Computer Science Honours Final Paper
2019

Title: Bayesian Networks in Neo4j: a Case Study

Author: Edwin Kassier

Project Abbreviation: Neo4j4us

Supervisor(s): Associate Professor Sonia Berman and Associate Professor Deshen Moodley

Category	**Min**	**Max**	**Chosen**
Requirement Analysis and Design	0	20	20
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	0
System Development and Implementation	0	20	20
Results, Findings and Conclusion	10	20	10
Aim Formulation and Background Work	10	15	10
Quality of Paper Writing and Presentation	10		10
Quality of Deliverables	10		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	0
Total marks		**80**	**80**

Bayesian Networks in Neo4j: a Case Study

Edwin Kassier
Department of Computer Science
University of Cape Town
Cape Town, Western Cape,
South Africa
KSSRUB001@myuct.ac.za

ABSTRACT

This project investigated the viability of using the graph database Neo4j as a medium for storing Bayesian Network (BN) machine learning models. The study included the development of an encoding protocol for storage of Bayesian Networks in such a database, as well as the protocols needed to retrieve and reliably recreate a chosen Bayesian model in a local environment at runtime. Additionally, the study provides a description of the planning and development of a foundational unified system made up of these components; capable of storing, retrieving and then performing inference using a given BN, this inference being made possible by user provided evidence. This is followed by describing the development of specialised tools using this foundational system as its source, with the purpose of providing a useful set of functionalities to the University of Cape Town's Computer Science department, using their network of choice as a focus. The goal of this given network is to reduce student attrition rates and thus increase aggregate student performance through identifying students at risk of not completing their degrees in the correct time frame. Through the construction of these software sets, Neo4j was proven to be an adequate medium for storing BN models, with the native structure of Neo4j databases mapping well to the directed graph structure of these models. However, further future exploration is required to fully realise the benefits of a Neo4J database.

KEYWORDS

Neo4j, big data, machine learning, risk analysis, Risk classification

1. INTRODUCTION

This project was formulated as a result of the University of Cape Town's Computer Science department who wanted to explore the viability of using a graph database, in this case Neo4j[10], to store Bayesian Network machine learning models. This was to be followed by exploring how such a graphical database and its graph algorithm driven query engine could be leveraged to facilitate: (i) data storage, (ii) understanding previously stored models, and (iii) retrieving the model to perform inference.

1.1 Project Significance

Due to recent advances, and subsequent proliferation of machine learning tools and methodologies for a wide array of use cases, the ability to access and use these models has become increasingly esoteric for those developers who understand the models and their use, or those who have access to developer tools or software to utilise and understand the model and its inner workings. At present, machine learning models have been commoditised to function only on the platforms on which they were created, leading to a lack of access for important models for those unable to access the platforms the models were created on.

Attrition rates at tertiary education institutions have always been an area of concern as they do not only affect the students that decide to deregister from the programme, but also those remaining in the programme. Deregistration of students does not only affect the reputation of the department in question, but also adversely affects the students left in the programme as departmental resources have already been expended on students who, had they been identified earlier, could have been transferred to the extended degree programmes. Thus, freeing up resources that can then be used to increase the likelihood of success of students remaining in the programme. Maximising student pass rates is especially vital in degrees with high attrition rates such as those in the STEM field where the need for these scarce skills are imperative to the continued growth of important public sectors [5].

The significance of this project lies in it providing increased access to important machine learning models that can in turn be used to increase pass rates in STEM degrees.

1.2 Project Aims

Due to the nature of this project, it was decided to modularise the aims to ensure that should any issues arise during its development, or unforeseen circumstances occur in the project timeline, it would not result in the delivery of an incomplete or non-functioning product. Therefore, the project aims were divided into the following two modules, with each being able to exist as a modular system, but together, culminating in a project fulfilling the full scope of the desired system.

1. Develop Neo4j queries to store and then retrieve the relevant model data at runtime, and

2. Parse data into inference engine at runtime to provide the relevant results and reasoning behind the given result.

In addition to the aims outlined above, it was decided to include stretch aims, so that (time permitting), it would result in a system that met the needs of the project, as well as increasing its usefulness in a production environment. Hence, these aims firstly

focused on creating a basic interface for users to interact with when using the tool. Secondly to exploring avenues to leverage Neo4j's query engine to more effectively store and retrieve model data, as well as performing inference on the chosen model. Finally tailoring all these parts into a tool aimed at providing a risk analysis toolset for CS lecturers or CS students.

1.3 Structure of Report

This report has six major sections. Section one gave a project overview, its significance and aims of its final iteration. Section two provided project background, as well as the main mechanisms that were used in development, through a summary of the literature review. Section three detailed the gathering of requirements that took place before project inception, as well as designing the relevant artefacts that would satisfy the aims of the project. Section four discussed the development process, including descriptions of major functionality algorithms and their outputs. In addition, this section described the different modules that make up the final system and how they were developed. Section five reviewed the project as a whole, and addressed the project aims that were met and those that were not. Finally, section six outlined the limitations of the current project as well as any improvements/extensions that future projects of a similar nature could use this project as a basis for.

