Nicky Cen, Calvin Quach
Professor He
CSCI 493.71
06 April 2018

Project 1

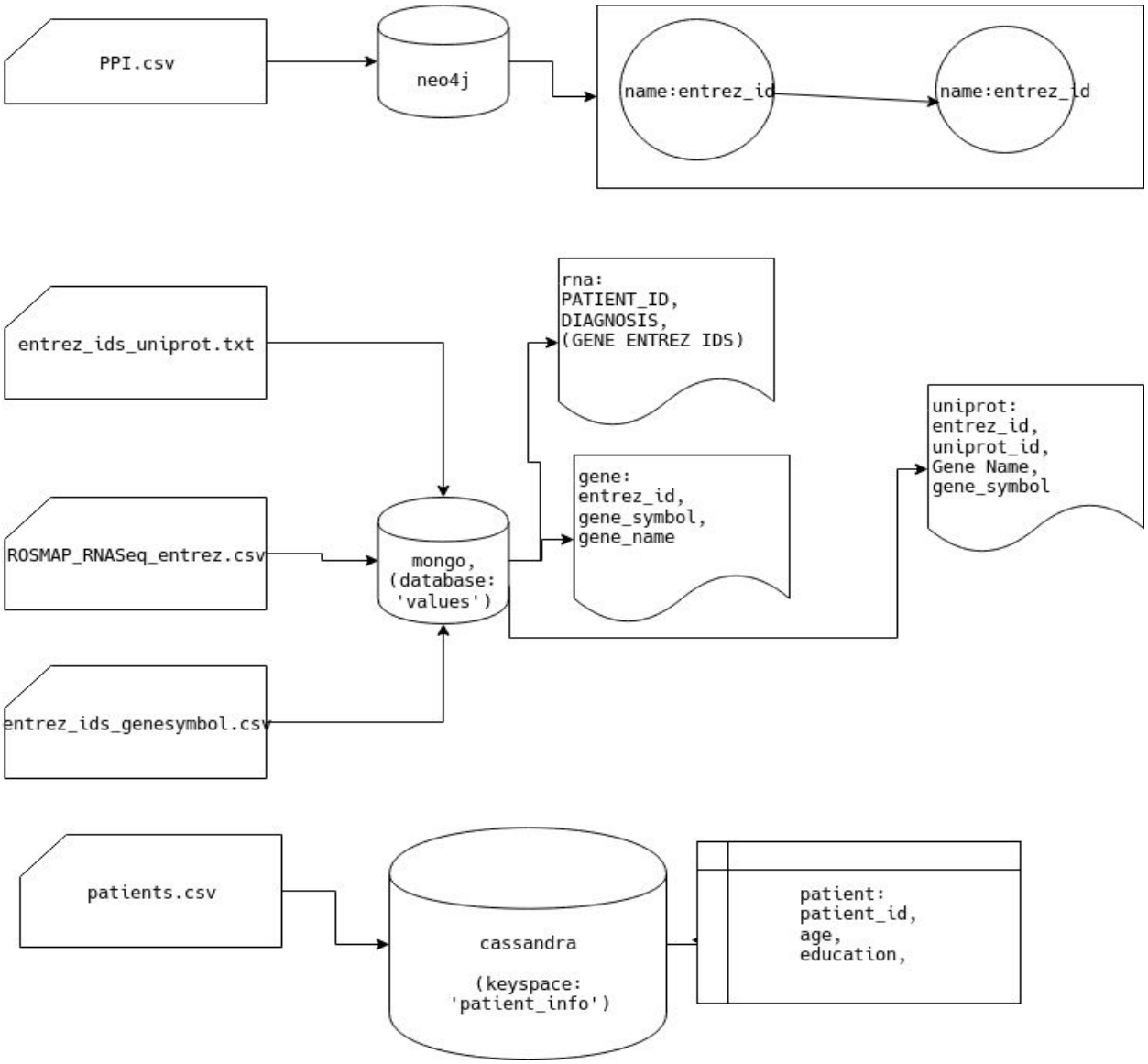
Introduction

In this project we were provided with gene, protein, and patient data sources. Using these data sources we must be able to provide a data storage system to will allow users to perform read and write operations to these system. Along with transferring the data into database, we also designed a interface and CLI for users to perform queries to these databases.

Some questions that our interface must be able to answer are:

1. Given a gene, find all n-order interacting genes
2. Given a gene, find mean and std of gene expression values for AD/MCI/NCI, respectively
3. Given a gene, find all other information associated with this gene
4. Given a patient id, find all patient information (age, gender, education, etc)

Data flow Diagram



The diagram displayed how the file information is uploaded into the knowledge base system and how they the data is represented.

