Documentation

Github link:
https://github.com/Lycbel/410ProjectFinal/tree/master/Web/cs410
video link: https://mediaspace.illinois.edu/media/t/1_5ctl3eym

There is size limitation of Github, so there are two missing files:
FDA_processed.dict and word_matrix.dict, we can provide it if needed.

1. Project Overview

In our project, we built a knowledge graph for drugs and their adverse reactions that have been reported. The knowledge graph will explain the possible reasons why a drug causes a side effect.

The knowledge graph contains four types of nodes: drug, adverse reactions, medical subject headings retrieved from the MeSH database, GO terms database. We extract the reported drug-adverse reaction pair from FDA database, and mine knowledge for each edge from PubMed literature database. Finally, we use the networks to give a possible explanation of the relation between drugs and adverse reactions.

We also built a web application to provide a visualization and better interactivity with users. It takes drug and adverse reaction pairs as input and returns a knowledge graph contains interconnected nodes, which stand for medical terms, to explain the potential relationship between the drug and the adverse reaction. Hopefully, this project will be a useful tool for patient, doctors and drug manufacturers for a quick reference.

2. Project Implementation

2.1 Prepare datasets

In-depth investigation has been done and several representative datasets was chosen for construct nodes in the following part. Our team crawled and formatted datasets from Go database, PubMed database, FDA Adverse Event Reporting System (FAERS) database and Medical Subject Headings(MeSH) database.

GO database: Because of the staggering complexity of biological systems and the ever-increasing size of datasets to analyze, biomedical research is

becoming increasingly dependent on knowledge stored in computable form. The Gene Ontology (GO) project provides the most comprehensive resource currently available for computable knowledge regarding the functions of genes and gene products. In our project, we extract all the 47131 Gene Ontology (GO) terms, which provides the logical structure of the biological functions ('terms') and their relationships to one another, manifested as a directed acyclic graph

PubMed database: PubMed comprises more than 28 million citations for biomedical literature from MEDLINE, life science journals, and online books. Citations may include links to full-text content from PubMed Central and publisher web sites. In our project, we extract about 16731200 literature titles and their abstract.

FDA Adverse Event Reporting System (FAERS) database: Raw patient case report from one quarter in xml format, roughly 350,000 cases for each quarter. The drugs and the reactions under the same patient case may have potential connections, which remain to be verified. The raw xml data can be downloaded from FDA website directly. Then we process data and retrieve needed information. We extracted the drugs, active substance, reactions from each case reports; organize them into key value pairs and count the number of every key respectively. We processed reports published from 2014q2 to 2017q4.

Medical Subject Headings (MeSH) database: MeSH is a comprehensive controlled vocabulary for the purpose of indexing journal articles and books in the life sciences; it serves as a thesaurus that facilitates searching. We use the mesh 2018 data which is in xml pattern. "desc2018.xml" consisting of Descriptor_name, Descriptor_ID, Descriptor's terms and Qualifiers. "Pa2018" is in the similar pattern but contains Pharmacologic Action information. We extract the descriptor name and its ID in one file 'descrip.txt'; terms name and its ID in another file 'pa.txt'; qualifier name and ID in one file 'qa.txt'; and PA file's descriptor name and its ID in another file 'pa.txt'. There is 28939 records in 'descrip.txt'; 6669 in 'pa.txt'; 79 in 'qa.txt'; 116706 records in 'term.txt'.

2.2 Data processing

We processed data retrieved from datasets mentioned above by extracting data and make them into python dictionaries. Dictionary is an ideal data structure because it has O(1) access time. Two main dictionaries are made for the creation of the knowledge graph. One is the dictionary for

all the medical terms (word_dict), including MESH terms, GO terms, drugs, adverse reactions and active substance of drugs. The keys of this dictionary are medical terms and the values are indexes starting from 1. Indexes are used as indexes in the relation matrix, which will be discussed further in the next section. The other is the dictionary for all the medical terms appeared in literatures (doc_dict). The key is literature ID and the value is a list of all medical terms appeared in the literature.

Moreover, when dealing with FDA database, we find a lot of Drug-ADR pair are only reported little like once. We think these pairs may be reported by some mistake. Thus for each drug, we remove the drug-ADR pair which the reported cases is in bottom 5% of the distribution.

Result:
In total, for inner node in the knowledge graph, we have 47131 Go term and Go term id; 35608 Mesh Heading term;
For drug and ADR information, we have 63707 drugs; 13544 Adverse drug reactions; 6479 active substance; and 3507300 drug-ADR pairs.
For paper information, we have abstracts of over 160 million literature.

