ECS8052 Final Assignment

Hugo Collins
Student Number: 40446925
Email: hcollins05@qb.ac.uk
MSc in Artificial Intelligence 2024

Word Count (Excluding Front Page, References, Figure Titles, Titles and Tables): 3151

Abstract

This report focuses on analysing and creating a knowledge graph using the PrimeKG dataset. This dataset contains information from thousands of diseases and information about them. In this report I detail my findings and explain the steps taken to recreate the knowledge graph throw the python package OwlReady2. I describe how I extracted a subgraph of two disease nodes and examined the relations between nodes using logic. Finally, the dataset was also analysed in the context of a Bayesian network, revealing further insights into the probabilistic relationships among entities.

Data Exploration and Cleaning:

The first step in the data exploration process was loading the data from the *kg.csv* file. This file holds the structural information of the graph, where each row represents an edge, and includes details about the connected nodes, such as their type, name, and the source and target nodes for the relationship. This makes the dataset ideal for reconstructing the knowledge graph. Also, I loaded the *disease_features.tab* and *drug_features.tab* files, as they contained important domain knowledge necessary for the graph which will after all, be used for medical purposes. These files were tab-separated, so I adjusted the read_csv() function to specify the delimiter as '\t'. Interestingly, while importing the drug data, I had no issues. However, when importing the disease data, I encountered duplicates, which may have been caused by errors during web scraping.

Next, I wanted to see how many nodes and edges were in the dataset. Extracting the number of edges was simple and just required counting the length of the dataset. The number of nodes, however, required counting the number of unique x_index and y_index values rather than x_type. This was due to the layout of kg.csv and the fact that each row represented a relation. In total, there were 129,375 nodes with 8,100,498 relationships. In the original paper, they document the same number of nodes but only 4,050,249 relationships. Interestingly, this is exactly half the number of relationships I found. This would indicate to me that despite the publication referring to undirected edges, there appear to be two alternatively directed relationships between each pair of nodes, leading to double the number of relationships. I will investigate this further.

Section 1

1. Question 1

On closer inspection of the data there was 30 distinct types of relation within the data which are listed below:

```
['anatomy_anatomy', 'anatomy_protein_absent', 'anatomy_protein_present', 'bioprocess_bioprocess', 'bioprocess_protein', 'cellcomp_cellcomp', 'cellcomp_p
rotein', 'contraindication', 'disease_disease', 'disease_phenotype_negative', 'disease_phenotype_positive', 'disease_protein', 'drug_drug', 'drug_effec
t', 'drug_protein', 'exposure_bioprocess', 'exposure_cellcomp', 'exposure_disease', 'exposure_exposure', 'exposure_molfunc', 'exposure_protein', 'indica
tion', 'molfunc_molfunc', 'molfunc_protein', 'off-label use', 'pathway_pathway', 'pathway_protein', 'phenotype_phenotype', 'phenotype_protein', 'protein
_protein']
```

Fig. 1 *Unique Relation*

To delve deeper into this, I extracted the counts of each relation type

```
anatomy_protein_present occurs 3036406 times
drug_drug occurs 2672628 times
protein_protein occurs 642150 times
disease_phenotype_positive occurs 300634 times
bioprocess_protein occurs 289610 times
cellcomp_protein occurs 166804 times
disease_protein occurs 160822 times
molfunc_protein occurs 139060 times
drug_effect occurs 129568 times
bioprocess_bioprocess occurs 105772 times
pathway_protein occurs 85292 times
disease_disease occurs 64388 times
contraindication occurs 61350 times
drug_protein occurs 51306 times
anatomy_protein_absent occurs 39774 times
phenotype_phenotype occurs 37472 times
anatomy_anatomy occurs 28064 times
molfunc_molfunc occurs 27148 times
indication occurs 18776 times
cellcomp_cellcomp occurs 9690 times
phenotype_protein occurs 6660 times
off-label use occurs 5136 times
pathway_pathway occurs 5070 times
exposure_disease occurs 4608 times
exposure_exposure occurs 4140 times
exposure_bioprocess occurs 3250 times
exposure_protein occurs 2424 times
disease_phenotype_negative occurs 2386 times
exposure_molfunc occurs 90 times
exposure_cellcomp occurs 20 times
```

Fig. 2 *Unique Relation Counts by Type*

The dataset highlights a strong focus on anatomy and protein interactions, with anatomy_protein_present being the most common with 3,036,406 occurrences. Protein-related relationships, including protein_protein (642,150) and bioprocess_protein (289,610), stand out, signifying that protein nodes are a key link between the different node types. There appear to be some underrepresented areas, like exposure-related connections (e.g., exposure_molfunc at 90) which could be an area to expand the graph. My final takeaway was that intra-category relationships exist for all 10 node types, such as disease-to-disease and drug-to-drug associations. These relationships are clearly of significant importance from a medical standpoint.

2. Question 2

Then I looked at the nodes of the graph. There were 10 unique nodes in the graph

```
['anatomy', 'biological_process', 'cellular_component', 'disease', 'drug', 'effect/phenotype', 'exposure', 'gene/protein', 'molecular_function', 'pathway']
```

Fig. 3 *Unique Nodes*

They had count:

```
10 node types, sorted by their frequency of occurrence:
- biological_process: 28642
- gene/protein: 27671
- disease: 17080
- effect/phenotype: 15311
- anatomy: 14035
- molecular_function: 11169
- drug: 7957
- cellular_component: 4176
- pathway: 2516
- exposure: 818

The most common node type is 'biological_process', occurring 28642 times.
The least common node type is 'exposure', occurring 818 times.
CPU times: total: 8min 6s
Wall time: 8min 10s
```

Fig. 4 *Unique Node Counts by Type*

The takeaways were the same as the relations with proteins and biological processes dominating the knowledge graph. I was surprised by the number of drugs however, which I thought would be higher, considering there are over double the number of diseases.

I then decided to plot pie charts of the counts to visualise my data in a more appealing way.