2. LITERATURE REVIEW & BACKGROUND

2.1 Bayesian Networks

A Bayesian network, in its truest sense, functions as a Directed Acyclic Graph (DAG) where one node per random variable of the dataset exists. Each node contains a separate conditional probability table that catalogues the probabilities of its occurrence, depending on all outcomes from its parent nodes. The above is performed using the following Bayes theorem as its basis:

$$P(A \mid B)=(P(B \mid A)*P(A))/(P(B))$$

Bayes theorem is a mathematical function that can predict the probability of an event A occurring knowing that an event B occurs. To illustrate the above, the example of three variables in a Bayesian DAG that all have the possibility of being true or false is used: Grass is wet, it is raining, and the sprinkler system is on. "Raining" and "Sprinkler system is on" are parent nodes to "Grass is wet" as they have the potential to affect the probability of it being true or false, depending on their own values. This implies that "grass is wet" will have a conditional probability table cataloguing its own probability of being wet, depending on these parent variable values. This classification of relationships between edges is only possible in a select number of simple problems where an expert could reliably provide the relationships between nodes. Bayesian networks then need some mechanism for reliably learning the relationships between large numbers of nodes to ensure reliable prediction for future questions in complex datasets. Bayesian networks perform this by a process called structural learning, where, broadly speaking, the algorithm will infer the relationships between nodes to establish how the conditional probabilities between nodes should change depending on the training data. Once this is completed the system may make predictions using a set of evidence values in a process called inference. Inference uses posterior probability as a method of generating a probability of some event given a set of evidence.

2.2 Neo4j and its use in Machine Learning

Neo4J is a graph database system that follows in the footsteps of the relatively recent NoSQL trend. It stores information in directed graphs, in an attempt to overcome the inefficiencies that large relational databases incur due to the high number of potential join operations within the database.

In an empirical study into the performance differences between twelve separate Neo4J and MySQL databases, it was shown that relational databases are best suited to storing and querying relatively small databases, with Neo4J performing only slightly slower in the small-scale tests [11]. However, as the size of the databases being used were scaled up, there was a dramatic change in the levels of performance between the two database types, with Neo4J performing queries ten times faster (in some cases) than its MySQL counterparts.

At present, the literature, does not document any specific challenges that a particular machine learning algorithm, let alone machine learning in general, might experience when integrating graph databases into the machine learning workflow. There is only mention of the increased efficacy a graph database brings to data mining and machine learning due to the reasons described above [8].

2.3 Cypher

Cypher is Neo4j's answer to the standard SQL language. It is an open source declarative language derived from SQL that allows users to state what they want to access and how they want to access it from a Neo4j database in a relatively simplified format and syntax compared to standard SQL.

3. REQUIREMENT ANALYSIS AND DESIGN

3.1 Requirement Gathering

To ensure the maximised utility of this tool both for its current intended users as well as for any other users' future extensions it may allow for, the requirements of the project were reviewed on a weekly basis. This was followed by retrieving the requirements incrementally to ensure that a secure base of functionalities was in place before shifting focus onto other functionalities, allowing for a modular construction of the final product

The requirement gathering in this project took the form of interviews with the main stakeholders of the final product. The main stakeholders being the supervisors of this project who are also the Computer Science lecturers who would be using this set of tools in the course of their analyses of first year students in the department.

Through these sessions, the following requirements and their related priorities were gathered as outlined in Table 1 below

Table 1: Project requirements and related priorities

Requirement	Priority
Develop encoding protocol for Bayesian Models	High
Develop storage and retrieval protocol of Bayesian Models to and from Neo4j	High
Develop decoding protocol for Bayesian Model	High
Develop parser/validator for raw Bayesian Model files	High
Perform Inference using specific Bayesian Network as use case	High
Develop unified program to showcase developed functionalities	High
Develop CS use case to batch analyse csv files and update Neo4j student database	Medium
Develop CS use case to allow students/lecturers to access a specific student's Bayesian test results using a student ID number as the query value	Medium
Develop version to analyse the efficacy of given use case in providing accurate predictions	Medium
Develop UI	Low

3.2 Stakeholder Analysis

In addition to extracting a list of project requirements from the aforementioned interview sessions, various potential project stakeholders were also discussed and explored, as well as their respective interests in the final product. After identifying these stakeholders and their power within the project, their interests were prioritised as an aid in identifying the respective project aims that should receive priority. The full figure outlining the stakeholder analysis can be found in supplementary Table 1 in the appendices section.

3.3 Ethical, Professional, and Legal Issues

Conversations with the project sponsors revealed that the only ethical consideration for the project was its source and use of any personal student data.

As the test/demo data provided for the project is anonymised and has been the focus of numerous past projects its use has already received ethics approval from the UCT Ethics in Research Committee.