2.3 Build knowledge graph

Having the two main dictionaries, we are able to build a matrix indicating relationship between one medical term and another. We call it “Relation matrix”. If two medical terms appear in the same literature, we consider that they relate to each other and we add 1 to RelationalMatrix[index of term1][index of term2], where indexes can be easily accessed with the word_dict. After processing all the documents in doc_dict, we have a huge relation matrix which x-axis and y-axis are medical terms and values are weights. A specific value w = RelationalMatrix[index of term1][index of term2] in the matrix means that term1 and term2 are simultaneously mentioned in w literatures. The relational matrix is huge, roughly 150,000 * 150,000. So we use sparse matrix instead of nested list to save memory and disk space. The matrix generation program requires a computer with more than 30 gigabytes of memory. The final matrix has a total size of 6 gigabytes.

With the relation matrix, we are able to search for paths which link drug and adverse reaction together. We choose IDDFS (Iterative Deepening Depth First Search) for this problem. First of all, this is a totally uninformed search problem with no possible heuristics. Secondly, we consider that long paths are not as reliable and persuasive as short paths. With IDDFS, we can control the preferred length of paths very easily.

Our IDDFS takes a drug and an adverse reaction as input and returns all the paths found before the pre-defined depth.

Graph score indicating the reliabililty. Every path will have a score based on the mean weight of this path. In addition, for the hub node we will lower the connect weight to punish it. finally we will filter out the paths with low score and only keep the top 5.

Then we have functions that convert paths to json file for the visualization.

2.4 Visualize results with website

It is based on PHP and MYSQL as website backend. HTML, JS and CSS as front end.
Database there are two tables in database cs410: user and history. to create the database and table:

Create database cs410;

```
CREATE TABLE `user` (  `name` char(100) DEFAULT NULL,
`pass` char(100) DEFAULT NULL,  `email` char(100) DEFAULT
NULL ) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE `history` (  `email` varchar(100) DEFAULT NULL,
`drug` varchar(200) DEFAULT NULL,  `adr` varchar(200) DEFAULT
NULL,  `timeS` timestamp NOT NULL DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP )
ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

For the website:
It has Auth.php, search.php, vis.php, login.php, main.php,history.php.

Auth.php handles the login and register.
main.php is the main page of search interface which provides the functions of search, view history, log out, and visualization frame update.

For search:
Main only uses ajax to perform search and view history. Upon searching, the request sends to search.php.

search.php is the logic page for search ajax request, it returns the status of

search procedure which use the python script to perform search and return the result. The Python script gets the drug and adr from the search.php. It checks if the drug and adr are medical terms in our dataset. If not, it will just return -1. Otherwise, it performs the real search on the relation matrix which contains all the connections. The resulted json files are stored in the "/data" folder.

For visualization:
Json file is used as the input of the Alchemy visualization tool. It is a convenient JavaScript tool for network graph visualization. It is in the main.php. After user searches and gets the result, the ajax will return an 'OK' status. Then src for the Iframe whose src will be changed to direct the Iframe to a new resource which will be vis.php?newSource(the created json file), then the iframe will be reloaded and show the visualization of the search result.

For search history:
The history functionality is based on history.php. it is also a page for ajax requests. It returns all the recent 30 searches by the user. And JavaScript will use the returned value to update all the corresponding elements which are all in the table with id ('#hisTable').

3. Software Usage

Users need to login or register to use the service. After logging in, users can type in drug and ADR (adverse reaction) under the corresponding text field and press enter. After the search is finished, the resulted knowledge graph will be shown on the visualization frame. To view history user just click the history button and clicking the item will show the visualization.

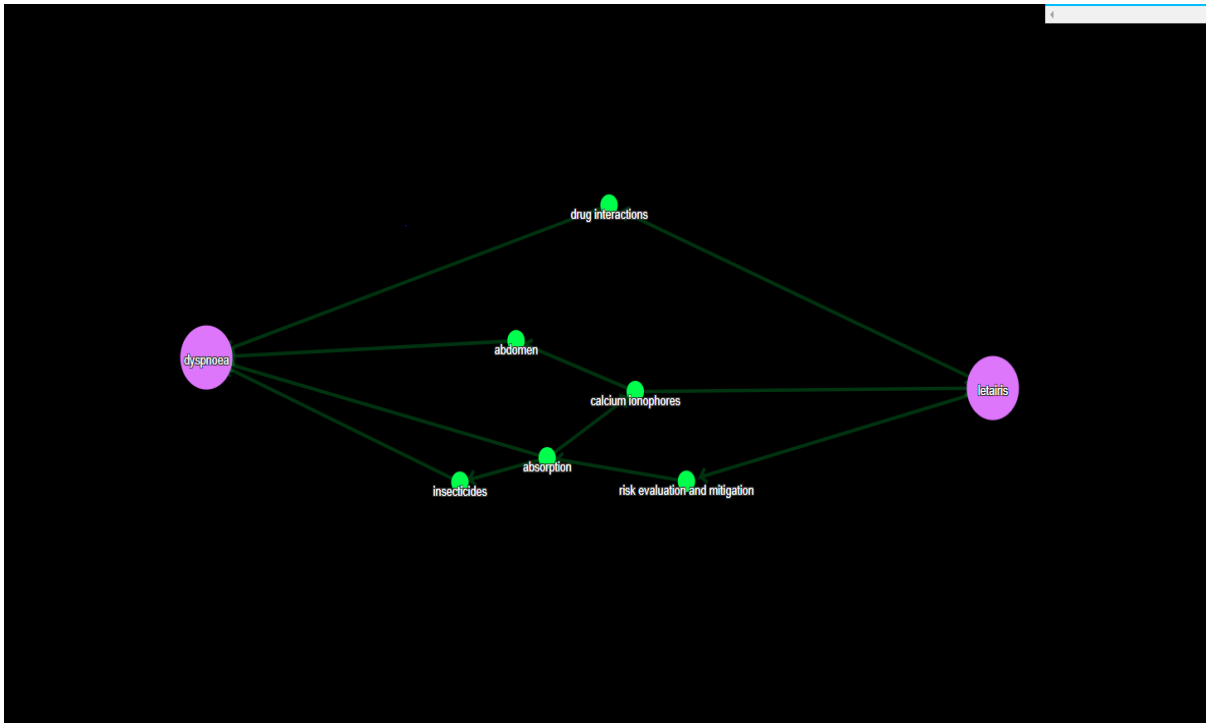
API:
generateGraph.py:

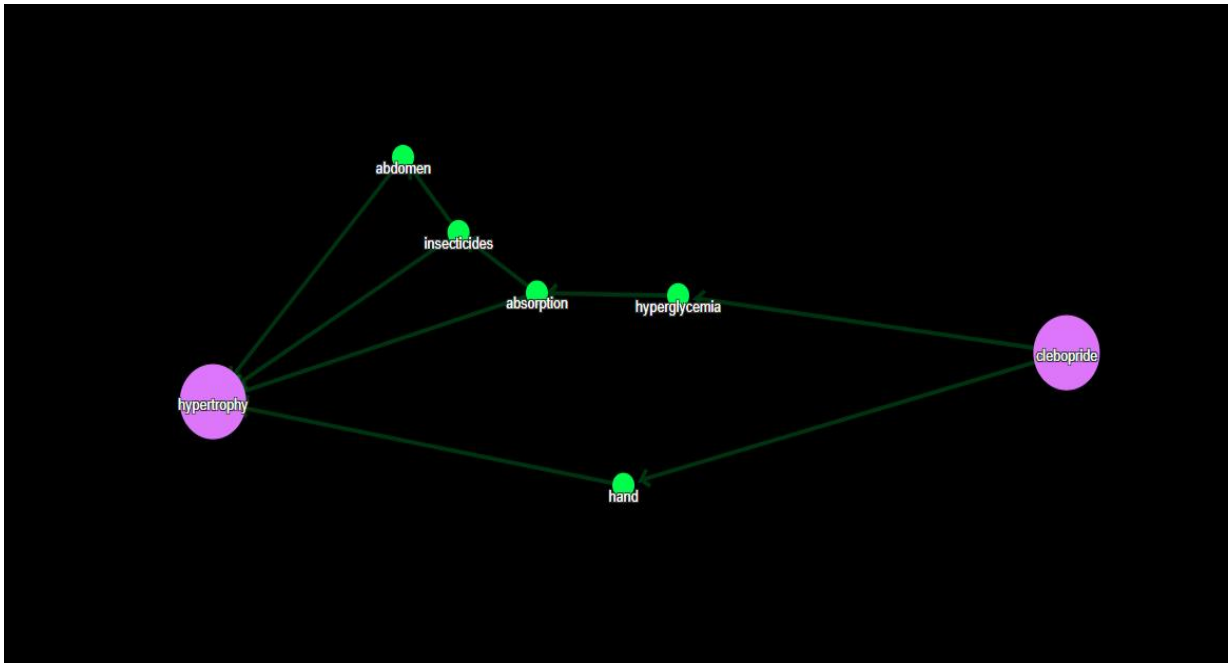
check(drug, adr): Returns true if both drug and adr are medical terms in our dataset. Otherwise, false.

IDDFS.py:

webAppSearch(drug, adr, path): Returns true if one or more path is found, otherwise false. If one or more path is found, the function will write the resulted json file to path and print all the paths to console.

The sample output:





4. Team

Team Leader:
Huamin Zhang

Team Member:
Shijie Guo, Sixuan Shen, Yichang Liang