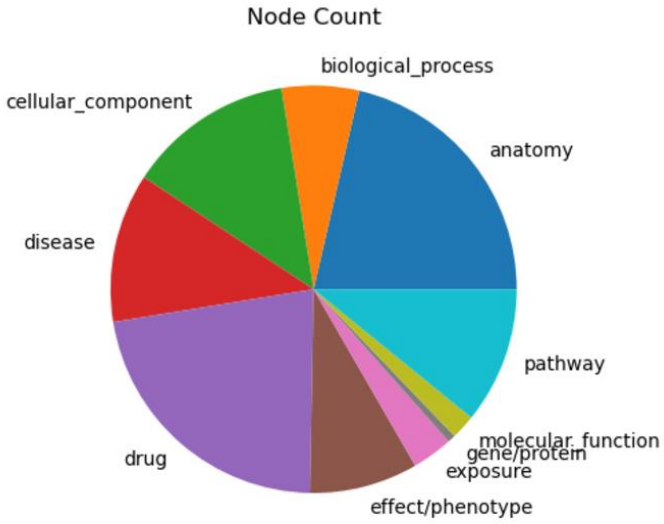


Fig. 5 *Pie Chart of Unique Node*

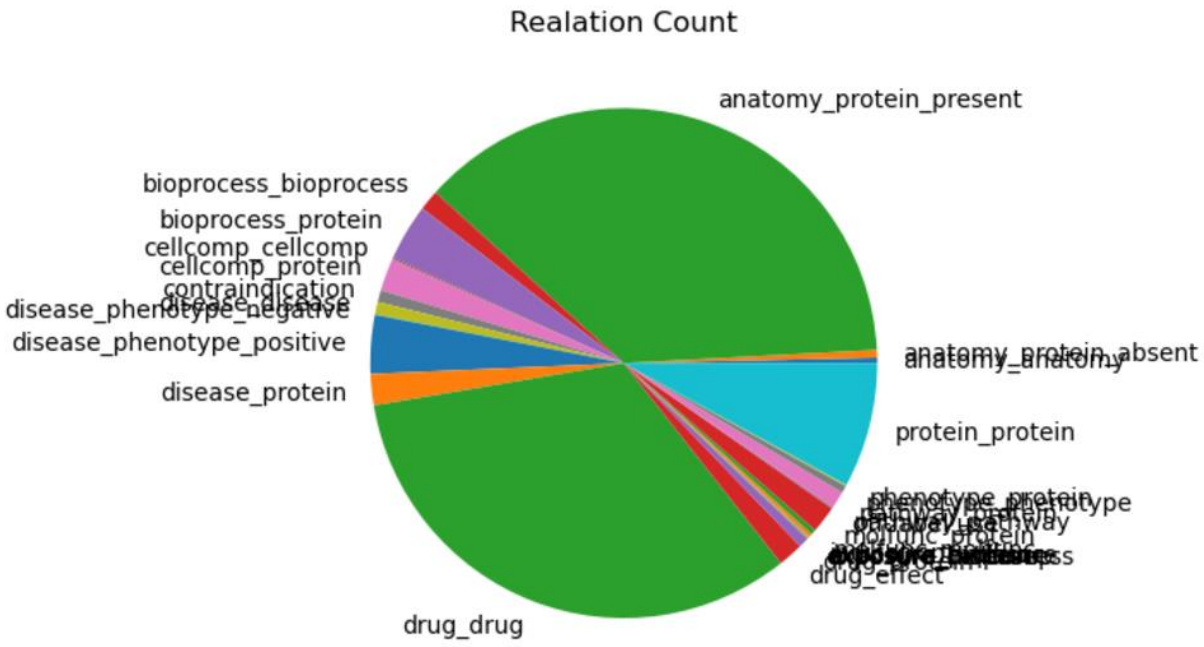


Fig. 6 *Pie chart of Unique Relations*

Question 3:

On first appearance my node counts do appear to relate to those in the paper published by Chandak et al 2023. Originally when counting nodes based on x_type there was a discrepancy, but this was due to the format of the data and the fact that each row represented a relation resulting in far too many nodes.

However, In the paper, the authors reference 4,050,249 million relationships, whereas the KG data I analysed contains 8,100,498 million—exactly double. In the report the publishers reference 30 undirected edges but the fact that there are 8,100,498 relations in the graph indicated to me that there are two directed edges for each undirected one. On closer inspection it does appear that for every X_type and Y_type pairing there is a reverse pairing resulting in two directed edges for each relation. I would imagine this is due to the way the data is formatted, as only having one x_type to y_type would result in one directed relation rather than an unrelated one and this may have been the only way to capture that. This does seem to be an area that is quite unclear, however.

Question 4

Here I just extracted the first three nodes alphabetically from each type. Some nodes names started with numbers or symbols which moved them to the top of the ordered list.

```
Node Type: gene/protein
First 3 nodes and their indices:
Index 1: Node Index 7097, Node Name A1BG
Index 2: Node Index 83051, Node Name A1BG-AS1
Index 3: Node Index 2724, Node Name A1CF

 Node Type: drug
 First 3 nodes and their indices:
 Index 1: Node Index 15992, Node Name (+)-2-(4-biphenyl)propionic acid
 Index 2: Node Index 17614, Node Name (+)-Rutamarin alcohol
 Index 3: Node Index 19012, Node Name (1'R,2'S)-9-(2-Hydroxy-3'-Keto-Cyclopenten-1-yl)Adenine
```

```
Node Type: effect/phenotype
First 3 nodes and their indices:
Index 1: Node Index 88363, Node Name 1-2 finger syndactyly
Index 2: Node Index 86390, Node Name 1-2 toe complete cutaneous syndactyly
Index 3: Node Index 88370, Node Name 1-2 toe syndactyly


Node Type: disease
First 3 nodes and their indices:
Index 1: Node Index 39879, Node Name 'psoriatic arthritis, susceptibility to
Index 2: Node Index 32869, Node Name 10q22.3q23.3 microduplication syndrome
Index 3: Node Index 32497, Node Name 11p15.4 microduplication syndrome


 Node Type: biological_process
 First 3 nodes and their indices:
 Index 1: Node Index 107968, Node Name 'de novo' AMP biosynthetic process
 Index 2: Node Index 114817, Node Name 'de novo' CTP biosynthetic process
 Index 3: Node Index 110881, Node Name 'de novo' GDP-L-fucose biosynthetic process


Node Type: molecular_function
First 3 nodes and their indices:
Index 1: Node Index 115560, Node Name (+)-2-epi-prezizaene synthase activity
Index 2: Node Index 115686, Node Name (+)-abscisic acid 8'-hydroxylase activity
Index 3: Node Index 121092, Node Name (+)-abscisic acid D-glucopyranosyl ester transmembrane transporter activity


 Node Type: cellular_component
 First 3 nodes and their indices:
 Index 1: Node Index 124820, Node Name 1,3-beta-D-glucan synthase complex
 Index 2: Node Index 125583, Node Name 1-alkyl-2-acetylglycerophosphocholine esterase complex
 Index 3: Node Index 125558, Node Name 2-iminoacetate synthase complex


Node Type: exposure
First 3 nodes and their indices:
Index 1: Node Index 61827, Node Name 1,1,1-trichloroethane
Index 2: Node Index 61828, Node Name 1,1,2,2-tetrachloroethane
Index 3: Node Index 127452, Node Name 1,12-benzoperylene


 Node Type: pathway
 First 3 nodes and their indices:
 Index 1: Node Index 127818, Node Name 2-LTR circle formation
 Index 2: Node Index 128914, Node Name 5-Phosphoribose 1-diphosphate biosynthesis
 Index 3: Node Index 127841, Node Name A tetrasaccharide linker sequence is required for GAG synthesis


Node Type: anatomy
First 3 nodes and their indices:
Index 1: Node Index 68465, Node Name 1st arch mandibular component
Index 2: Node Index 71026, Node Name 1st arch mandibular ectoderm
Index 3: Node Index 71027, Node Name 1st arch mandibular endoderm
```