The project is a proof of concept and hence uses anonymised data. However, if such a system were to be used in practice, it would need access to a copy of actual UCT student records. Therefore, it was decided to add additional security to the system by not showcasing any mark values in plain text (the actual numerical values) when showcasing the results of an inference test. This was done as it would be possible to use the system as an avenue to access a student's academic history if an attacker knew a student's full student number.

3.4 Modular Construction

As previously mentioned, the development of the final product was broken down into modules and protocols that could function as standalone programs. These programs would then function with the help of any relevant, previously developed modules that contributed to it. The staging of the final product's development was broken down into three distinct stages:

Stage 1 focused on creating the building blocks of the system, achieving the basic aims set out by the project sponsors and creating integration points for these building blocks, allowing them to be connected when integrating these blocks into future iterations.

Stage 2 focused on developing a general system that while not meeting the requirements of the stretch goal of the system, i.e. being a system tailored towards use with CS lecturers and potentially students, it did provide a general proof of concept for a Neo4j Bayesian Network powered system, as well as providing all the building blocks for future iterations. Meaning that any future studies into the use of Bayesian Network powered tools could utilise the stage 2 artefact as a basis for their code and simply customise it to their needs

Stage 3 focused on providing all the stretch goals set out by the project sponsors. This involved tailoring the stage 2 artefact into 3 separate programs that achieved these different goals.

The development process of these stages and their included modules is depicted in Supplementary figure 2 in the appendix section.

3.5 Activity Diagram of Full System

Seeing as the stage 2 general system artefact will be the basis for future iterations, the following activity diagram displays the set of functionalities a user can interact with, as well as the steps involved in these functionalities within the system.

Figure 2: General system activity diagram

4. SYSTEM DEVELOPMENT AND IMPLEMENTATION

4.1 Background

Discussions with the project supervisors indicated that while the system was to be designed to handle any Bayesian Network model to showcase the system as a proof of concept, the AtRIsk network developed by a previous UCT study [7] was to be the case study for this work. This study, performed by Nudelman et al. was undertaken to develop a Bayesian Network capable of identifying first year students at risk of failing their major courses, in this case their CS and MAM courses. Therefore, all example figures will be in reference to this network to showcase the functionalities of the systems design.

4.2 Technologies and Libraries

This project was developed using python as its language of choice. The reason behind Python's use is twofold, in that it: (i) provides integration with many well-maintained libraries that provided the project the ability to develop in rapid iterative cycles; and (ii) provides flexible creation of new code free of the relative rigidity of other languages such as Java. However, while the use of Python does allow for rapid development of software, it lacks the reliability of Java due to the increased effort required to perform unit tests as it does not have the well-known system of junit tests attached to it. Therefore, necessitating the use of a suitable alternative.

In terms of external libraries used the list is as follows:

- pgmpy[1] - pgmpy is a Python machine learning library that provides functionalities to create probabilistic graphical models, train them and then finally perform inference using them all within a local programming environment.
- Pandas[2] - Pandas is an open source library providing a toolset allowing for the construction and analysis of large datasets within a Python programming environment. It is able to read data from files such as .csv into the program and then allows the program to manipulate and update this read-in data.
- py2neo - py2neo is a Python library that simplifies the connection and interaction process with a Neo4j database (including methods to perform basic operations such as adding or retrieving nodes as well as the ability to execute custom cypher queries). It was built and is maintained by Neo4j itself, meaning all interactions and methods are kept up to date as the system is updated.

4.3 Historical Data

For the creation of a student database for use in this project, a csv file of historical student transcript, high school and (where applicable) NBT results was provided. This dataset has been fully anonymised and holds UCT CS student data from 2007 to 2016.

4.4 Raw Bayesian Network Model Data Parsing

At present, an industry wide accepted standard format for representing Bayesian Network models does not exist[3]. Work has been done on developing such a format in the form of the Bayesian Interchange Format (.BIF)[3]. However, this format has failed to gain traction and widespread use in popular Bayesian Network modelling tools such as Netica and BayesServer. As

such, each tool has developed their own flat file formats for storing Bayesian Network models developed in their system. This, in conjunction with a lack of methods to convert these tool specific flat files to the .BIF format means that any system hoping to utilise already trained Bayesian Network models, must have parsing systems in place to deal with each of these tool specific flat file formats. In the case of the current system, the predominant use case is a model trained and saved in Netica's .DNE format[4]. An example of a Bayesian Network node within one of these flat files is showcased in Figure 3 below.

```
node AveSciBin {
        discrete = TRUE;
        states = (High, Mid, Low);
        kind = NATURE;
        chance = CHANCE;
        parents = (Aptitude, WorkEthic);
        probs =
                // High       Mid        Low        // Aptitude WorkEthic
                  (0.7650293,   0.1217741,   0.1131966,   // High    Low
                   0.2748595,   0.5955484,   0.1295921,   // High    High
                   5.43732e-7,  8.747693e-6, 0.9999907,   // Low     Low
                   0.4279719,   0.3340286,   0.2379995);  // Low     High    ;
        numcases =
                            // Aptitude WorkEthic
                  (155.2048,    // High    Low
                   50.53888,    // High    High
                   399.5363,    // Low     Low
                   131.7201);   // Low     High    ;
        title = "AveSciences";
        whenchanged = 1521500890;
        belief = (0.2551749, 0.126044, 0.6187811);
        visual V6 {
                center = (360, 252);
                height = 5;
                };
        };
```

