SI 507 Final Project Data Checkpoint

Project: Representing Drug-drug Interaction Using Graphs
Name: Ming-Jan (Randy) Pan

Part 1: Project Code

GitHub Link: https://github.com/Ludougan123234/SI507-Final
Heroku Web application: https://mjpan-fp507-2d393f67a958.herokuapp.com/index

All instructions to run the code are written in the README.md file, and the packages to run the code are detailed in the requirements.txt file. Both files are available on GitHub, as well as the .zip file submitted to Canvas.

Part 2: Data sources

Data source 1: NIH RxNorm API (challenge score 3)

- Format: JSON
- Access method: Based on user input on an ad-hoc basis or using JSON cache. The cache can be found in the my_app folder in the GitHub repository. The code responsible for the caching functionality can be found in the views.py get_rxnorm() function.
- Summary of data:

  According to the latest RxNorm November release notes, RxNorm currently has 584904 data records. The number of records retrieved will be based on user input. A user inputs the generic or brand name(s) of drugs. The drug names will then be searched against the RxNorm database for RxCUIs (Rx Concept Unique Identifiers, a type of machine-readable code for identifying unique drugs). Therefore, the fields of interest are the 'rxcui' and 'name' fields within the JSON response. Both of these are nested under the `drugGroup>conceptGroup>conceptProperties` field in the JSON response.

  The function that is responsible for retrieving RxCUI (`get_rxnorm()`) will only return a maximum of 50 CUI-drug name pairs at random by default because the NIH Drug Interaction API can only check for interactions between 50 drugs at a time.

Data source 2: NIH Drug Interaction API (challenge score 3)

- Format: JSON
- Access method: Based on user input on an ad-hoc basis. Caching is not implemented for this API since, in general, each request going into this API will be mostly unique. It is not practical to capture all interaction pairs.
- Summary of data: Since the record of this database is fetched from two different proprietary databases (DrugBank and ONCHigh), there is no open documentation available regarding the number of records. The number of records retrieved from this API will be based on the number of drug interactions that exist in users' queries. The fields of interest are mainly the name of the source, RxCUI, and the name of the drug, severity, and description of the interaction.

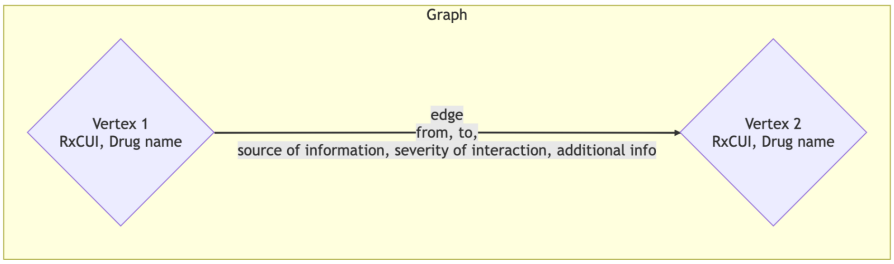
Data source 3: OpenFDA Adverse events API (challenge score 4)

- Format: JSON

- Access method: Based on user input on an ad-hoc basis. Caching is not implemented for this API since the API data is updated on a streaming basis. Users will get different results at different times of interacting with the API.
- Summary of data: This database contains 17036983 drug adverse event records on the date of writing. The number of records retrieved will be based on the drug and types of information users decide to retrieve using a checkbox form (available information includes aggregated counts of patient sex, hospitalization, age of onset, date, reporting country, and reaction type). The estimated maximum number of records retrieved per user query is 300 records (max of 6 types of information * max of 50 drugs).

Part 3: Data structure

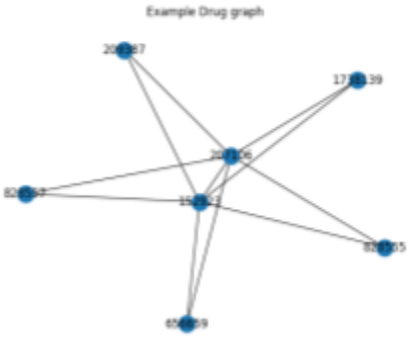
The retrieved data is represented using a graph. On a high level, the graph is modeled using a graph class, an edge class, and a vertex class. The following diagram shows the relationship of graph, vertices, and edges.

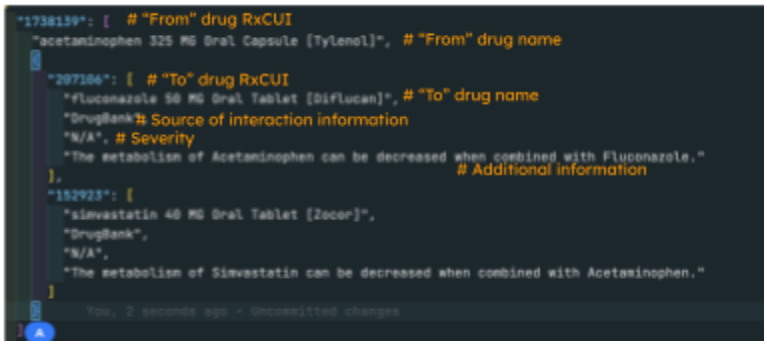


A vertex holds the RxCUI code and corresponding drug name. An edge represents an interaction between drugs and holds information like the data source, severity, and additional drug interaction information aside from the "from" vertex and the "to" vertex. The graph has a vert_list and num_vertices attribute. The edge is modeled separately from the vertex class since the edge needs to hold the severity and description of the interaction. The complete graph definition and the BFS code for the graph can be found in the graph.py file.