Question 5

	anatomy	biological_process	cellular_component	disease	drug	effect/phenotype	exposure	gene/protein	molecular_function	pathway
anatomy_anatomy	28064.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
anatomy_protein_absent	19887.0	0.0	0.0	0.0	0.0	0.0	0.0	19887.0	0.0	0.0
anatomy_protein_present	1518203.0	0.0	0.0	0.0	0.0	0.0	0.0	1518203.0	0.0	0.0
bioprocess_bioprocess	0.0	105772.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
bioprocess_protein	0.0	144805.0	0.0	0.0	0.0	0.0	0.0	144805.0	0.0	0.0
cellcomp_cellcomp	0.0	0.0	9690.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
cellcomp_protein	0.0	0.0	83402.0	0.0	0.0	0.0	0.0	83402.0	0.0	0.0
contraindication	0.0	0.0	0.0	30675.0	30675.0	0.0	0.0	0.0	0.0	0.0
disease_disease	0.0	0.0	0.0	64388.0	0.0	0.0	0.0	0.0	0.0	0.0
disease_phenotype_negative	0.0	0.0	0.0	1193.0	0.0	1193.0	0.0	0.0	0.0	0.0
disease_phenotype_positive	0.0	0.0	0.0	150317.0	0.0	150317.0	0.0	0.0	0.0	0.0
disease_protein	0.0	0.0	0.0	80411.0	0.0	0.0	0.0	80411.0	0.0	0.0
drug_drug	0.0	0.0	0.0	0.0	2672628.0	0.0	0.0	0.0	0.0	0.0
drug_effect	0.0	0.0	0.0	0.0	64784.0	64784.0	0.0	0.0	0.0	0.0
drug_protein	0.0	0.0	0.0	0.0	25653.0	0.0	0.0	25653.0	0.0	0.0
exposure_bioprocess	0.0	1625.0	0.0	0.0	0.0	0.0	1625.0	0.0	0.0	0.0
exposure_cellcomp	0.0	0.0	10.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0
exposure_disease	0.0	0.0	0.0	2304.0	0.0	0.0	2304.0	0.0	0.0	0.0
exposure_exposure	0.0	0.0	0.0	0.0	0.0	0.0	4140.0	0.0	0.0	0.0
exposure_molfunc	0.0	0.0	0.0	0.0	0.0	0.0	45.0	0.0	45.0	0.0
exposure_protein	0.0	0.0	0.0	0.0	0.0	0.0	1212.0	1212.0	0.0	0.0
indication	0.0	0.0	0.0	9388.0	9388.0	0.0	0.0	0.0	0.0	0.0
molfunc_molfunc	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	27148.0	0.0
molfunc_protein	0.0	0.0	0.0	0.0	0.0	0.0	0.0	69530.0	69530.0	0.0
off-label use	0.0	0.0	0.0	2568.0	2568.0	0.0	0.0	0.0	0.0	0.0
pathway_pathway	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5070.0
pathway_protein	0.0	0.0	0.0	0.0	0.0	0.0	0.0	42646.0	0.0	42646.0
phenotype_phenotype	0.0	0.0	0.0	0.0	0.0	37472.0	0.0	0.0	0.0	0.0
phenotype_protein	0.0	0.0	0.0	0.0	0.0	3330.0	0.0	3330.0	0.0	0.0
protein_protein	0.0	0.0	0.0	0.0	0.0	0.0	0.0	642150.0	0.0	0.0

Fig. 7 *Table that shows how often each type of node is in each type of relation.*