Figure 3: Raw .DNE file node format

The parsing system functions in two phases. In the first phase the system will extract the entire block of text as shown in Figure 3 above, by using bracket matching and searching that focuses on a "node" keyword used to identify node blocks of text. Thereafter the systems stores this block of string in a list to be passed onto the second phase of parsing which will extract the relevant information and then encode it into the standard form for Bayesian Network nodes as set out by the system. This second phase will be explained in the subsequent encoding Bayesian Networks section.

4.5 Encoding Bayesian Networks

After in- depth research into the functionalities provided by the top three most popular inference engines for Bayesian Networks in Python (pgmpy[1],Pomegranate[9] and BayesPy), it was possible to identify the minimum amount of information needed from any flat file to effectively store, retrieve and then reconstruct Bayesian models to perform inference. These fields are:

1. Belief Table (including the nodes possible states)
2. Conditional Probability Table
3. Conditional Probability Table states (Being the states the parents can take on)
4. Name of the node
5. Name of parent node/s
6. Possible states the node can take on

Knowing the information needed from any flat file, the second phase of the raw .DNE parsing first creates dictionaries for the relevant headings and then extracts the required information from each node string block using a similar method to the first phase by using the relevant tags and bracket matching to differentiate the different sections within the string block. After extracting the relevant information from the string block, the values are then added to their respective dictionaries as shown in Figure 4 below.

```
{"bt": {"High": "0.2551749", "Low": "0.6187811", "Mid": "0.126044"}}

{"cpt": [[0.7650293, 0.1217741, 0.1131966], [0.2748595, 0.5955484, 0.1295921], [5.43732e-07, 8.747693e-06,
0.9999907], [0.4279719, 0.3340286, 0.2379995]]}

{"cpt_states": [["High", "Low"], ["High", "High"], ["Low", "Low"], ["Low", "High"]]}

{"name": "AveSciBin"}

{"parents": ["Aptitude", "WorkEthic"]}

{"states": ["High", "Mid", "Low"]}
```

Figure 4: Intermediate dictionary form of Bayesian node

The reason for using dictionaries is twofold. Firstly, it allows for ease of access when trying to use stored data, but also reduces the risk of accessing information from the incorrect place, as the identifying name of the dictionary must be used in order to access any data within it, therefore ensuring the correct information is being accessed and used. After having stored the relevant information in the aforementioned dictionaries, the dictionaries are then concatenated into one large dictionary which is subsequently converted into a string Json representation. The reason for producing this is because when reproducing the network into a local representation, the inference system must have all this node information or none of it, as it is an atomic piece of information and must therefore be stored atomically. An example of this json representation is shown in Figure 5 below. However, the un-concatenated dictionaries are still retained for use in later tasks.

```
{"bt": {"High": "0.2551749", "Low": "0.6187811", "Mid": "0.126044"},

"cpt": [[0.7650293, 0.1217741, 0.1131966], [0.2748595, 0.5955484, 0.1295921], [5.43732e-07, 8.747693e-06,
0.9999907], [0.4279719, 0.3340286, 0.2379995]],

"cpt_states": [["High", "Low"], ["High", "High"], ["Low", "Low"], ["Low", "High"]],

"name": "AveSciBin",

"parents": ["Aptitude", "WorkEthic"],

"states": ["High", "Mid", "Low"]}
```

Figure 5: Form 3, Json representation

4.6 Storing Encoded Bayesian Networks

Having created the Json representation of the nodes data, the system then proceeds to start constructing the Bayesian model within Neo4j. This process takes the form of constructing nodes individually from each other with three attributes each, these being: (i) the ID of the node within Neo4j (for indexing purposes), (ii) the data of the node, being the Json representation of the node mentioned previously, and finally, (iii) a name, which is needed when linking nodes in the Neo4j representation of the network as well as acting as a method of verifying the data stored within the nodes json representation

<id>:
469
data:
{"bt": {"High": "0.2551749", "Low": "0.6187811", "Mid": "0.126044"},
"cpt": [[0.7650293, 0.1217741, 0.1131966], [0.2748595, 0.5955484, 0.1295921], [5.43732e-07, 8.747693e-06,
0.9999907], [0.4279719, 0.3340286, 0.2379995]],
"cpt_states": [["High", "Low"], ["High", "High"], ["Low", "Low"], ["Low", "High"]],
"name": "AveSciBin",
"parents": ["Aptitude", "WorkEthic"],
"states": ["High", "Mid", "Low"]}
name:
AveSciBin

