Knowledge Engineering

Module Report

submitted towards differentiation for the degree of

Msc Artificial Intelligence

by

Joseph McInerney - 40460549

EEECS
Queen's University Belfast

Lecturer
Professor Iain Styles

Wordcount: 2193

December 2024

Contents

1 Data Exploration **ii**
1.1 Relations . . . ii
1.2 Nodes . . . iii
1.3 Comparison to Original Paper . . . iii
1.4 Sorting Nodes . . . iii
1.5 Frequency of Node by in Relation . . . iv

2 Exploring the Knowledge Graph **vii**
2.1 The Ontology . . . vii
2.1.1 Constructing the Ontology . . . vii
2.1.2 Visualising the Ontology . . . vii
2.2 Constructing the Knowledge Graph . . . viii
2.2.1 Nodes . . . viii
2.2.2 Edges . . . viii
2.3 Finding a Subgraph . . . viii

3 Deriving the Knowledge Base **x**
3.1 Knowledge Base . . . x
3.2 Inference . . . x

4 A Bayesian View of the Data **xi**
4.1 Bayesian Network . . . xi
4.1.1 Topology . . . xi
4.1.2 Deriving Probabilities . . . xii
4.2 Finding Anatomical Region Associated to Drug . . . xiii

Bibliography **xiv**

1 | Data Exploration

1.1 Relations

Pandas' value_counts() method is used to count the frequency of relation types. There are 30 unique relationships in the data as shown in table 1.1. There is a large variation in the number of relationships by relationship type.

Relation	**Count**	**Order**
anatomy_protein_present	3,036,406	1
drug_drug	2,672,628	2
protein_protein	642,150	3
disease_phenotype_positive	300,634	4
bioprocess_protein	289,610	5
cellcomp_protein	166,804	6
disease_protein	160,822	7
molfunc_protein	139,060	8
drug_effect	129,568	9
bioprocess_bioprocess	105,772	10
pathway_protein	85,292	11
disease_disease	64,388	12
contraindication	61,350	13
drug_protein	51,306	14
anatomy_protein_absent	39,774	15
phenotype_phenotype	37,472	16
anatomy_anatomy	28,064	17
molfunc_molfunc	27,148	18
indication	18,776	19
cellcomp_cellcomp	9,690	20
phenotype_protein	6,660	21
off-label use	5,136	22
pathway_pathway	5,070	23
exposure_disease	4,608	24
exposure_exposure	4,140	25
exposure_bioprocess	3,250	26
exposure_protein	2,424	27
disease_phenotype_negative	2,386	28
exposure_molfunc	90	29
exposure_cellcomp	20	30
Total	8,100,498	30

Table 1.1: Relation counts with order and total count

1.2 Nodes

There are 10 different node types, sorted in ascending order and their counts in the data. These nodes are listed as 'x_type' in the data set. Table 1.2 shows the count of unique nodes identified by the column filter 'x_index' for each node type. It is important to get counts of unique nodes as if nodes can appear more than once in multiple relations. There is large variation in the number of nodes for node type.

Node	**Count**
biological process	28,642
gene/protein	27,671
disease	17,080
effect/phenotype	15,311
anatomy	14,035
molecular function	11,169
drug	7,957
cellular component	4,176
pathway	2,516
exposure	818
Total	129,375

Table 1.2: Node Types and their Counts

1.3 Comparison to Original Paper

In Chandak's et Al's [1] paper, they state that Precision Mediciing KNowledge Graph (PrimeKG) has 129,375 nodes having 10 different types and 4,050,249 relationships including 30 types of undirected relations. So, the number of nodes remains consistent whereas the number of relationships is exactly double for this project compared to the original paper. This is summarized in table 1.3. Each relation is listed twice in the data, indicating bidirectionality.

	Chandak	Project	Difference
Nodes	129,375	129,375	0
Relationships	4,050,249	8,100,498	4,050,249

Table 1.3: Comparison of Knowledge Graph Between Project Dataset and Chandak 2016.