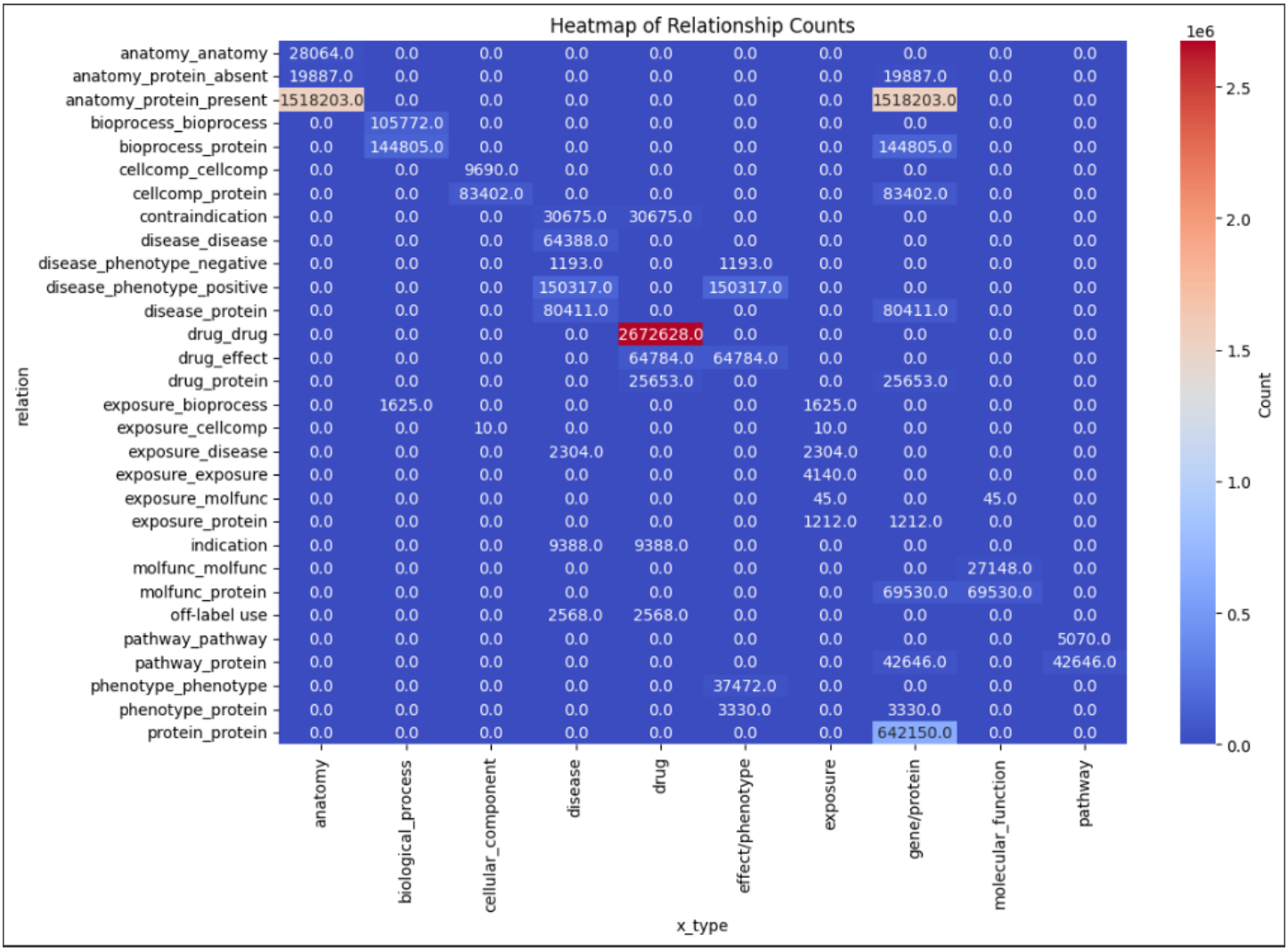


Fig. 8 *Heatmap that shows how often each type of node is in each type of relation.*

The data shows that some relationships are more common than others. For example, the drug_drug relationship has over 2.6 million entries, and anatomy_protein_present has more than 1.5 million. On the contrary, many other relationships, like those involving bioprocess, cellcomp, and phenotype, have very few. Many of the relationships are self-explanatory from their names, but the table above revealed that less intuitive ones, such as contraindication and indication, primarily connect drug and disease nodes. A heatmap of these counts highlights the relationships that occur most

frequently, emphasizing areas with the highest activity. Interestingly, while disease, gene/protein, and drug columns contain most connections, they do not correspond to the largest number of nodes. Surprisingly, bioprocesses despite having the highest node count, actually doesnt connect to a lot of different nodes.

Ontology

To construct the knowledge graph using the kg.csv file, I used the Python package owlready2 used in our lectures, which supports ontology-oriented programming in Python. To create the graph, I first assembled the ontology by extracting entity types from the x_type column and relations from the relation column. I then dynamically created the ontology nodes using new_class and the Thing property, while using a dictionary to store the relations. Finally, I mapped the relations to the nodes using the ObjectProperty from owlready2. All code which is described in further detail can be found in my attached notebook.

Problem with plotting:

I first tried to plot my ontology using the d3graph() function which is a java script library for producing dynamic interactive graphs. Unfortunately, I found this ineffective for several reasons. Firstly, I struggled to find a way to label the relations which was not ideal. Also as seen in the example below, the graph was struggling to capture the correct number of relations and was not acknowledging the same type connections (e.g. disease disease) at all. I discovered this error was a result of the vec2adjmat creating 32 relations when I needed 50.

In Section 1, I identified 30 unique relationships, including 10 links between nodes of the same type. The graph also accounts for bidirectional links between nodes of different types, doubling those 20 connections (30 total - 10 same-type links), resulting in 50 links in the graph. An example of the original plot can be seen below:

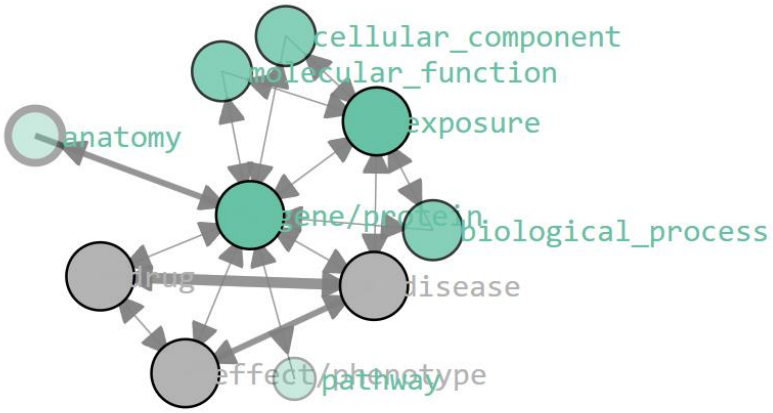


Fig. 9 *Plot of Ontology using Original Code*

Plotting using gravis library

I decided to research alternative methods to plot the ontology and came across a medium page claiming this gravis package was "*The New Best Python Package for Visualising Graphs*". It was still a java-based library but allowed the graph to be plotted in the notebook and also add colours to nodes and easily label relations. I used my dynamically created code through owlready2 but made a slight change to my relation section. I inputted the data in tuple format and allowed for self-loops where range and domain were the same. I created a colour dictionary and then created a graph using the library igraph as found online. I plotted using the gravis code then and the ontology can be seen below.