Figure 6. Bayesian Node attribute data

After the creation of all relevant nodes, the system then proceeds to set up relationships between nodes. In this case, the relationship name "Affects" was chosen to describe the relationship between nodes. For example, node A may affect node B. This relationship represents the effect one node's probability values may have on another. An example of this might be the node describing the probability of rain directly affecting the probability of the grass being wet. This relationship is created using the cypher query below, alongside the data from the parent dictionary for the node in question. If there are no parents attributed to a node, this command will not be executed. This concept is illustrated in Figure 7.

```
1. MATCH (a:{name}), (b:{parent})
2. WHERE a.name = "{name}" AND b.name = "{p
   arent}"
3. CREATE (b)-[: Affects]->(a)
```

Figure 7: Neo4j query to create Bayesian Nodes

After having created the nodes and the relationships between them, the given Bayesian Network is fully stored within Neo4j and is ready to be retrieved for later use.

It can be observed what this would look like, given the use case of the AtRisk network provided from a previous UCT study[7]. See Figure 8 below for the representation of this network within Neo4j.

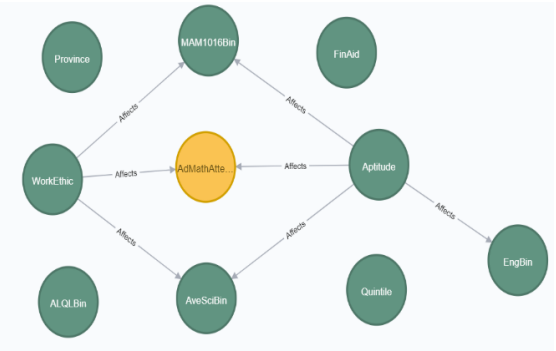

Figure 8: Full Bayesianl network in neo4j database

4.7 Retrieving Encoded Bayesian Network Models

When querying the database in order to extract a model, the entire Bayesian Network is retrieved and its data components converted back into a large dictionary, as can be seen in Figure 5. Thereafter, the data is named and placed into a dataframe created using the Python Pandas library using the built in .to_data_frame() function. This provides an intermediate form of the data for the program for future use in the construction of the final local representation.

In addition, through experimentation, it was found that any nodes that did not possess an existing relationship between it and some/child/parent node did not qualify for use in the network as their inclusion did not affect the outcome of any inference being performed. As such, if a node did not have a child or parent related to it, the node was dropped from the dataframe and excluded from further use. After performing this operation, the dataframe is left with the functional network that may then be used to construct a local representation for an inference engine to work on.

An example of this can be seen using the AtRisk model mentioned previously. This model was constructed using many nodes that were hypothesised to be relevant in identifying at risk students (such as the students' schooling background as well as their receipt of financial aid). However, in the process of determining the structure (i.e. learning which nodes are related to one another), it was discovered that four of the ten nodes did not have a relationship with any other nodes in the model. This meant that when performing inference using the model, these unconnected nodes did not affect any prediction outcome and thus were not needed when retrieving the model for inference. As can be seen in Figure 9 below, the functional model of the given use case can be seen with all non-relevant nodes excluded. When comparing this to Figure 8 shown below, it is evident how the system filters unconnected nodes out of the model from Neo4j.

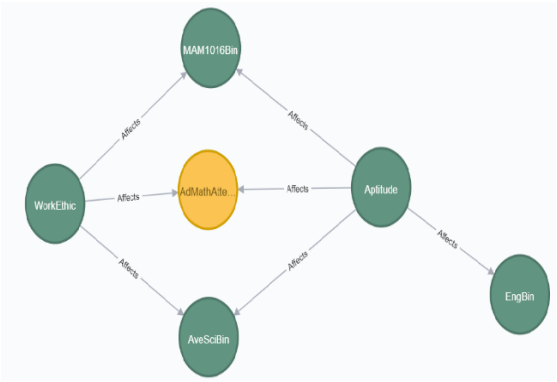

Figure 8: Functional Bayesian Network in Neo4j

4.8 Constructing a Local Representation of a Bayesian Network Model

In the process of formulating the rules for constructing a local representation of the Bayesian Network from the Neo4j data that was compliant with the used inference engine, it was discovered that the order in which nodes are added into the local network, must be done in such a way that the parents of any node must exist before a node is allowed to be added to the local network. This is important to ensure that the relationships between nodes and their parents lead to pre-existing nodes and not placeholder values. If this is not performed correctly, the inference engine is not able to solidify the topology of the network and thus is not able to perform inference.

This system uses the python library pgmpy as a provider for an environment in which a local representation of a Bayesian network can be created, as well as an inference engine that can perform inference on the given network within the local environment.