1.4 Sorting Nodes

Nodes were sorted alphabetically by their name given their type. The first 3 entries were then displayed. This is shown in table 1.4. This table facilitates have a closer look at the data set which is too large to view all at once. From this table it can be gathered that nodes can appear more than once, indicating that nodes can have more than one relation. The indexes of the nodes also are close relative to type demonstrating that the data set is organized by type. This table is effective at getting more familiar with the data set.

Table 1.4: Summary of Nodes Grouped by Type and Sorted by Name.

x_type	**index**	**x_name**
anatomy	3840031	1st arch mandibular component

Continued on next page

x_type	index	x_name
anatomy	3842592	1st arch mandibular ectoderm
anatomy	3842593	1st arch mandibular endoderm
biological_process	6161451	'de novo' AMP biosynthetic process
biological_process	6405428	'de novo' AMP biosynthetic process
biological_process	6405429	'de novo' AMP biosynthetic process
cellular_component	6194209	1,3-beta-D-glucan synthase complex
cellular_component	6194210	1,3-beta-D-glucan synthase complex
cellular_component	6195455	1-alkyl-2-acetylglycerophosphocholine esterase complex
disease	3348216	'psoriatic arthritis, susceptibility to
disease	6062059	'psoriatic arthritis, susceptibility to
disease	3198429	10q22.3q23.3 microduplication syndrome
drug	327434	(+)-2-(4-biphenyl)propionic acid
drug	332953	(+)-Rutamarin alcohol
drug	336477	(1'R,2'S)-9-(2-Hydroxy-3'-Keto-Cyclopenten-1-yl)Adenine
effect/phenotype	5788650	1-2 finger syndactyly
effect/phenotype	5932029	1-2 finger syndactyly
effect/phenotype	5783695	1-2 toe complete cutaneous syndactyly
exposure	3783373	1,1,1-trichloroethane
exposure	6499608	1,1,1-trichloroethane
exposure	6501443	1,1,1-trichloroethane
gene/protein	22646	A1BG
gene/protein	32327	A1BG
gene/protein	41615	A1BG
molecular_function	6180479	(+)-2-epi-prezizaene synthase activity
molecular_function	6180607	(+)-abscisic acid 8'-hydroxylase activity
molecular_function	6188956	(+)-abscisic acid D-glucopyranosyl ester transmembrane transporter activity
pathway	6503545	2-LTR circle formation
pathway	6508455	2-LTR circle formation
pathway	6515767	2-LTR circle formation

1.5 Frequency of Node by in Relation

The data is then grouped by *relation* and *x_type* to identify unique combinations of these values. For each group, the count of rows is computed. This provides the overview of node occurrence for each relation. The table shows how reflexive relations, such as *anatomy_anatomy*, only involve one node type as expected. It also demonstrates that other relations indicate the two node types they govern. Exceptions are *indication*, *contraindication*, and *off-label use*. These relations all connect nodes of type *disease* and *drug*. The less explicit naming convention here requires domain knowledge for meaningful interpretation. Chandak [1, p.2] outlines that these relations are included in order to determine 'how drugs target disease-associated molecular perturbations'. The following table 1.5.

Table 1.5: Counts of entities grouped by relation and type.

Relation	x_type	Count
anatomy_anatomy	anatomy	28064
anatomy_protein_absent	anatomy	19887
anatomy_protein_absent	gene/protein	19887