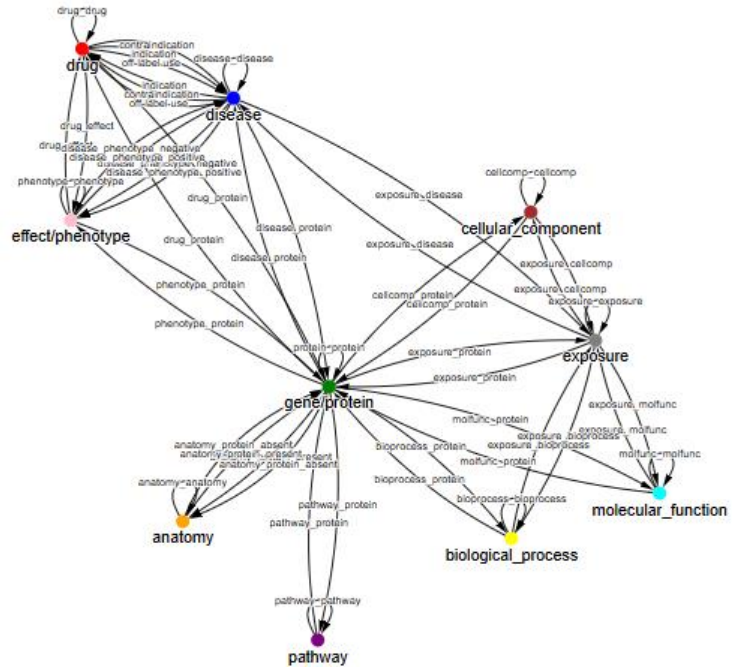


Fig. 10 *Plot of Ontology using Gravis*

Constructing the Knowledge Graph

The in-depth code can be found in my notebook as I am not going to talk through the entire process of creating the graph. This data was stored in a list which was different to the previosu examples which used dictionaries so a bit of adaptation to the code was necessary. I used a mapping system to hurry up the process for x_index and x_name also. I had originally tried creaing nodes based on name using

$$nodetype = x\_name\_map[name]$$

$$node = EntityClass[nodetype](id=name)$$

This, however, was not a suitable solution as it led to a class__assignment error. On closer inspection, this was because several of the node names shared the name with the entity types such as biological_process and python was unable to create anode as it thought it was type “Thing.” My first solution was to find all occurrences of these names and capitalise the first letter. This worked for all but pathway which had sentence length names and was much more difficult to deal with. I then noticed in the README that “x_index” and “y_index” were the primary keys which prompted me to change the approach to:

$$nodetype = x\_type\_map[index]$$

$$node = EntityClass[nodetype](id=index)$$

I added the node name, type, and relations as attributes and included an 'attributes' dictionary to store additional metadata from the disease and drug tab files for the relevant nodes. This approach was much easier than creating individual attributes for each data entry in the tab files, which would have resulted in numerous empty attributes for non-disease and non-drug type nodes.

```
Extracted Disease Node:
Node ID: 27158
Name: osteogenesis imperfecta
Type: disease
Attributes: {'mondo_id': 13924, 'mondo_name': 'osteogenesis imperfecta type 13', 'group_id_bert': '13924_12592_14672_13460_12591_12536_30861_8146_8148_
32846_13459_44329_14544_9805_49223_9804_14086_8147_13515_14029_12581_19019', 'group_name_bert': 'osteogenesis imperfecta', 'mondo_definition': 'Any ost
eogenesis imperfecta in which the cause of the disease is a mutation in the BMP1 gene.', 'umls_description': 'collagen diseases characterized by brittl
e, osteoporotic, and easily fractured bones. It may also present with blue sclerae, loose joints, and imperfect dentin formation. Most types are autoso
mal dominant and are associated with mutations in collagen type i.', 'orphanet_definition': nan, 'orphanet_prevalence': nan, 'orphanet_epidemiology': n
an, 'orphanet_clinical_description': nan, 'orphanet_management_and_treatment': nan, 'mayo_symptoms': nan, 'mayo_causes': nan, 'mayo_risk_factors': nan,
'mayo_complications': nan, 'mayo_prevention': nan, 'mayo_see_doc': nan}
Relationships: 267
Connected Nodes:
  - Hearing impairment (ID: 22304, Type: effect/phenotype)
  - Intellectual disability (ID: 22675, Type: effect/phenotype)
  - Bruising susceptibility (ID: 84774, Type: effect/phenotype)
  - Wormian bones (ID: 85212, Type: effect/phenotype)
  - Abnormality of the dentition (ID: 22425, Type: effect/phenotype)
  - Hearing abnormality (ID: 22288, Type: effect/phenotype)
  - Abnormal sclera morphology (ID: 22388, Type: effect/phenotype)
```

Fig. 11 *Example Disease Node*

A sample disease node can be seen above after being extracted from the graph. Please note there are more connections, but I could not include them all due to figure size. (All 267 are present this is just an example)

```
[99]: node_count = len(TheGraph)
      relationship_count = sum(len(node.rel) for node in TheGraph.values())

      print(f"Number of nodes in my Graph : {node_count}")
      print(f"Number of relationships (edges) in my Graph: {relationship_count}")

      Number of nodes in my Graph : 129375
      Number of relationships (edges) in my Graph: 8100498
```

Fig. 12 *Counts of nodes and relations in my graph*

This was just a sanity check to ensure all nodes and edges were present. Interestingly as with the original graph published my Graph contained over 8 million nodes meaning each node had two bidirectional edges between each connection. There is a definite discrepancy here as in the paper they refer to undirected edges.

Challenges:

While attempting to populate the graph using [node.name], I repeatedly had errors related to ObjectProperties. After much debugging, I discovered the issue came from several nodes sharing or containing names identical to those of entities. This caused type errors when the graph attempted to process these nodes, as some were treated as objects and others as entities.

Initially, I tried resolving this by capitalising the first letter of conflicting node names, which worked to some extent. However, I found a cleaner and more efficient approach: using the x_index as the unique node identifier while storing the original name as an attribute. This method resolved the naming conflicts and ensured the graph could be populated without errors.

