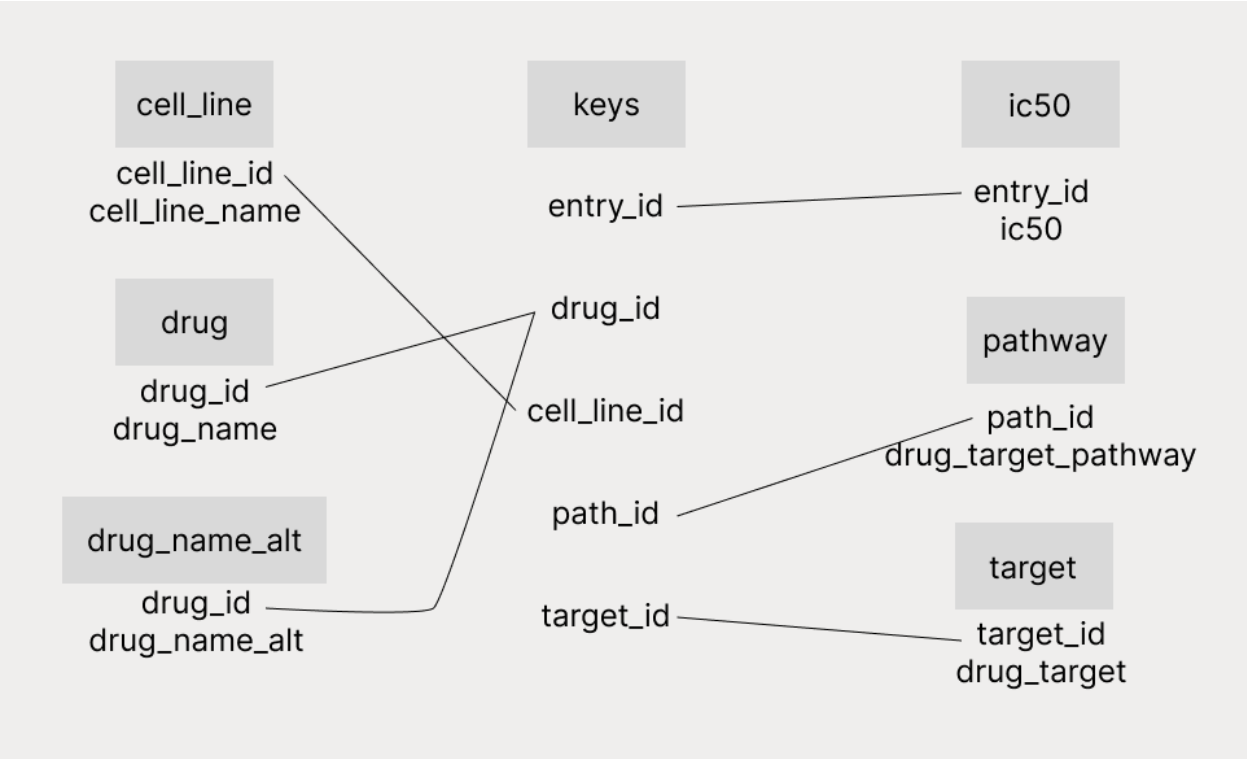

Figure 1. Data Structure for the Final.DB database. Lines indicate how variables are connected.

With that cleared up, I started working with the data. The first issue I ran into with the data had to do with the units for the IC50. I assumed that the values were already in µM, but the data was given in the natural log of µM, so to go back to the original units, I had to do e^{LN_IC50} for all the data points. To do so, I iterated over the numbers ('num') in the 'LN_IC50' column of df_final, and used the math module's exp() function to convert the num. Code is below:

```
for num in df_final['LN_IC50']:
    df_final['IC50'] = m.exp(num)
```

That issue was easy enough, but next I found that while I was loading the data, the entries were doubling up (Figure 2). The issue seemed to come from the way the `drug_ids` and `synonyms`

```
...: print(df_final)
   CELL_LINE_NAME  DRUG_NAME PUTATIVE_TARGET  \
0             ES5  Erlotinib           EGFR
1             ES5  Erlotinib           EGFR
2             ES7  Erlotinib           EGFR
3             ES7  Erlotinib           EGFR
4           EW-11  Erlotinib           EGFR
```

