Exploring Drug Interactions

Ben Kosiborod, Bryce Russell-Benoit, Maris McGuinness, Deepak Ramesh Sethuraman, Genevieve Jawor
Group 16

Project Overview

Problem Statement

- Knew we wanted to do a project at the intersection of medicine and data science
  - *Where to start?* → **Data!**
- Stanford Biomedical Network Data Collection
  - *(Drug/Disease/Side Effects)*
- **Why is this important?** → Tool for medical professionals & patients to understand the interactions and side effects associated with the use of different drugs to treat disease

Possible Applications

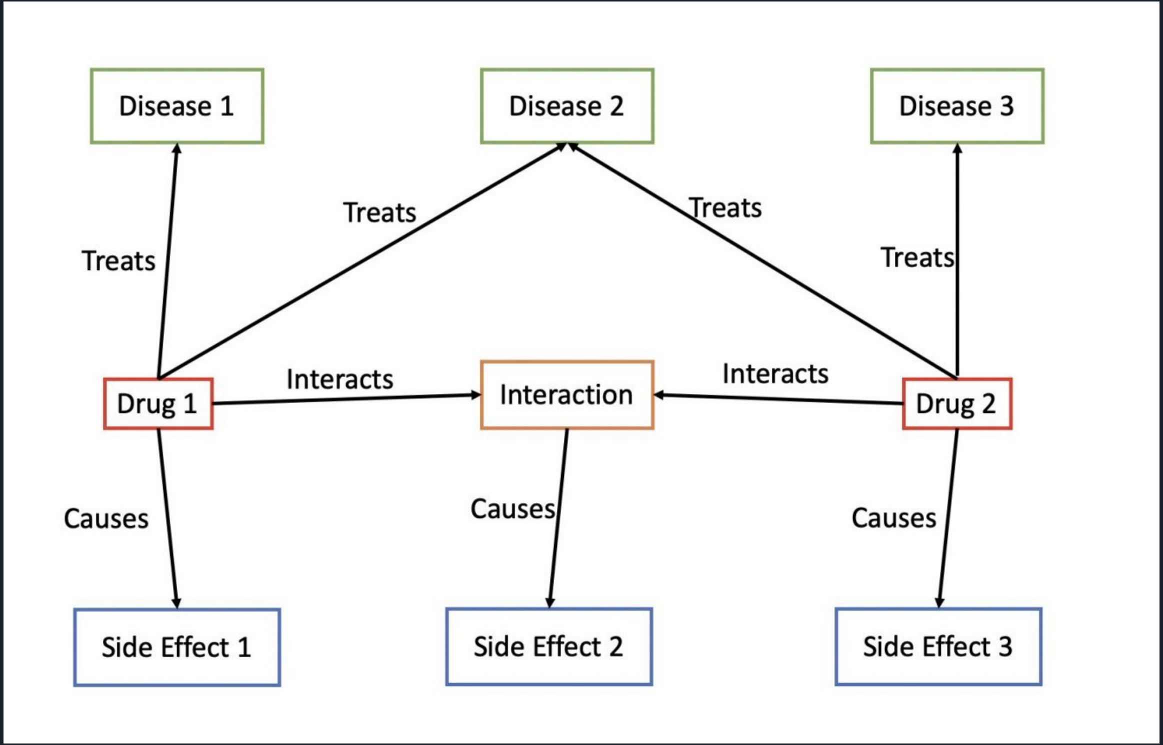
- Side effects of a drug
- Given a list of drugs, find all the side effects
- Side effects of adding a new drug to a patient’s drug regimen
- Given a side effect, tell me which drugs (or combinations of drugs) may be causing that side effect
- Given a disease, find what new drug for that disease that will cause the least number of side effects, given the drugs you’re already taking

Data Exploration

Cleaning the Data

- Raw Data (3 CSVs)
  - Drug → Disease (~466K rows)
  - Drug → Drug → Side Effect (~4M+ rows!)
  - Drug → Side Effect (~175K rows)
- Challenges:
  - Mixing and matching IDs from different databases (MeSH, NCBI, PubChem)
  - Trial-and-error with RESTful API responses
  - Querying without timing out
  - A LOT of data!!