Subgraph:

When creating the subgraph, it had to meet certain criteria. Firstly, the two nodes had to be of type disease and not directly connected. Also, I wanted the shortest path to be under length 5 and also tried to start at nodes with fewer neighbours. This was because previously, the subgraphs were too large to be able to transform into a rule-based system. My first idea was to search for interesting nodes, so I decided to start with finding the shortest path between a cancer and a diabetes node. Eventually, I found a path between gestational diabetes and breast cancer which was quite interesting but unfortunately, I could not use this as it still had far too many neighbouring nodes as many cancers share many neighbours.

So instead, I randomly selected two nodes that met the conditions and examined their subgraph until I found a suitable two disease nodes. The path I found was from as follows:

```
myeloperoxidase deficiency -> MPO -> hepatocellular carcinoma -> PTEN -> atypical endometrial hyperplasia
```

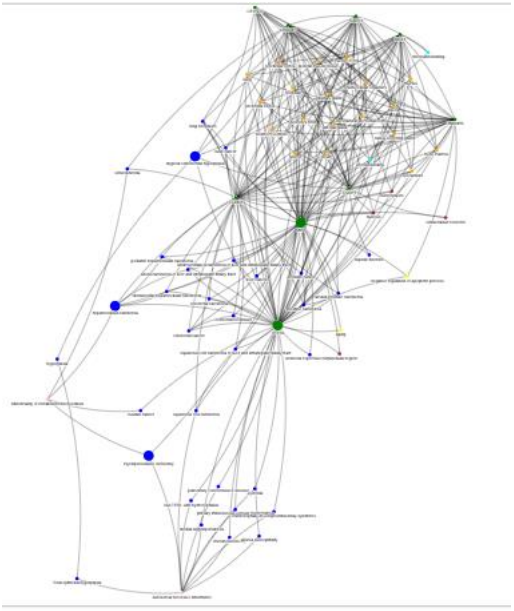
Just to gain a quick understanding I did some research on both. Myeloperoxidase deficiency is an iron-based deficiency which leads to increased infections in an individual. Atypical endometrial hyperplasia is a condition where the womb lining called the endometrium gets thicker and can lead to uterine cancer. The main cause of atypical endometrial hyperplasia is infection which made this an interesting link to evaluate. It had a shortest path of four relations and 5 nodes containing:

```
Node Types and Counts:
gene/protein: 2
disease: 3

Edge Types and Counts:
disease_gene/protein: 4
```

The link included proteins MPO and PTEN (which provides instructions for making an enzyme that is found in most tissues in the body) and the disease node hepatocellular carcinoma which is the most common type of liver cancer in adults.

For plotting the graph, I again used the gravis package and added colour to the node types. I labelled nodes by their name and decided to increase the size of the nodes on the shortest path so that it was clear where they were on the graph. The final graph, which contains **70 nodes** and **307 edges**, along with the distribution of edge types, is shown below. However, it is presented more clearly in the notebook.



```
Node Type: disease, Count: 32, Color: blue
Node Type: gene/protein, Count: 9, Color: green
Node Type: cellular_component, Count: 4, Color: brown
Node Type: molecular_function, Count: 2, Color: cyan
Node Type: anatomy, Count: 19, Color: orange
Node Type: effect/phenotype, Count: 2, Color: pink
Node Type: biological_process, Count: 2, Color: yellow
```

Fig. 13&14 *Plot of Subgraph of diseases myeloperoxidase deficiency and atypical endometrial hyperplasia / Types of nodes in Subgraph*

Part 3: Deriving a Knowledge Base and Inferring New Relations

To create a knowledge base, I simply created a for loop and extracted the list of connections. One important note was that my subgraph had undirected edges but to keep my rules consistent with those in the knowledge graph I had to add a reverse relation as there are two directed edges for each relation in the graph, one from each node. My first rules had the format:

```
If node is schizophrenia then node is connected to CCND1.
If node is CCND1 then node is connected to schizophrenia.
If node is schizophrenia then node is connected to CDKN1C.
If node is CDKN1C then node is connected to schizophrenia.
```

Fig. 15 *Knowledge Base 1*

But I also wanted to put the knowledge base in a tuple format consistent with the W3C N-Triples format of Subject– Predicate – Object.

```
('HNRNPK', 'connected_to', 'thymus')
('thymus', 'connected_to', 'HNRNPK')
('HNRNPK', 'connected_to', 'bone marrow')
('bone marrow', 'connected_to', 'HNRNPK')
('HNRNPK', 'connected_to', 'dorsolateral prefrontal cortex')
```

Fig. 16 *Knowledge Base 2*

Inferring New Relations:

For this section I wanted to infer some interesting relationships. I used forward chaining which involved selecting a start node and searching until the end clause is found. I used breadth first search which involved checking all immediate neighbours of a node before moving on to the next set of neighbours. As you can imagine, this was quite a computationally exhaustive search as a number of the nodes, especially the protein nodes, had a lot of neighbours. I first did a manual approach before checking using code I created which mapped out the process. I ordered the neighbouring nodes alphabetically to give an order to the way my bfs was searching.

Inferred Relationship 1: hyperplasia (disease) -> dystonia (disease) (inferred_relation_type = disease_disease)

Path: hyperplasia -> atypical endometrial hyperplasia -> PTEN -> dystonia

There is an inferred relationship here between hyperplasia and dystonia, which involved checking multiple nodes in the graph. The pathway shows that there is a link between hyperplasia and dystonia through changes in the PTEN gene. My logical reasoning for this inferred relationship can be seen below:

- **Premise 1**: If hyperplasia is connected to atypical endometrial hyperplasia,
- **Premise 2**: and atypical endometrial hyperplasia leads to changes in the PTEN,
- **Premise 3**: and **gene mutations** are known to cause dystonia,
- **Conclusion**: Then hyperplasia can have an inferred relationship with dystonia

The path taken to reach the goal using breath first and forward chaining was as follows:

```
Order of visited nodes:
hyperplasia -> atypical endometrial hyperplasia -> focal epithelial hyperplasia -> CDKN1C -> CELSR2 -> CSRP1 -> DDX1 -> PTEN -> Autosomal recessive
inheritance -> Ammon's horn -> CCND1 -> RAD21 -> adrenal gland -> blood -> bone marrow -> cerebellar cortex -> dorsolateral prefrontal cortex -> for
ebrain -> heart -> heart left ventricle -> lung -> lung cancer -> lung neoplasm -> multi-cellular organism -> muscle of leg -> myocardium -> nucleus
-> pituitary gland -> prefrontal cortex -> protein binding -> saliva-secreting gland -> schizophrenia -> skeletal muscle tissue -> spleen -> thymus
-> bipolar disorder -> extracellular exosome -> familial prostate carcinoma -> prostate cancer -> prostate carcinoma -> HNRNPK -> chromatin binding
-> nucleoplasm -> VACTERL with hydrocephalus -> adenocarcinoma of liver and intrahepatic biliary tract -> aging -> colorectal cancer -> colorectal c
arcinoma -> colorectal neoplasm -> dystonia
```

The gene/protein PTEN had a lot of neighbours which needed to be searched before moving on.