Continued on next page

Relation	**x_type**	**Count**
anatomy_protein_present	anatomy	1518203
anatomy_protein_present	gene/protein	1518203
bioprocess_bioprocess	biological_process	105772
bioprocess_protein	biological_process	144805
bioprocess_protein	gene/protein	144805
cellcomp_cellcomp	cellular_component	9690
cellcomp_protein	cellular_component	83402
cellcomp_protein	gene/protein	83402
contraindication	disease	30675
contraindication	drug	30675
disease_disease	disease	64388
disease_phenotype_negative	disease	1193
disease_phenotype_negative	effect/phenotype	1193
disease_phenotype_positive	disease	150317
disease_phenotype_positive	effect/phenotype	150317
disease_protein	disease	80411
disease_protein	gene/protein	80411
drug_drug	drug	2672628
drug_effect	drug	64784
drug_effect	effect/phenotype	64784
drug_protein	drug	25653
drug_protein	gene/protein	25653
exposure_bioprocess	biological_process	1625
exposure_bioprocess	exposure	1625
exposure_cellcomp	cellular_component	10
exposure_cellcomp	exposure	10
exposure_disease	disease	2304
exposure_disease	exposure	2304
exposure_exposure	exposure	4140
exposure_molfunc	exposure	45
exposure_molfunc	molecular_function	45
exposure_protein	exposure	1212
exposure_protein	gene/protein	1212
indication	disease	9388
indication	drug	9388
molfunc_molfunc	molecular_function	27148
molfunc_protein	gene/protein	69530
molfunc_protein	molecular_function	69530
off-label use	disease	2568
off-label use	drug	2568

Continued on next page

Relation	**x_type**	**Count**
pathway_pathway	pathway	5070
pathway_protein	gene/protein	42646
pathway_protein	pathway	42646
phenotype_phenotype	effect/phenotype	37472
phenotype_protein	effect/phenotype	3330
phenotype_protein	gene/protein	3330
protein_protein	gene/protein	642150

2 | Exploring the Knowledge Graph

2.1 The Ontology

2.1.1 Constructing the Ontology

An ontology provides a visual overview of the network. The ontology is defined as the set of unique pairs of node classes in the data set. The entire data set is iterated through in order to construct this set. This is not very time efficient, as the same information can be retrieved from grouping methods shown in 1.1. The method Iterrows() was initially used but proved very costly in time. The choice then to iterate over the data set using itertuples() was made, which sped things up.

2.1.2 Visualising the Ontology

The following network shown in Figure 2.1 represents the ontology of the PrimeKG data. The bidirectional encoding of the relations can be seen here by the arrows. This is due to the sets of pairs of nodes being defined as unique relative to order as well. A limitation of this graph is that edges are not displayed. Standard relations, such as *drug*, *disease*, can be inferred from the graph. However, reflexive relations are not shown. Furthermore, the previously mentioned drug and disease relations of *indication*, *contraindication*, and *off-label use* are not encoded in the graph. What can be gleamed from the graph, is the overarching structure of the data. The gene/protein node is central, from table 1.4, it is the protein more than the gene aspect of the class that relates to the other nodes. While other nodes such as *disease*, and *exposure* (referring to environmental exposure), are also well-connected. Further knowledge of biology and medicine would definitely help for a more in depth interpretation of the ontology.

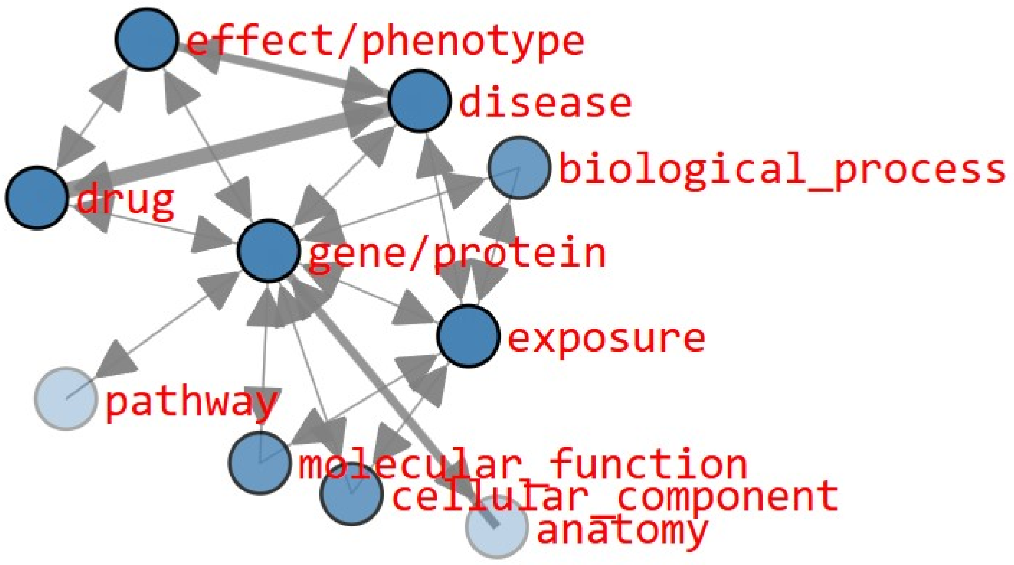

Figure 2.1: Ontology representation of the relationships between entities in the system.