Figure 2. Entry double up issue.

dataframes (df) are created and merged. Specifically, the drug_ids dictionary is being populated with unique drug names from `df_drug_names['DRUG_NAME']`, but the `explode()` function is being applied to `df_synonyms`, which contains all drugs in `df_drug_names`. Therefore, if a drug name has multiple synonyms, the `explode()` function will create duplicate rows for each synonym with the same drug_id, leading to the duplication issue. So, to combat this, I ended up foregoing `df_synonyms`, and making sure `df_drug_names` and `df_GDSC1` agreed as far as drugs available. I dropped drugs from `df_drug_names` that were not in `df_GDSC1`, and then made the `drug_id` dictionary, which worked out.

I then had to add the IDs for the other variables. When I was adding IDs for the variables, everything worked out except for the format of the `target_id` was always different. For example, ID# 81 was formatted as ’81.0’ rather than the ‘081’ format I had for the other ones. I was really stuck because I was using the same exact code for all of them, but for some reason this one was different. After looking, I saw that it may be because the ‘NaN’ values are considered floats (chthonicdaemon, 2018; jpp, 2018), so I replaced the ‘NaN’ values with the ‘NaN’ string.

With the data finally loaded up and the files on the server, I began testing the files. I kept getting an After looking at the error logs in the class server, I saw that the template loader was not working as intended due to the file path. After changing from the relative file path to the absolute one, the cgi script was able to be run.

Finally, when running the cgi script, everything rendered correctly, except the values that were rendered in the template were the keys of the dictionary rather than the values. To solve this, I changed the HTML code. Prior, I had a for loop that contained all the variables I wanted, like so:

```
{% for drug_target, drug_target_pathway, ic50, cell_line in sql_query %}

        <tr>

          <td class="text">{{ drug_target }}</td>

          <td class="text">{{ drug_target_pathway }}</td>

          <td class="text">{{ ic50 }}</td>

          <td class="text">{{ cell_line }}</td>

        </tr>

        {% endfor %}
```

It turns out that this didn't specify what to print correctly, so it kept rendering the "drug_target, drug_target_pathway, ic50, cell_line" repeatedly. Then, because the sql_query was a list of dictionaries, I thought of just iteration over each dictionary and pulling out the variable I wanted, which worked out in the end:

```
{% for entry in sql_query %}

        <tr>

          <td class="text">{{ entry.drug_target }}</td>

          <td class="text">{{ entry.drug_target_pathway }}</td>

          <td class="text">{{ entry.ic50 }}</td>

          <td class="text">{{ entry.cell_line }}</td>

        </tr>

        {% endfor %}
```

To validate the CGI script, I performed test searches on the class server. I also used DB Browser to perform test queries on the SQL database itself.

Tuan Nguyen
tnguy256@jhu.edu
AS.410.712.82 - Adv. Practical Comp. Concept for Bioinformatics
Final Project – Narrative

Summary

Although I ran into many challenges for this relatively simple task, I learned a lot from this project. I like project-based learning a lot, so this was a great way to apply the skills I learned in this course and to even learn new things as I troubleshooted. This has given me a strong foundation for future learning in bioinformatics, and I am excited to take it with me into the future.

Tuan Nguyen
tnguy256@jhu.edu

References

chthonicdaemon. (2018, February 1). *Answer to "Why is NaN considered as a float?"* Stack Overflow. https://stackoverflow.com/a/48559301

DB Browser for SQLite. (n.d.). Retrieved May 11, 2023, from https://sqlitebrowser.org/

jpp. (2018, February 1). *Answer to "Why is NaN considered as a float?"* Stack Overflow. https://stackoverflow.com/a/48559277

Yang, W., Soares, J., Greninger, P., Edelman, E. J., Lightfoot, H., Forbes, S., Bindal, N., Beare, D., Smith, J. A., Thompson, I. R., Ramaswamy, S., Futreal, P. A., Haber, D. A., Stratton, M. R., Benes, C., McDermott, U., & Garnett, M. J. (2013). Genomics of Drug Sensitivity in Cancer (GDSC): A resource for therapeutic biomarker discovery in cancer cells. *Nucleic Acids Research*, *41*(D1), D955–D961. https://doi.org/10.1093/nar/gks1111