Inferred Relationship 2: aging (biological_process) -> prostate cancer (disease)

Path: aging -> MPO -> prostate cancer

I found an inferred relationship here between aging and prostate cancer, which I was surprised to find was not explicitly represented in the graph. In the graph, there is no direct biological process → disease relation, which seems odd to me, given the well-established link between age and the increased risk of prostate cancer. The pathway shows that MPO connects aging and prostate cancer. The logical reasoning behind this inferred relationship is as follows:

- **Premise 1:** Aging contributes to changes in gene/protein activity in the MPO gene.
- **Premise 2:** MPO gene mutations or effects are linked to prostate cancer.
- **Conclusion:** Therefore, aging can increase the risk of prostate cancer, showing an inferred relationship.

The path taken to reach the goal using breath first and forward chaining was as follows:

```
Order of visited nodes:
aging -> MPO -> PTEN -> Ammon's horn -> CCND1 -> HNRNPK -> RAD21 -> adenocarcinoma of liver and intrahepatic biliary tract -> adrenal gland -> bipol
ar disorder -> blood -> bone marrow -> cerebellar cortex -> chromatin binding -> colorectal cancer -> colorectal carcinoma -> colorectal neoplasm ->
dorsolateral prefrontal cortex -> extracellular exosome -> extracellular region -> familial prostate carcinoma -> fibrolamellar hepatocellular carci
noma -> forebrain -> heart -> heart left ventricle -> hepatocellular carcinoma -> ischemia reperfusion injury -> liver cancer -> lung -> lung cancer
-> lung neoplasm -> multi-cellular organism -> muscle of leg -> myeloperoxidase deficiency -> myocardium -> negative regulation of apoptotic process
-> nucleoplasm -> nucleus -> pediatric hepatocellular carcinoma -> pituitary gland -> prefrontal cortex -> prostate cancer
```

The gene/protein MPO, like PTEN above, had a lot of neighbours which needed to be searched before moving on.

Inferred Relationship 3: heart (anatomy) -> Abnormality of metabolism/homeostasis (effect/phenotype)

Path: heart -> CCND1 -> hepatocellular carcinoma -> Abnormality of metabolism/homeostasis

Again, I wanted to infer some interesting meaningful relationships. Using forward chaining and breadth first search I inferred the relationship here between the heart and Abnormality of metabolism/homeostasis. The logical reasoning behind this inferred relationship is as follows:

- **Premise 1**: If heart is connected to CCND1,
- **Premise 2:** and CCND1 is associated with Hepatocellular carcinoma (liver cancer),
- **Premise 3:** and Hepatocellular carcinoma can clearly lead to abnormalities in metabolism and homeostasis,
- **Conclusion:** Then a relationship between heart and abnormalities in metabolism/homeostasis can be inferred through the CCND1 gene and disese Hepatocellular carcinoma.

The path taken to reach the goal using breath first and forward chaining was as follows:

```
Order of visited nodes:
heart -> CCND1 -> CDKN1C -> CELSR2 -> CSRP1 -> DDX1 -> HNRNPK -> MPO -> PTEN -> RAD21 -> Ammon's horn -> adenocarcinoma of liver and intrahepatic bi
liary tract -> adrenal gland -> blood -> cerebellar cortex -> colorectal cancer -> colorectal carcinoma -> colorectal neoplasm -> dorsolateral prefr
ontal cortex -> familial prostate carcinoma -> fibrolamellar hepatocellular carcinoma -> forebrain -> heart left ventricle -> hepatocellular carcino
ma -> liver cancer -> lung -> lung cancer -> lung neoplasm -> multi-cellular organism -> muscle of leg -> myocardium -> nucleoplasm -> nucleus -> pe
diatric hepatocellular carcinoma -> pituitary gland -> prefrontal cortex -> prostate cancer -> prostate carcinoma -> protein binding -> saliva-secre
ting gland -> schizophrenia -> skeletal muscle tissue -> spleen -> squamous cell carcinoma -> squamous cell carcinoma of liver and intrahepatic bili
ary tract -> thymus -> undifferentiated carcinoma of liver and intrahepatic biliary tract -> atypical endometrial hyperplasia -> bone marrow -> bipo
lar disorder -> extracellular exosome -> chromatin binding -> negative regulation of apoptotic process -> aging -> extracellular region -> ischemia
reperfusion injury -> myeloperoxidase deficiency -> VACTERL with hydrocephalus -> dystonia -> familial angiolipomatosis -> glioma susceptibility ->
immunodeficiency -> macrocephaly-developmental delay syndrome -> ovarian cancer -> primary intraosseous venous malformation -> pulmonary venoocclusi
ve disease -> Abnormality of metabolism/homeostasis
```

Again, it was the gene CCND1 that had the most neighbors to search through. This is a recurring trend I have noticed that gene/protein is a key type in liking the entire graph, which is reflective of its higher relation count noticed in section 1.