2.2 Constructing the Knowledge Graph

2.2.1 Nodes

In order to construct a knowledge graph, the graph is populated from the data. The graph data structure is a dictionary. Where, each key is a node retrieved from the data in line 2 and 3 of the code shown in listing 2.1. Then, on line 6 a look-up dictionary is created in order to access node types given their name in constant time. Then in from line 9 to 12, all nodes are iterated, each node is then added to the graph with its respective type and an empty list serving as a placeholder for the nodes related to that node (its *relations*).

```
1  # TheGraph: {node: ([relations], type)}
2  node_names = kg['x_name']
3  TheGraph = dict.fromkeys(node_names, None)
4
5  # zip() combines into pairs
6  nodes_types_dict = dict(zip(kg['x_name'], kg['x_type']))
7
8  # For all nodes in data
9  for name in node_names:
10     type_corresponding_to_node = nodes_types_dict[name]
11     # Initialise empty list to later add relations
12     TheGraph[name] = ([], type_corresponding_to_node)
```

Listing 2.1: Code for Initialising the Knowledge Graph Data Structure and Populating it with Nodes

2.2.2 Edges

When iterating over the entire date set row by row in order to construct the ontology, another data structure *source_target* was also created. This data structure is a dictionary where the nodes x are the keys and their respective list of nodes related to them Y as values. These relations are now added to the graph object that was populated with nodes and types previously. This code is shown in listing 2.2. A problem encountered while populating the graph was that certain relations where the nodes name and type were the same caused key errors. As such, relations where this was an issue had *'_name'* appended to them to avoid this issue. This solution worked for the current data set but perhaps a more general solution would be more appropriate going forward.

```
# Add List of relations for each node, retrieve information from sorce_target(s) dictionary
for source in source_target:
    targets = source_target.get(source)
    # empty temp 'relations'
    relations, node_type = TheGraph[source]
    # populate relations
    relations = targets
    TheGraph[source] = (relations, node_type)
```

Listing 2.2: Code for Adding Related Nodes to Respective Nodes in the Knowledge Graph

2.3 Finding a Subgraph

The next task then was to generate a subgraph between two disease nodes. In order to find a suitable subgraph to visualise, certain criteria were outlined and encoded in the search. These criteria aimed to generate a suitably sized subgraph that wasn't too small leading to a lack of information or too big making inference difficult and increasing time complexity. Therefore, it was decided that no nodes in the graph would have more than 6 relations, and were to be 2 nodes in between the root node and the target node.

Depth First Search (DFS) and Breadth First Search (BFS) were considered as graph traversal algorithms. BFS was selected as it models better this idea of finding a subgraph where the two disease nodes are separated by 2 nodes as it expands its radius. DFS on the other hand risks exploring long lines of inference and then backtracking, increasing time complexity.

The subgraph shown in figure 2.2, has root node *progressive peripheral pterygium*, and target node *pseudopterygium*.