4.9 Performing Inference

The pgmpy library has three available methods for performing inference on Bayesian Networks. These include: 1. Variable Elimination (VE), 2. Belief Propagation (BP), and 3. Max-Product Linear Programming (MPLP). The current system utilises Variable Elimination as its method of inference. The reason for this is due to its generality and its ease of application onto typical Bayesian Networks [6]..

4.10 System Versions

4.10.1 Full Version

Due to the multitude of potential use cases that a system such as the current one could be catered to, it was decided to first develop a general system that could accept any valid .DNE flat file and allow the user full freedom in choosing their focus for inference, as well as allowing them the ability to provide all relevant evidence for the inference by hand, in line with the other nodes within the network. While also being able to perform inference on a given .DNE file, this system is also able to add these files to a curated library of other Bayesian Models that the user may choose to work with at runtime via a selection process through the main UI. This general system then provides a basis for the further development of more specialised systems to be described in the headings below.

4.10.2 Student/Student Batch Version

The student version of this system is a specialised version of the generic system and was designed to function using the AtRisk Bayesian Network .DNE model provided by a previous study that focused on identifying at risk Computer Science students at UCT by Nudelman et al[7]. The purpose of this version is to provide a tool to calculate and then store the risk probabilities for each student in a given set, as well as the reasoning behind their respective risk values. In addition, this version of the system is able to retrieve a student's risk results when given a student number. This is achieved by using the Neo4j student database created by the project partner Josh Redelinghuys, as a foundation and then performing the modifications described above. Using this modified database, it is possible to retrieve a specific student's inference test results using the following query.

```
1. MATCH (s:Student{StudentID:StudentNumber
   })-[:HAS_RISK]->(bn:BayesianFact)
2. return (bn)
```

Due to the lack of required information in the existing Neo4j databases needed for evidence to perform the relevant inference, it was decided to use the full data file of historical student data to calculate student risk and explanation values. This was followed by updating the main student database with another node containing all the risk and explanation values. This operation is performed using the following code:

```
1. USING PERIODIC COMMIT 500
2. LOAD CSV WITH HEADERS FROM "file:///Baye
   sianFacts.csv" AS csvLine
3. MATCH (s:Student {StudentID: toInteger(c
   svLine.StudentID)}), (bn: BayesianFact {
   BayesianID: toInteger(csvLine.BayesianID
   )})
4. CREATE (s)-[:HAS_RISK]->(bn)
```

4.10.3 Student Analysis Version

Using the updated Neo4j database described above, the last version of the system is a tool that can be used to analyse the effectivity of the AtRisk network against the historical dataset of students provided. However, this version could be used as a proof of correctness for any model provided to it. In addition, It is an extension of the student/student batch version described above, as this version will query every student in the database that has a risk and explanation value related to it, and thereafter performs checks on whether the risk value is consistent with the student's future performance. This system will also provide summarised statistics on the performance described above. Apart from providing an evaluation of the given model, the system can also generate other queries that a lecturer may find useful. This includes the allocation of students to groups with high aptitude and low work ethic so that when assignment groups are being made, a combination of student types within groups can better utilise the hard-working students as well as those with an intuitive sense of the subject matter to ensure a higher aggregate performance across the class.

5. CONCLUSIONS

Achievement of aims

The project has accomplished the aims set out by the project supervisors. This was achieved through the construction of multiple standalone modules, with each module addressing a specific aim. These modules when integrated, met the requirements of the final system.

The first major module was developed by distilling the Bayesian Network model structure into a generic set of Bayesian attributes that can be extracted from a given .DNE file. This data when paired with a protocol that uses a set of database commands, allows the dynamic construction and storage of Bayesian Network models within a Neo4j database. In addition, once the Bayesian Network is stored within Neo4j, it was possible to visually inspect the network and its structure within Neo4j's data browser, allowing for a deeper understanding of how a given model works.

The second major module was developed through the use of a protocol that retrieved Bayesian Models from a Neo4j database and then reconstructed it into a local representation that the chosen inference engine (pgmpy) can act on when performing inference. This local representation of a model could then be provided with evidence to perform a prediction using pgmpy's inference engine.

To prove the practicality of the current implementation for use as a tool by UCT CS lecturers, the two major abovementioned modules, were integrated into a singular system that could be used through a command line UI interface.

Having then created a generic system, it was possible to create specialised tools for a chosen use case. In this instance, it was the AtRisk network developed by Nudelam et al[7]. Using this network as a focus to produce a set of tools that are able to analyse the use case model's efficacy in providing predictions, performing specialised queries such as grouping students into teams consisting of a combination of students based on aptitude and work ethic and finally, allowing for the querying of the student database for a specific student's risk value, using student number as the query value.

Neo4j viability

Neo4j proved to be an adequate medium for storing Bayesian Networks, as it provided a set of native structures that mapped well to the structure of Bayesian models. However, beyond the ease of construction Neo4j provides, the advantages of using Neo4j over other databases is not apparent. Further exploration of how Neo4j can be used in the context of storage and use of Bayesian Networks, is needed to fully ascertain how well the Neo4j structure and query engine can be used in this context.

