DS4300
Final Project Report

Ben Kosiborod, Bryce Russell-Benoit, Maris McGuinness,
Deepak Ramesh Sethuraman, Genevieve Jawor

Neo4j Diseases, Medications, and Side Effects

Overview

This project was a great way for us to show our database skills that we've acquired over the course of the semester as well as mixing in a biological component that could be used to benefit other people. We chose this project because we were interested in looking at drug interactions and wanted to create something that could have a variety of different applications and could be used by both patients and health professionals.

We sourced our data from the Stanford Biomedical Network Dataset. Similar to the gad dataset we worked on in class, these datasets explore relationships between drugs, genes, diseases, proteins, and much more. From this collection of data we chose 3 datasets to combine: DCh-Miner which explores diseases and drug relationships, ChSe-Decagon which documents drugs' side effects, and ChChSe-Decagon that contains drug interactions and their side effects. The combination of these 3 datasets creates a robust network of diseases, medications, and possible side effects from single or multiple medications.

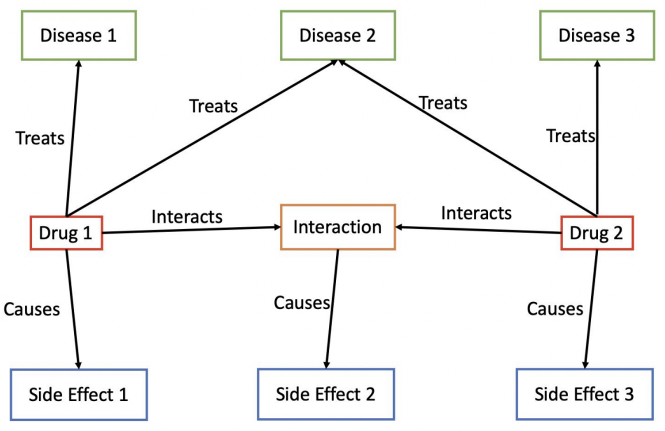
This project allows insights to drug regimens, and allow for several different conclusions to be drawn including: side effects of a drug, given a list of drugs - find all side effects and combined effects, side effects of adding a new drug to a patient's regimen, given a side effect deduce which drug may be causing it, and given a disease find the combination that will lead to the least amount of side effects. Overall this project serves as a knowledge base for healthcare professionals to consult for possible occurring side effects of a large database of drugs and diseases. This project could be particularly helpful for health care professionals that may have forgotten the possible side effects of a drug or from drug interactions, or healthcare professionals who are seeing a new patient and need to get an idea of which drug is appropriate to prescribe, based on only their current drug regimen. Overall, this project will help them treat patients in the most effective way possible.

Technical Design

Cleaning the Data

Our raw data consists of 3 CSV files: one mapping drug to diseases (~466K rows), one mapping drug interactions to side effects (~4M+ rows), and lastly one mapping drugs to side effects (~175K rows). An issue we faced off the bat in our cleaning process was that our files only contained disease/drug name IDs, with some of the IDs coming from varying namespaces, such as PubChem, NCBI's MeSH database, and other NCBI-related namespaces outside of those two. We decided to use Python and PubChemPy, as well as querying traditional RESTful APIs (such as MeSH's) to map drug and disease names to the IDs in our data (luckily, side effect names were already provided in our data, and used the PubChem namespace). In our first Python script, we read in our data into pandas DataFrames, found the unique PubMed drug/compound IDs from our datasets, and queried PubMed for the names of each unique drug/compound ID. Then, we mapped these names to each row in our dataset to create a 'name' column in each of our DataFrames for each ID present in the data (drug IDs and disease IDs).

Designing the Database



Our data was easily translatable to a graph database, as the data actually came from a graph-oriented dataset. We created nodes for each disease, drug, and side effect. One thing to note was the importance of creating a node for a drug 'Interaction' to distinguish the side effect of a drug from the side effect of a drug interaction. For example, two drugs pointing to the same side effect are not necessarily interacting with each other, which is an important distinction in this database.

Importing the Data

Initially, we explored using Cypher to import our raw CSV data using Cypher's LOAD CSV query. However, with the amount of data that we were dealing with, it was unfeasible to use this function of Neo4J, as it would take too long to import all of our data. So, we created another Python script that takes our cleaned & name-populated data and converts it into the format desired for neo4j-admin's import tool. Finally, we also created a shell script that would run the import tool for us safely. Setup instructions that explain the steps for cleaning the data, as well as importing the data and getting a local Neo4J up-and-running with our data is included in our source code, in the setup folder's README.

Database Queries

1. What are the side effects of a given drug?
    - This query has a fairly straightforward approach of matching drug nodes matching the given name and returning the connected side effects.
2. Find all side effects of a given list of drugs
    - This query starts similar to the previous one but uses a list of drug nodes instead of a single drug. After we have all those side effects we do a subquery to check for interactions between the drugs.

Ben Kosiborod, Bryce Russell-Benoit, Maris McGuinness,
Deepak Ramesh Sethuraman, Genevieve Jawor

3. What are the NEW side effects resulting from adding a drug to a patient's current drug *regimen* (represented as a list of drugs)?
    - This query again starts similar to the previous query and finds all side effects of the given drugs and the interactions between them. It then builds on this by taking a new drug and finding its side effects and the side effects of any possible interactions between this new drug and the list of drugs from the first part of the query. It compares these side effects to those from the first part and returns only the unique side effects.
4. Given a side effect, find what drugs or drug combinations may be causing this side effect
    - For this query we start at the side effect nodes and search for individual drugs that cause those side effects and collect the result. Then we Look for interactions that cause that side effect and collect the drugs in that interaction. Then we combine those collections and return the result.
5. Given a drug *regimen* (list of drugs) and a newly diagnosed disease, find the drug that treats that disease with the lowest number of new side effects.
    - This query starts by getting all the side effects of a patient's current drug regimen. Next, we find the list of drugs that treat the given patient's new disease. Next, we find the side effects that are caused by each potential drug (potential drug meaning a drug that can treat the given disease), including the side effects of that are caused by the interaction between a potential new drug and any of the drugs in the patient's current regimen. Finally, we count how many new side effects are introduced by each potential drug and return the one with the lowest number of new side effects.

Website
The website was created in HTML to simplify and improve the user experience of querying our database. The website ties together the project and serves as a good platform for expanding upon the project in future use. Potential further expansion includes building a scraping tool to gather web data to include a larger list of diseases, drugs, and side effects (individual and compound). Expanding this scope enables more connections to be made, and could serve as a useful tool to medical researchers. Another future exploration would be utilizing Kubernetes to automate scaling and batch processing to cut down on the time it takes to organize and process our data.

Group Member Contributions
- **Cleaning the Data** - *Genny, Maris, Ben*
- **Designing the Database & Importing the Data** - *Ben*

- Querying the Data - *Ben, Bryce*
- Creating the Website - *Deepak*
- Proposal - *All team members*
- Final Presentation - *All team members*
- Final Paper - *All team members*