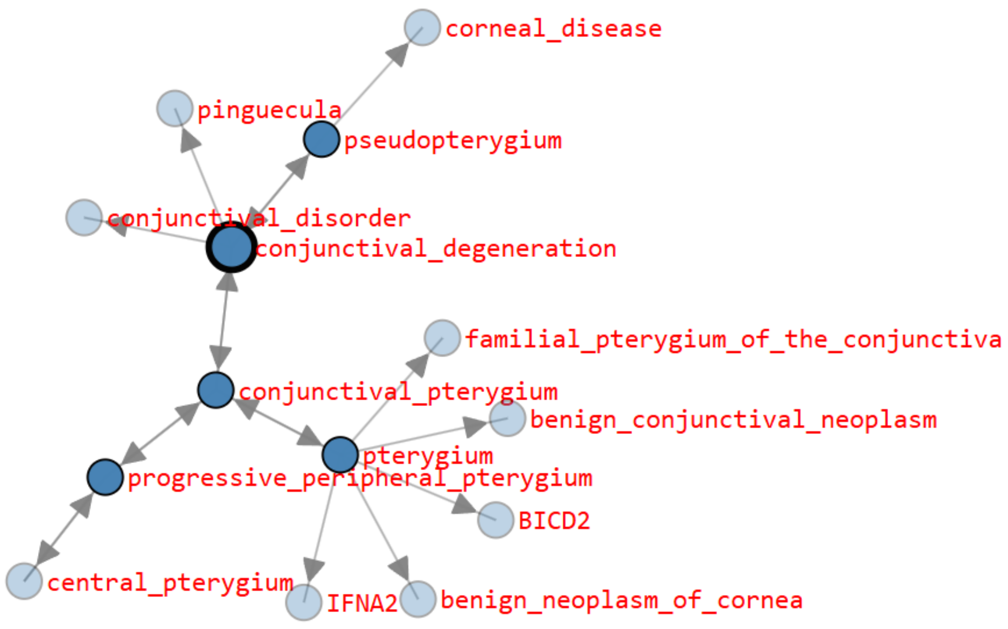
corneal_disease
pinguecula
pseudopterygium
conjunctival_disorder
conjunctival_degeneration
familial_pterygium_of_the_conjunctiva
conjunctival_pterygium
benign_conjunctival_neoplasm
pterygium
progressive_peripheral_pterygium
BICD2
central_pterygium
IFNA2
benign_neoplasm_of_cornea

Figure 2.2: A Subgraph Found Using BFS Given the Constraints

We can see that this subgraph pertains to the eye with nodes mentioning the cornea, conjunctival and ptergium.

3 | Deriving the Knowledge Base

3.1 Knowledge Base

The subgraph in shown in section 2.3 is a visual representation of a set of facts. This set of facts, called a knowledge base, can be directly understood from the graph. The graph was visualised using tuples in the form *(x,y)*. It was straight forward to convert these to natural language. The list of sentence and the graph can be viewed as equivelent.

```python
# subgraph tuples contains a tuples of source -> target pairs which can also be
   understood as rules like p -> q
# rule form:  'if the node is x then the node is connected to y'
def get_rules_from_tuples(tuples):
    rules = []
    for tuple in tuples:
        x = tuple[0]
        y = tuple [1]
        rules.append(f'If the node is {x} then the node is connected to {y}')
    return rules
```

Listing 3.1: Code for Adding Related Nodes to Respective Nodes in the Knowledge Graph

3.2 Inference

While these facts are explicitly represented by the knowledge graph, further relations can also be inferred. By the law of transitivity, if $(A \to B)$ AND $(B \to C)$, then $(A \to C)$. BFS can be used to forwards chain and traverse the graph. The inferred rule will then be $A \to C$, and the reasoning being $(A \to B)$ AND $(B \to C)$. The following shows 3 rules inferred using this logic. Rules are represented as tuples *(x,y)*, which means $x \to y$.

Rule: ('conjunctival degeneration', 'corneal disease')
Reasoning: (conjunctival degeneration → pseudopterygium AND pseudopterygium → corneal disease) IMPLIES conjunctival degeneration → corneal disease

Rule: ('conjunctival degeneration', 'pterygium')
Reasoning: (conjunctival degeneration → conjunctival pterygium AND conjunctival pterygium → pterygium) IMPLIES conjunctival degeneration → pterygium

Rule: ('conjunctival degeneration', 'progressive peripheral pterygium')
Reasoning: (conjunctival degeneration → conjunctival pterygium AND conjunctival pterygium → progressive peripheral pterygium) IMPLIES conjunctival degeneration → progressive peripheral pterygium

Given that the subgraph was created using no disconnected nodes and the bidirectionality of the relations, all nodes in the subgraph can be inferred as being connected.

