**Graph Databases for Patient Analysis**

Pranav Anthapu

Middlesex County Academy for Science, Mathematics, and Engineering Technologies

**Assistance received**

**Dr. Latha Nair: Discussed Idea, Helped me brainstorm experimental strategies**

**Abstract**:

Graphs can be used to analyze connected information. With disease vectors increasing, it becomes imperative to analyze the Patient diagnosis and response to medicine holistically. Graph databases can provide a better solution to this problem efficiently.

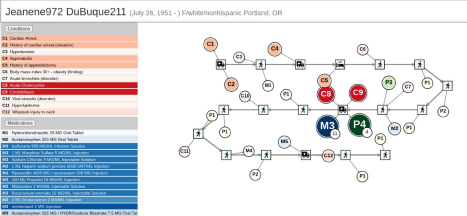
**Introduction:**

This process can provide insights into how a timely diagnosis and treatment can lead to better outcomes and identify the hindrances for such outcomes. A machine learning model can be trained to predict future outcomes based on existing patient data.

**Methodology**:

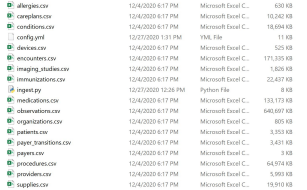
If one were to feel the symptoms of common cold or COVID-19, they would immediately plan to visit the doctor. At the doctor’s office, they would probably have multiple visits, at multiple times in their treatment, with some visits having the doctor prescribe a certain medication. To sort through these numerous visits for an enormous number of people would be challenging , as there are almost 9 million people in New Jersey itself. In order to find connections between certain patients, one could use a graph to make these connections. Graphs, to put it simply, are nothing more than different nodes (or components) that are connected with other nodes through different relationships. Some nodes may have multiple relationships to other nodes, which increases the complexity of graphs. Using graphs to analyze patterns in diseases is straightforward, once you have everything in the database. The Neo4J graph software provides an user-friendly interface, as well as easy commands to use and is supported by their extensive documentation.

Considering each patient’s journey from diagnosis to the outcome as a graph, the process can build a relationship using multiple graphs. These graphs are connected by various factors like age, gender and race, etc. In order to start this project, I needed some patient data to work with. Using data from doctors and other medical companies was not a feasible solution, as that data would be personal. In order to resolve this challenge , I used Synthea, which simulates statistically significant data. [1]

*Figure 1 - Sample Person's medical history as generated by Synthea*

Using a sample that I created using Synthea, I loaded the data using data in CSV(Comma Separated Value) format into the graph, such as in Figure 2. This data would not be useful yet, however, as it was just a text file. In order to make this data useful, it would need to be loaded into the graph which is another step in the process, using programming in Python and Cypher, the official language for Neo4J software. Below are the list of files that are used in the process.

**Graph Databases for Patient Analysis**

****

*Figure 2 - CSV files generated by Synthea simulation*

After using Synthea to simulate the data, I used the tool pyingest, a python program that can read through large amounts of data and a configuration file with Cypher to load the data into the graph itself.

**Results:**

The research was to find the patients that have allergies and how they have been cured or not. The analysis included external factors like insurance, ethnicity, etc. When the simulation and the loading were completed, I looked over sample commands, such as the one shown below.

**Sample Command:**

MATCH (n:Patient)-[:HAS\_ENCOUNTER]->(e:Encounter)-[:HAS\_ALLERGY]->(a:Allergy), (e)-[:HAS\_PAYER]->(p:Payer)

WITH n,

size((e{isEnd: false})-[:HAS\_END]->()-[:HAS\_ALLERGY]->(a)) as hasEnd, p.name as PayerName