6. LIMITATIONS

While the project achieved its aims, there were two constraints experienced that prevented the project from achieving its goals as they were formulated for the project's original scope, namely a singular integrated system where users may interact with all the relevant functionalities through one portal given any Bayesian Network model of any format. These constraints are discussed below.

Neo4j version

In the course of developing the student versions of the system, it was found that the free version of Neo4j only allows for one database to be running at any given time. As the required database had to be running before any of the developed systems could access the information within, the implication was that any of the programs could read and write data to either the Student Database or the Bayesian Network database. However, is was not possible to access both simultaneously. As the inference tests required the Bayesian Network database to be running, the calculation of Bayesian Test results for students required a local copy of the student database, this had to be retrieved beforehand from the Neo4j student database and then placed into a csv accessible by the inference program. Once this batch inference was performed, it was thereafter possible to start the student database and have a separate program update the main Neo4j student records with their Bayesian Test results by using this local file for reference.

Bayesian Network formats

As mentioned previously, the proposed standard for storing Bayesian Networks, through the use of the Bayesian Interchange Format (BIF), has failed to gain widespread use in popular tools such as Netica and others. As the use case given to the project was a model developed in Neticas .DNE format, the system was built to handle this file format. As a result, this project lacks the ability to store and use other potentially useful models developed in other formats.

7. FUTURE WORK

For the project to achieve its goal within the scope set out in its original form, the following areas will need to be addressed in future, in order to refine the program.

To .BIF Compiler

To increase the effective scope of models the inference system has access to, it would be highly beneficial to develop a tool capable of compiling Bayesian Models from their tool specific files (.DNE etc.) to the Bayesian Interchange Format. Thereafter, changing the parsing method within the Bayesian Network storage protocol to parse the Bayesian Interchange Format instead of the .DNE format.

Explore inference through the Cypher query engine

To complete the exploration into the use of Neo4j as a medium for the storage and use of Bayesian Networks, future iterations of the tools will require the exploration of performing inference within Neo4j itself. This will entail the creation of query sets that can isolate certain parts of the model when performing posterior probability, as well as coding an appropriate posterior probability calculation to use within these queries.

Creation of a GUI

As this set of tools is projected to be used by not only the CS lecturers, but the students as well, this necessitates the creation of a full GUI that a user could intuitively understand with minimal explanation. While not improving the functionality of the toolset, it increases its usability, which is equally important as creating a toolset that students would want to use in determining their own academic programmes.

Integration between this project and its counterpart

As this project forms half of a full system, with the other half being developed by the project partner Josh Redlinghuys, the aim is to provide a set of tools that could identify students at risk of not completing their academic programmes, as well as analysing planned curriculums, and highlighting any inconsistencies within the courses. Hence the integration of these systems would allow students the ability to have greater agency when deciding their academic futures.

8. ACKNOLEDGEMENTS

I'd like to thank my mother, Dr Suna Kassier, and my project supervisor, Ascsociate Professor Sonia Berman. Both of whom have made what should have been an extremely difficult situation, wrought with hospital visits and delays, into an educational experience of struggle and success that I won't soon forget.

9. REFERENCES

[1] Ankur Ankan and Abinash Panda. 2015. pgmpy: Probabilistic Graphical Models using Python. In *Proceedings of the 14th Python in Science Conference*, 6–11. DOI:https://doi.org/10.25080/majora-7b98e3ed-001

[2] Daniel Y. Chen. *Pandas for everyone : Python data analysis*.

[3] Carlos Chesñevar, Jarred Mcginnis, Sanjay Modgil, Iyad Rahwan, Chris Reed, Guillermo Simari, Matthew South, Gerard Vreeswijk, and Steven Willmott. 2006. Towards an argument interchange format. *Knowl. Eng. Rev.* 21, 4 (December 2006), 293–316. DOI:https://doi.org/10.1017/S0269888906001044

[4] Norsys software Corporation. Netica. Retrieved September 17, 2019 from https://www.norsys.com/netica.html

[5] Peter A. Daempfle. 2003. An Analysis of the High Attrition Rates among First Year College Science, Math, and Engineering Majors. *J. Coll. Student Retent. Res. Theory Pract.* 5, 1 (May 2003), 37–52. DOI:https://doi.org/10.2190/dwqt-tya4-t20w-rcwh

[6] Fabio Gagliardi Cozman. 2000. *Generalizing Variable Elimination in Bayesian Networks*.