Using nested dictionaries and lists, the graph can be exported as a .json file. The first level of the dictionary has the "from" drug RxCUI as the key, and a list as the value. The first element of the list is the name of the "from" drug, and the second element is a nested dictionary with information about the "to" drug. The nested dictionary uses the RxCUI number of the "to" drug as its key, and the value is a list consisting of the name of the "to" drug, source of information, interaction severity, and additional information. The code that exports and reads the .json file can be found in the graph_to_json.py file. The generated .json file is in graph.json.

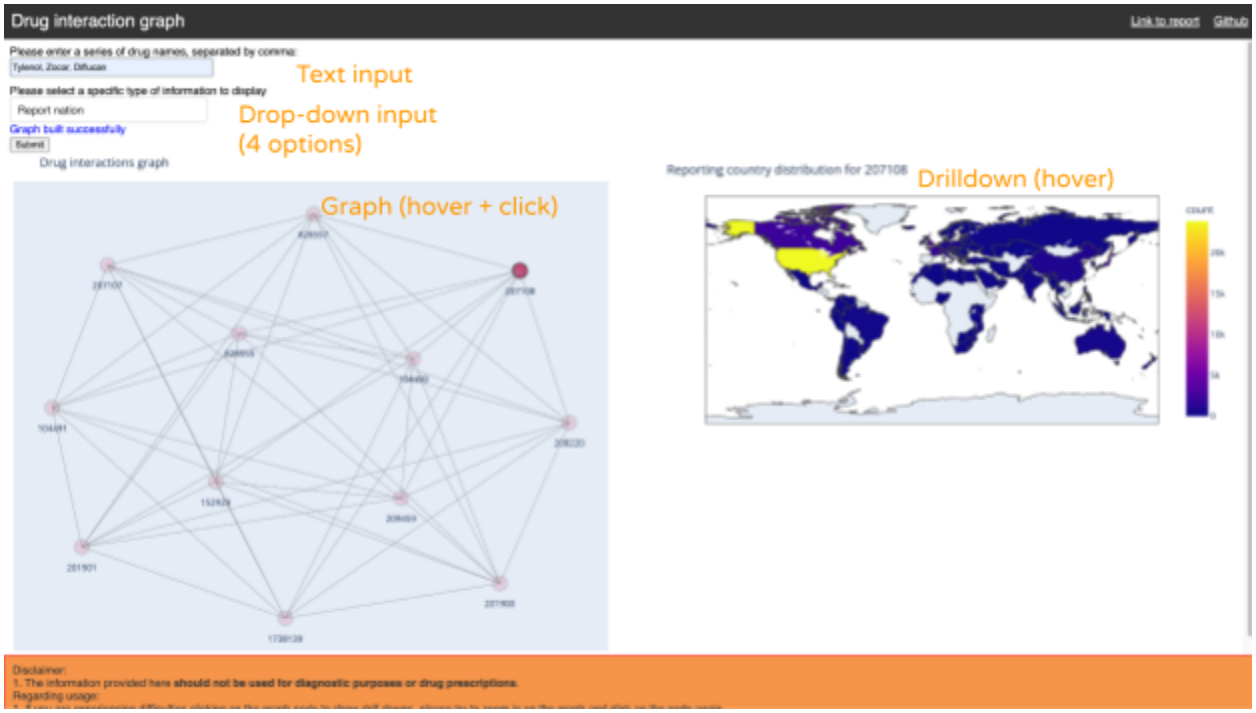
The image on the left below presents an example drug interaction graph that shows the interaction between different forms of Tylenol, Zocor, and Diflucan that are represented with the following RxCUIs: 207106, 828557, 178139, 152923, 828555, 656659, 209387. The image on the right shows a snippet of the exported .json graph, with RxCUI Code of 178139 as an example. This drug interacts with two other drugs in the graph: 207106 (Diflucan) and 152923 (Zocor). As can be seen from the graph visualization, the 1738139 node is connected to 207106 and 152923.





Part 4: Interaction and presentation

The interaction with the program is done on a web application. Users can enter a series of comma-separated drug names in an input box and select a specific type of aggregated drug adverse event data they would like to see in a drop-down menu. After submitting the input, the program outputs an interaction graph visualization. Users can hover over a vertex to see the name of the drug, the number of connections, and the estimated average shortest path from the vertex to other vertices. Users can also hover over an edge to see the drug interaction information. After a graph has been successfully generated, users can click on a graph vertex to inspect the specific type of adverse event data associated with the drug. The available options are: patient sex distribution, age of onset, reporting nation, and types of adverse events. These options are visualized using bar graphs or choropleth mapping. The screenshot below shows the interface of the application.



Part 5: Demo video

Narrated demo:
https://drive.google.com/file/d/1myPXo2QW8KKa7ZzJcoZqcfmvESec-3qK/view?usp=sharing

Short demo (unnarrated):
https://drive.google.com/file/d/1OWcb8oovu2z5seZLCm__oaMao2jOr3b7/view?usp=sharing