Queries Used/Algorithms

1. Given a gene, find all n-order interacting genes

To answer this question we need analyze the data in the PPI.csv file. The data from this file was uploaded into a Neo4J database because Neo4J is optimized and designed to represent graph/relationship structures. Since the PPI file contained information about gene relationships we

thought it would be appropriate to upload the information into Neo4J graph database. To retrieve the gene symbols from the id results, we query the mongo collection containing the gene information.

Loop: from 1 to order {
//If the end node matches the gene, return the start node
MATCH (s:interactor)-[r:INTERACTS_WITH*<order>]->(e:interactor) WHERE s.name = '<entrez_id>'
RETURN e;
//If the start ndoe matches the gene, return the end node
MATCH (s:interactor)-[r:INTERACTS_WITH*<order>]->(e:interactor) WHERE e.name = '<entrez_id>'
RETURN s;
}

2. Given a gene, find mean and std of gene expression values for AD/MCI/NCI, respectively

To answer this question we need to analyze the data in ROSMAP_RNASeq_entrez.csv. Each record in this data file contained expression values for a gene. This data file was uploaded into a mongo database. We felt this was the best way to calculate mean and std because mongo provides built in aggregate functions to calculate mean and standard deviation in documents.

```
//Query for MCI,AD (because they have two labels associated with them, their labels are stored in a tuple
db.rna.aggregate([
        {
                $match: {
                        $or: [
                                "DIAGNOSIS": <tuple_arg1>,
                                "DIAGNOSIS": <tuple_arg2>
                        ]
                }
        },
        {
                $group: {
                        '_id': value
                        'Avg_entrez': { $avg : $<entrez_id> },
                        'Std_entrez': { $stdDevPop: $<entrez_id> }
                }

        }
])


//Query for NCI, Other Dementia db.rna.aggregate([
        {
                $match: {
```

```
                    “DIAGNOSIS”: <tuple_arg1>,
                }
        },
        {
                $group: {
                    ‘_id’: value
                    ‘Avg_entrez’: { $avg : $<entrez_id> },
                    ‘Std_entrez’: { $stdDevPop: $<entrez_id> }
                }

        }
])
```

3. Given a gene, find all other information associated with this gene

To answer this question we need to analyze the data in entrez_ids_uniprot.csv and entrez_ids_genesymbol.csv. The data is stored in uniprot collection. To get this information a simple call using py2neo package in the python interface retrieved the documents needed. We query uniprot collection for all docs that reference the gene symbol given to the function.

```
db.uniprot.find({‘gene_symbol’:gene})
```

4. Given a patient id, find all patient information (age, gender, education, etc)

To answer this question we need to analyze the data in patients.csv and ROSMAP_RNASeq_entrez.csv. Patient information was all stored in cassandra. We need to query the RNA collection because it provides us the information what the patient was diagnosed with.

The patient table in the cassandra database we created using cassandra python drivers ORM. To query for patient information use used:

```
Patient.objects.get(patient_id=<id>)
```

To query the rna collection

```
db.rna.find({PATIENT_ID:<id>})
```

Execution Results

Check the doc folder for screenshots.

Potential Problems/Prospects/Discussion

Although this project does answer the questions asked, there are some inefficiencies and potential flaws in the design. The first thing I want to address is the use of cassandra for patient information. The

problem with this is that although cassandra is a NoSql database their models are much more stricter than Mongo's document store system. Another thing about this is that we haven't used Cassandra full potential in high availability. Cassandra has a multiple master model which would be very beneficial to have implemented in our system. But because this is a small project, this isn't something we should worry too much about.

Another design problem I see is the merge tool we have implemented. The merge tool adds gene symbol attribute into the uniprot collection. The uniprot collection was built using entrez_ids_uniprot txt file. Then the gene symbols attribute was pulled from the gene collection. The gene collection was built from the entrez_ids_genesymbol csv. We basically have duplicate and redundant data. The merge tool is also a very slow process. This process may take up to 7 minutes to complete.

Another design problem is the upload tool for uploading PPI (upload_interactors.py) This script may take up to 10 minutes to complete.

Now onto the CLI. We believe that it would be more help for the user if we given them the option to enter entrez ids as an alternative to looking up gene information. Currently the user can look up information if they know the gene symbols.

A shortcoming to this project is with the NOrder Gene Query. Although the query is successful, for some ids, we can't find the corresponding gene symbol. Also for large orders, the operation of retrieving interactor node takes a longer time (as expected).

Another problem is that we have is with error checking. When the user enters incorrect gene names and patient ids, the CLI will crash.