4 | A Bayesian View of the Data

4.1 Bayesian Network

4.1.1 Topology

In order to create a Bayesian network, two things need to be defined. The topology of the network and the conditional distributions. This is enough information to account for the joint distribution of the network [2, p.493]. The joint distribution was for the following nodes: *disease, drug, gene/protein, anatomy*. To define this topology, the first step was to visualise the ontology to see how the nodes are connected. The subset of connections pertaining to the nodes in question was created, and the connections were visualised in the same way as for the ontology and the subgraph seen in part 3. This is shown in figure 4.1.

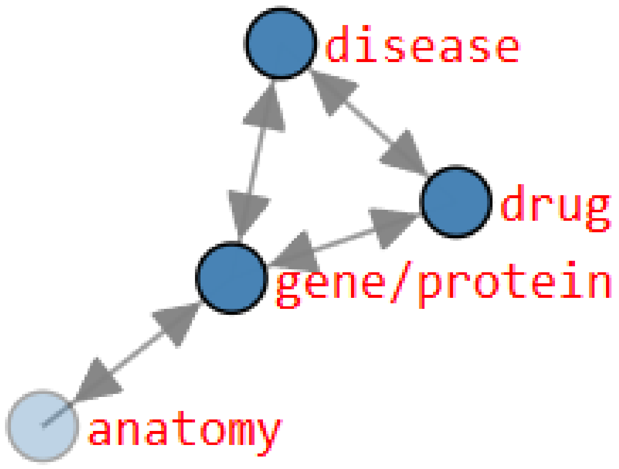

Figure 4.1: A Graph Showing the Relations of the Nodes with which the Bayesian Network is to be Created

Unlike this graph, the Bayesian network can not have bidirectional relations as these lead to cycles, whereas a Bayesian network is acyclic [2, p.493]. Therefore, the relations must be redefined as directed, representing that one node directly influences another. This is determined by domain knowledge.

I argue that a drug is dependent on the disease it is used to treat, this disease is dependent on an abnormality relating to a protein and finally the type of abnormality is directly influenced by the anatomical region, in which, it finds itself. The topology of the Bayesian network described is shown in figure 4.2.

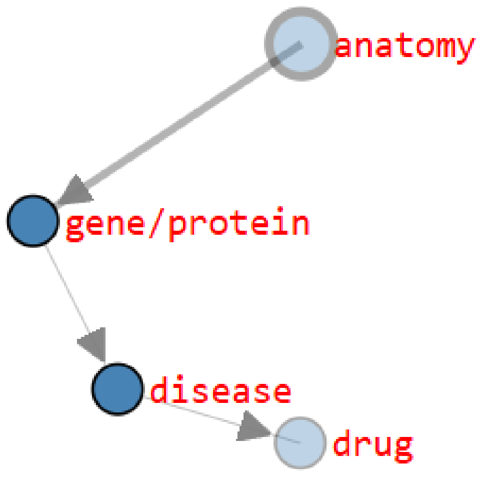

Figure 4.2: The Topology of the Bayesian Network

4.1.2 Deriving Probabilities

In order to calculate the joint distribution of this graph, the following needed to be calculated:

$$P(A, P, Di, Dr) = P(Dr|Di) \cdot P(Di|Protein) \cdot P(Protein|A) \cdot P(A),$$

where A is anatomical region, P is protein, Dr is drug, and Di is disease. The total count of relations is 8,100,498 The count of anatomy relations can be accessed using table 1.5 where the node type is anatomy.

$$\text{Count of anatomy relations} = 28,064 + 19,887 + 1,518,203 = 1,566,154$$

So the probability of a relation relating to anatomy can be represented as:

$$P(A) = \frac{\text{Count of anatomy relations}}{\text{Total count of relations}} = \frac{1,566,154}{8,100,498} \approx 0.1933.$$

This probability can be understood as the frequency anatomical regions appear in the data set.