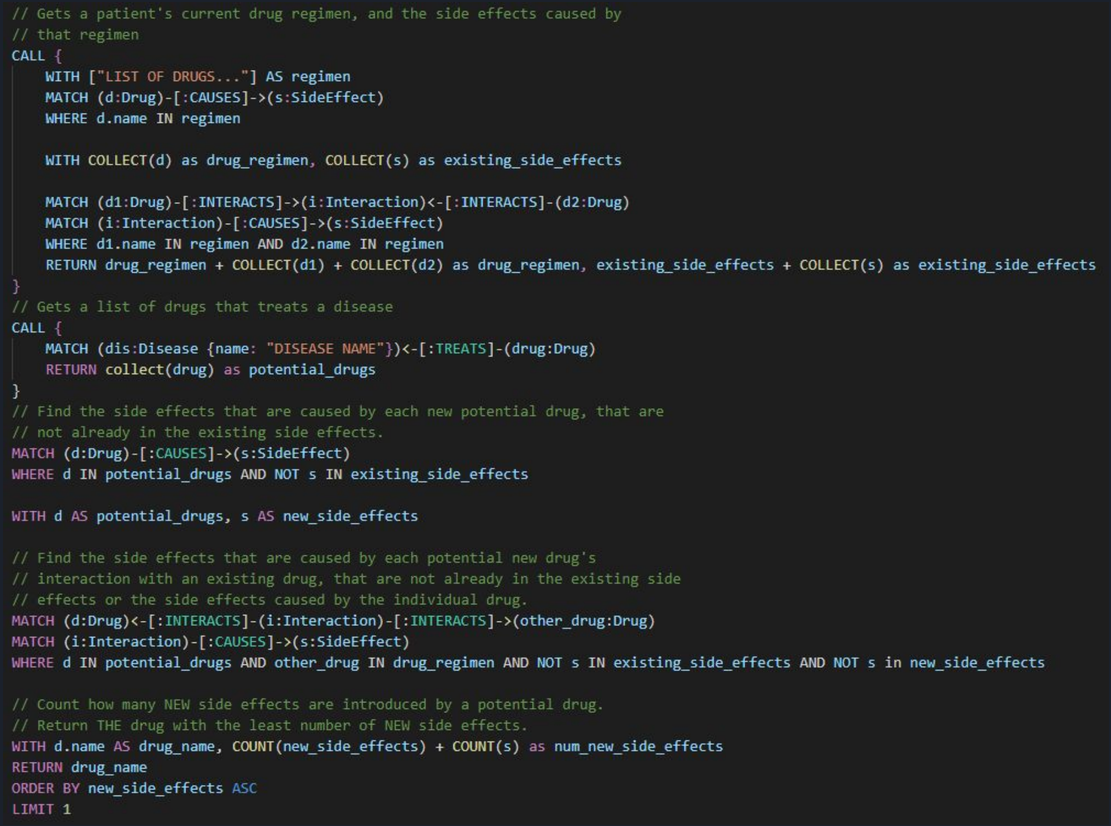
Designing our Database



Querying the Database (Cypher)

Example:

Least new side effects when adding a new drug to treat a given disease



```
// Gets a patient's current drug regimen, and the side effects caused by
// that regimen
CALL {
    WITH ["LIST OF DRUGS..."] AS regimen
    MATCH (d:Drug)-[:CAUSES]->(s:SideEffect)
    WHERE d.name IN regimen

    WITH COLLECT(d) as drug_regimen, COLLECT(s) as existing_side_effects

    MATCH (d1:Drug)-[:INTERACTS]->(i:Interaction)<-[:INTERACTS]-(d2:Drug)
    MATCH (i:Interaction)-[:CAUSES]->(s:SideEffect)
    WHERE d1.name IN regimen AND d2.name IN regimen
    RETURN drug_regimen + COLLECT(d1) + COLLECT(d2) as drug_regimen, existing_side_effects + COLLECT(s) as existing_side_effects
}
// Gets a list of drugs that treats a disease
CALL {
    MATCH (dis:Disease {name: "DISEASE NAME"})<-[:TREATS]-(drug:Drug)
    RETURN collect(drug) as potential_drugs
}
// Find the side effects that are caused by each new potential drug, that are
// not already in the existing side effects.
MATCH (d:Drug)-[:CAUSES]->(s:SideEffect)
WHERE d IN potential_drugs AND NOT s IN existing_side_effects

WITH d AS potential_drugs, s AS new_side_effects

// Find the side effects that are caused by each potential new drug's
// interaction with an existing drug, that are not already in the existing side
// effects or the side effects caused by the individual drug.
MATCH (d:Drug)<-[:INTERACTS]-(i:Interaction)-[:INTERACTS]->(other_drug:Drug)
MATCH (i:Interaction)-[:CAUSES]->(s:SideEffect)
WHERE d IN potential_drugs AND other_drug IN drug_regimen AND NOT s IN existing_side_effects AND NOT s in new_side_effects

// Count how many NEW side effects are introduced by a potential drug.
// Return THE drug with the least number of NEW side effects.
WITH d.name AS drug_name, COUNT(new_side_effects) + COUNT(s) as num_new_side_effects
RETURN drug_name
ORDER BY new_side_effects ASC
LIMIT 1
```

Finishing the Project

- Creating a Web App
- Process all the data (since processing was taking too much time)
- Technologies being used
  - Python Flask
  - Neo4j library (from neo4j import GraphDatabase)

Flask
web development,
one drop at a time

Building a Continuous Integration Model

(outside the scope of this class)

- Building a Scraping tool that scrapes the web for data
- Using Batch Processing to process the data
- Using Kubernetes

Any Questions?