Part 4: A Bayesian View of the Data

When considering how to calculate the probabilities from the data there was a lot to consider. Initially I broke the joint distribution into:

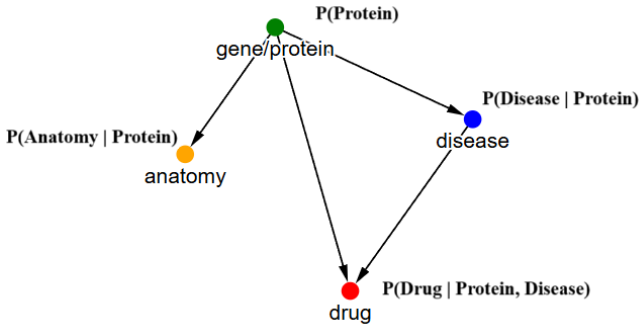


Fig. 17 *Original Decomposition*

However, this was not feasible as it was not possible to calculate P(Drug | Disease, Protein) using count as only two could occur together. So instead, I found an alterntaive decomposition:

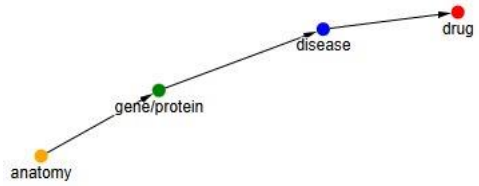


Fig. 18 *Final Decomposition*

$$P(\text{Protein, Drug, Disease, Anatomy}) = P(\text{Anatomy})\ P(\text{Protein}|\text{Anatomy})\ P(\text{Disease}|\text{Protein})\ P(\text{Drug}|\text{Disease})$$

In this analysis, I built a Bayesian network to model relationships among anatomical regions, proteins, diseases, and drugs using data from the PrimeKG dataset. I confirmed the variables were not independent, requiring conditional probabilities rather than simply multiplying marginal probabilities. For instance, proteins influence diseases, which in turn affect drug associations, making conditional probabilities essential.

Bayesian networks are directed acyclic graphs (DAGs), so I had to make sure the structure had no cycles, reflecting the natural causations from proteins to diseases and diseases to drugs. I assumed this directional causality was true and that the dataset was reliable, and other data did not influence the probabilities of these nodes

To ensure causality, I focused on x-type counts and paired them with y-type observations to maintain directionality. This approach preserved the causal structure by treating x-type variables as parent nodes influencing y-type outcomes. I only included observations where these four node types were involved in both the x_type and y_Type to avoid external data noise. I found this part of the project challenging. I also explored the possibility of counting by specific nodes but unfortunately, I was encountering many errors doing so which led to challenges with the second art of this assignment.

My final Bayesian model using pymc looked like

```
import pymc as mc

with mc.Model() as model:

# Probabilities as Bernoulli priors based on computed probabilities
        Anatomy = mc.Bernoulli("anatomy", p=P_Anatomy)  # Base probability of protein presence
        P_Protein = mc.Deterministic("P_protein", mc.math.switch(Anatomy, P_Protein_given_Anatomy, 1 - P_Protein_given_Anatomy))
        Protein = mc.Bernoulli("protein", p=P_Protein)

        P_Disease = mc.Deterministic("P_disease", mc.math.switch(Protein, P_Disease_given_Protein, 1 - P_Disease_given_Protein))
        Disease = mc.Bernoulli("disease", p=P_Disease)

        P_Drug = mc.Deterministic("P_drug", mc.math.switch(Disease, P_Drug_given_Disease, 1 - P_Drug_given_Disease))
        Drug = mc.Bernoulli("drug", p=P_Drug)

        # Sampling using Metropolis
        step = mc.Metropolis()
        trace = mc.sample(100000, step=step, tune=5000, random_seed=123, progressbar=True)
```

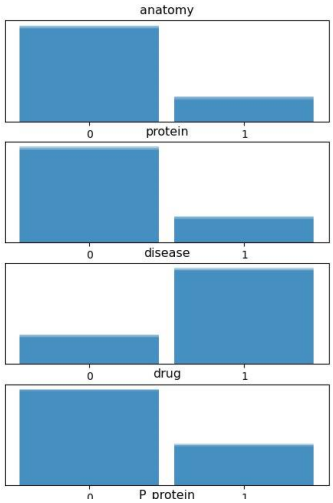


Fig. 19 & 20 *PYMC code and Output of Bayesian Model*

Section 4.2

Unfortunately, I had a lot of challenges with this section of the project. I could not find a valid link between drug and each anatomy type. I spent hours trying to find a way to incorporate the probabilities in the Bayesian model by individual nodes but to no avail. I was unable to find a way to complete this part of the project. I attempted to use the probabilities from the model, but this was returning static values. Also, as there is no link between anatomy and drug, I could not use count either to count their co-occurrences. This was extremely disappointing. I even tried counting the occurrences of each anatomical type and combining these probabilities with counts of anatomical-protein pairs, but the lengths of these distributions would not align, which prevented meaningful computation. Despite my efforts, I was unable to resolve these issues.

I did however calculate the probability of P(Drug | Anatomy) from my Bayesian model which returned a value of 0.567489

Conclusion

Overall, while this task was challenging, it was incredibly rewarding. Knowledge graphs proved to be powerful tools for organizing information and uncovering relationships within a system. The discrepancies between the actual research paper and the number of nodes in the data are intriguing, as there appear to be two directed nodes connecting each type, which could be explored further in future research. The subgraph analysis revealed fascinating insights into the relationships between two diseases and demonstrated how these graphs could be used to infer new connections between

drugs and diseases, shared proteins, and other potential biological relationships. Such discoveries could lead to advancements in medical research and improvements in treatment strategies in the future.

References:

Chandak, P., Huang, K. & Zitnik, M. Building a knowledge graph to enable precision medicine. Sci Data 10, 67 (2023). https://doi.org/10.1038/s41597-023-01960-3

Works Cited“Graph—Undirected Graphs with Self Loops — NetworkX 3.4.2 Documentation.” Networkx.org, 2024, networkx.org/documentation/stable/reference/classes/graph.html.

Lee, Benjamin. “The New Best Python Package for Visualising Network Graphs.” Medium, Towards Data Science, 23 Nov. 2023, towardsdatascience.com/the-new-best-python-package-for-visualising-network-graphs-e220d59e054e.

“Tutorial — NetworkX 2.5 Documentation.” Networkx.org, networkx.org/documentation/stable/tutorial.html.

“Visualisation of Graphs — Igraph 0.10.8 Documentation.” Igraph.org, 2023, python.igraph.org/en/0.10.8/visualisation.html.

2024.Zitnik, Marinka. “Precision Medicine Oriented Knowledge Graph.” Zitnik Lab, 2023, zitniklab.hms.harvard.edu/projects/PrimeKG/.