[7] Zachary Nudelman, Deshendran Moodley, and Sonia Berman. 2019. Using bayesian networks and machine learning to predict computer science success. In *Communications in Computer and Information Science*, 207–222. DOI:https://doi.org/10.1007/978-3-030-05813-5_14

[8] Kaspar Riesen and Horst Bunke. 2008. IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. . Springer, Berlin, Heidelberg, 287–297. DOI:https://doi.org/10.1007/978-3-540-89689-0_33

[9] Jacob Schreiber. Pomegranate. Retrieved from https://github.com/jmschrei/pomegranate

[10] Neo Technology. *Neo4j Graph Database*. Retrieved September 17, 2019 from http://neo4j.com/product/

[11] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. 2010. A comparison of a graph database and a relational database. In *Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE '10*, 1. DOI:https://doi.org/10.1145/1900008.1900067

Appendix

Stakeholder	Interests	Stakeholder Influence	Stakeholder Importance
Supervisors	• Successfully achieve project aims • The development of robust and well documented code • Diverse set of functionalities available • Ability for future extension in capabilities of toolset	Very High	Very High
UCT CS Lecturers	• Tool usability • Accuracy of inference	High	High
Student	• Accuracy of inference	Low	High
Team Members	• Meeting supervisor expectations	High	Low

Table 1. Stakeholder analysis table

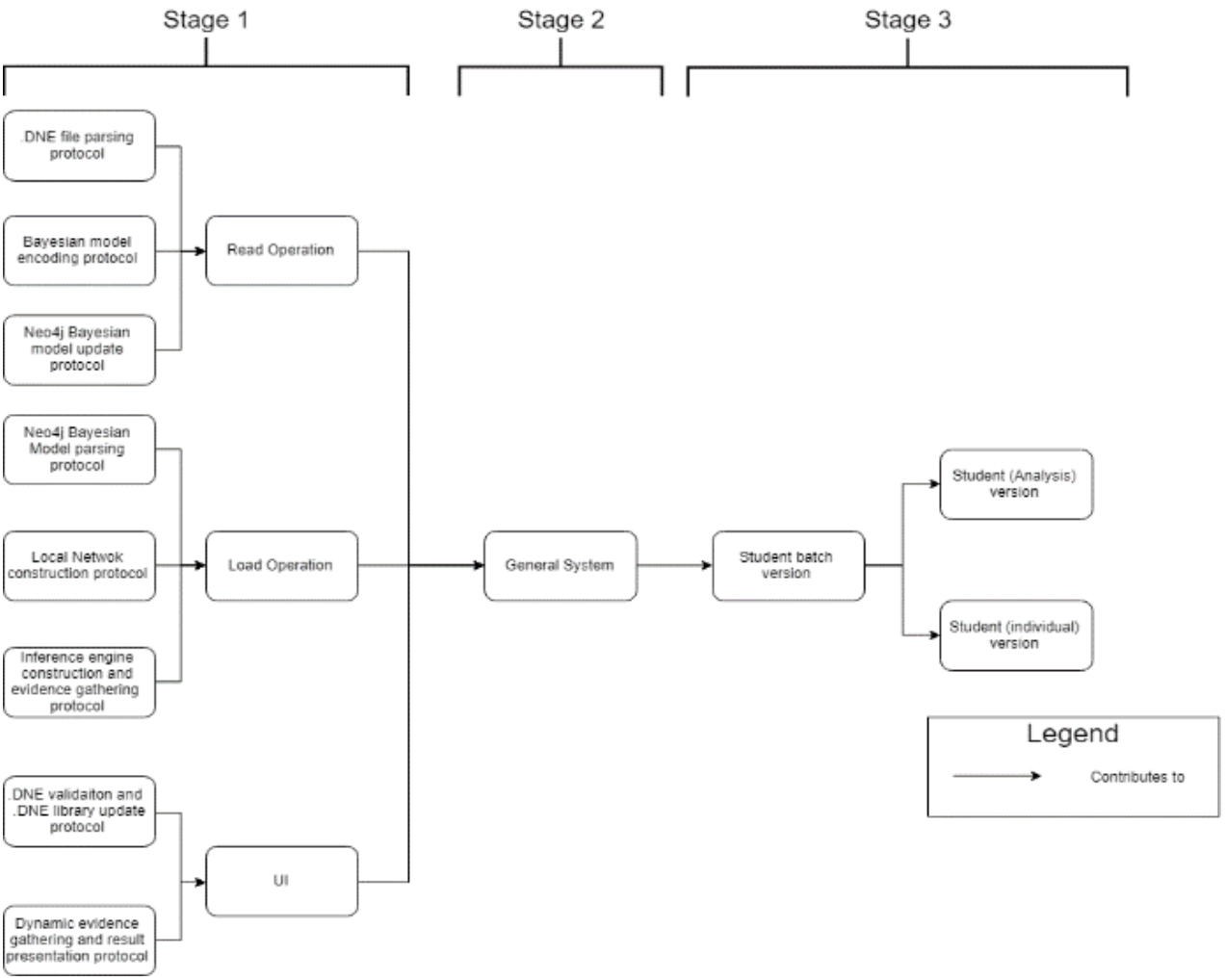

Figure 1. Module construction