Then protein given anatomy was calculated. The count of the intersection between protein and anatomy was calculated and divided by the total count of protein relations.

$$\begin{aligned}\text{Count of Protein relations} &= 1,538,090 + 80,411 + 25,653 + 144,805 + \\ &83,402 + 69,530 + 42,646 + 3,330 + 642,150 \\ &= 2,629,017\end{aligned}$$

The intersection:

$$\text{Count of Protein-Anatomy relations} = 19,887 + 1,518,203 = 1,538,090$$

So, therefore:

$$P(Protein|A) = \frac{\text{Count of Protein-Anatomy relations}}{\text{Count of Anatomy relations}} = \frac{1,538,090}{1,566,154} \approx 0.9821.$$

This is methodology is then repeated for the other two conditional probabilities.

The count of relations between drug and disease:

$$\text{Count of relations between Dr and Di} = 30,675 + 9,388 = 40,063$$

The count of all relations where disease is present:

$$\begin{aligned}\text{Count of all relations where Di is involved} &= 30,675 + 64,388 + 1,193 + 150,317 \\ &+ 80,411 + 9,388 + 2,568 + 2,304 \\ &= 341,244\end{aligned}$$

The conditional probability: $P(Di|Protein)$:

$$P(Di|Protein) = \frac{\text{Count of Disease-Protein relations}}{\text{Count of Protein relations}} = \frac{80,411}{2,629,017} \approx 0.0306.$$

Then, finally, $P(Dr|Di)$ was calculated:

$$P(Dr|Di) = \frac{\text{Count of relations between Dr and Di}}{\text{Count of all relations where Di is involved}} = \frac{40,063}{341,244} \approx 0.1174.$$

The joint probability is given as:

$$P(A, P, Di, Dr) = P(Dr|Di) \cdot P(Di|Protein) \cdot P(Protein|A) \cdot P(A),$$

$$P(A, P, Di, Dr) = 0.1174 \cdot 0.0306 \cdot 0.9821 \cdot 0.1933 \approx 0.000687.$$

These probabilities can now be added to complete the visual representation of the Bayesian network shown in figure 4.3.

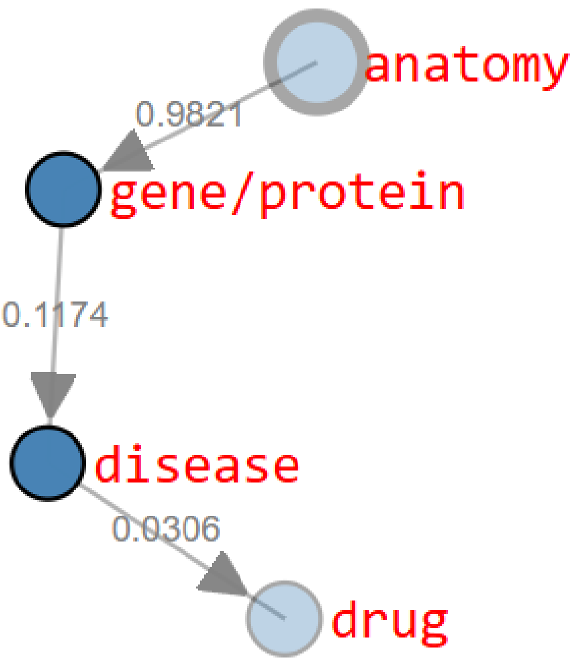

Figure 4.3: Bayesian Network With Conditional Probabilities Derived from the Data

4.2 Finding Anatomical Region Associated to Drug

To find the most likely anatomical region to be associated with a given drug relies on the calculation of conditional probabilities for instances of each class. Given their tendency to co-occur in the data, likelihood can be inferred. The frequency of unique relations was therefore calculated. Only 2 relations, out of the over 8 million, co-occured more than twice. It is possible therefore that the data set provided, while consolidating a lot of information, does not have predictive ability as there is very little variation of co-occurrence counts of instances of classes.

Bibliography

[1] Chandak, P., Huang, K., & Zitnik, M. 2023, Scientific Data, 10, 67. https://doi.org/10.1038/s41597-023-01960-3

[2] Russell, S. J., & Norvig, P. 2003, Artificial Intelligence, A Modern Approach, (Second Edition) (Prentice Hall)