RETURN hasEnd, PayerName, count(n) as count

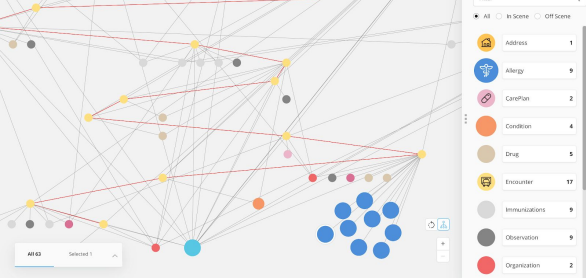
order by hasEnd DESC, count DESC

Essentially, this command finds all encounters(doctor visits or hospital visits), matches them with certain allergies and the payers that paid for that visit, and counts the number of patients that had those allergies and payers. In this specific example, all payers are listed, as well as if the specific visit reached an end to the allergy, as well as the number of cases for this payer. Some results for that command are shown in Table 1, where hasEnd signifies if the allergy had an end, PayerName is the name of the payer, and the count is the number of patients that applies to that payer/allergy end. If hasEnd has a value of 1, the patient is cured of that allergy, whereas 0 represents a situation where the allergy is not treated completely.

| **hasEnd** | **PayerName** | **Count** |
| --- | --- | --- |
| 1 | Humana | 114 |
| 1 | NO\_INSURANCE | 96 |
| 1 | Aetna | 78 |
| 1 | Blue Cross Blue  Shield | 77 |
| 1 | UnitedHealthcare | 76 |
| 1 | Cigna Health | 67 |
| 1 | Medicaid | 55 |
| 0 | NO\_INSURANCE | 1017 |
| 0 | Blue Cross Blue  Shield | 737 |

*Table 1 - Sample command and results from the command*

It is also possible to factor in various ethnicities and match patients/conditions based on ethnicities. For the previous example, it is possible to show the ethnicity in the results. As of this report, the ethnicities listed are only between “nonhispanic” and “Hispanic”. Using the sample data, it is shown to have less Hispanic patients overall, as well as more patients having no end to their allergies rather than an end.

After using the commands, I used an app from Neo4J called “Bloom”, which makes the graph look more aesthetic and also can focus on one person. Using this, I got Figure 3.  *Figure 3 - Graph using Bloom* 

Using Bloom made it clearer how many allergies this patient was diagnosed with and

what caused the death. With the allergies as the third to last encounter, it would make an

intelligent conclusion to claim that allergies had a greater impact on the patient’s death.

**Discussion and Conclusions:**

As this is not real data, the data simulated wouldn’t be accurate to determine patterns in

the real world. However, with this data, multiple patterns can be shown, such as what the

deadliest virus or deadliest allergy is, and to answer questions like “Which medication should be prescribed for this condition?”. Not every question can be answered solely through this, but this method makes it much easier to determine patterns between conditions, and can show similarities between 2 conditions. This method also makes it easier to prescribe medicines. For example, if multiple patients have the same, or similar, diseases in the same order, a doctor could more easily prescribe a medication based on this pattern.

**Related Work:**

Anything that has patterns can be analyzed using graphs. There have already been multiple uses of graphs in people’s lives, such as in software for tools like Google, Netflix, etc. One specific example of graphs in Life Sciences already exists. [2] To summarize, a doctor from Germany’s version of the CDC struggled to find a connection between DNA sequence data and metabolomics data. When he failed, he turned to graph databases and graph software from Neo4J, which enabled him to finally make this connection.

**Future work:**

Using graph databases to examine the payers people choose is a factor, to be considered in the treatment itself. This can be extended to multiple different scenarios, such as the drug or medication given at a certain visit, or which medication was given for a certain condition, and how that affected the patient. Finding out how a certain medication affects a certain patient would be extremely useful, and can be done more easily through graph databases than any other manual method, such as looking through multiple papers to find the medication name and the effects. With the data I currently have, it would probably not be possible to carry out any actual research into this field, but this can be expanded for any purpose, such as finding out what policies about COVID people like or dislike more in a certain county. Graph Databases, although a recent development, can definitely make organizing and finding patterns in multiple fields, including studies about diseases, much easier.

**Works Cited:**

[1] Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record | Journal of the American Medical Informatics Association | Oxford Academic (oup.com) (Walonski, Jason et al. March 2018)

[2] Why the German Centre for Diabetes Research changed its approach to handling data - Pharma Technology Focus | Issue 76 | November 2018 (nridigital.com)

Neo4J Graph Database Version 4.0.1 [Computer Software] (2020). Retrieved from Neo4j Graph Platform