Mining Co-Morbidity Patterns - E-Logbook

29002601 – Leah Gourley

# Week 0 – Initial discussion [26.09-02.10.2022]

Results of meeting:

* Pre-operative anaesthetic categorisation project is on the backburner. Focus shift to discovering disease trajectories from a dataset.
* Code basis exists from PHD student in Python – this is a process mining method – research further.
* We want to hopefully extend this to discovering drifts in processes/disease trajectories, such as impacts of Covid on disease progression and mortality – waiting on the data for this.
* We will be using a MIMIC-III database – the code above did not use this, so it’s required that we adapt it for use with MIMIC-III databases.
* Bear in mind this kind of interface would ideally be implemented at the hospital.

Links provided by supervisor:

* <https://github.com/dijahanga/DL_Approach_To_Process_Mining> - Python code
* <https://ieeexplore.ieee.org/document/9203823> - A Graph-Based Approach to Interpreting Recurrent Neural Networks in Process Mining
* [https://www.mdpi.com/1099-4300/24/7/910](https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.mdpi.com%2F1099-4300%2F24%2F7%2F910&data=05%7C01%7Cy.kovalchuk%40reading.ac.uk%7C7cea2e323c314d31db8908da906e772a%7C4ffa3bc4ecfc48c09080f5e43ff90e5f%7C0%7C0%7C637981101370287907%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=XS23z2IY8NHIfcDZupZ1K1%2FOClbqqK3OUc8npVLkgps%3D&reserved=0) - Methods for Drift Detection and Localisation Using Deep Learning Modelling of Business Processes
* <https://physionet.org/content/mimiciii/1.4/> - explanation of the MIMIC-III database
* <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4387452/> - Accessing data from a MIMIC-2 database
* <https://iopscience.iop.org/article/10.1088/1742-6596/971/1/012008> - Process mining in oncology using the MIMIC-III dataset
* <https://www.researchgate.net/publication/348637259_Prediction_of_Treatment_Medicines_with_Dual_Adaptive_Sequential_Networks> - Prediction of Treatment Medicines With Dual Adaptive Sequential Networks
* <https://link.springer.com/article/10.1007%2Fs13042-020-01155-x> – A hybrid method of recurrent neural network and graph neural network for next-period prescription prediction
* <https://www.nature.com/articles/ncomms5022> - Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients
* <https://psb.stanford.edu/psb-online/proceedings/psb18/beaulieu-jones.pdf> - Mapping Patient Trajectories using Longitudinal Extraction and Deep Learning in the MIMIC-III Critical Care Database

Goals for next stage:

* Familiarise myself with the existing Python code.
* Research on disease trajectories to increase understanding.
* Familiarise myself with MIMIC-III databases and how to generate event logs from the data.

# Weeks 1-6 – Background research [03.10-13.11.2022]

Python code:

* From the README in the code repository, the stated tasks this code can complete are:
  + Prediction of the next activity to be executed in a process instance.
  + Prediction of the continuation of a process instance (ie. Its suffix).
  + Generating a process model graph explaining the decision-making of the LSTM model when predicting process event sequences.
  + Computing the similarity between the graphs to validate the generalising ability of the model.
  + Perform some process mining tasks.
* Additionally, the README states: “The tool assumes the input is a complete log of all traces in the CSV format which has a case ID column, and an event column containing activity names or ID. The input log is in some cases split into 70% (training set) and 30% (test set).”
* One of the perceived goals of this project was to produce a new form of deep-learning-based prediction model that improves upon the flaw of existing deep-learning-based models that cannot explain how it reached the decision/outcome by incorporating existing graph-based prediction models.
* I can see how this can be applied to this project – those using the model I aim to produce (i.e. medical professionals) would want to understand why the model reached the conclusion.
* From the paper: “The most widely studied type of process mining is process discovery. Process discovery relies on data collected from an information system over a period of time, or real-time data output by running processes, widely known as an event log. From an event log, process discovery methods can automatically construct a process model appropriately displaying the observed behaviour as it is captured in the log without any inferred information.”
* The model is evaluated using a calculated accuracy percentage (accuracy and log loss were tracked in the training process using the Adam optimisation algorithm).

MIMIC-III databases:

* Multiparameter Intelligent Monitoring in Intensive Care.
* <https://physionet.org/content/mimiciii/1.4/> - it resides here.
* From above: “MIMIC-III is a large, freely-available database comprising deidentified health-related data associated with over forty thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. The database includes information such as demographics, vital sign measurements made at the bedside (~1 data point per hour), laboratory test results, procedures, medications, caregiver notes, imaging reports, and mortality (including post-hospital discharge).”
* This data is freely available under a data use agreement.
* Following HIPAA, the data has been cleansed. Particularly worth noting the dates have been moved forward at a random offset to between 2100-2200, but time of day, day of the week, general seasonality and intervals have been preserved.
* Also, patients aged >89 will appear in the database with an age >300.
* <https://github.com/nsh87/mimic-iii-vm> - here is a repository for creating a VM running PostgreSQL to hold the MIMIC-III database. I am yet to try this.

My next steps:

* Establish a method for generating event logs from MIMIC-III database as appropriate.
* Adapt existing code to achieve
* Generate training event logs from the MIMIC-III database and train the model to predict trajectories.
* Consider which tables/columns in particular from the database are needed.
* Establish a means of evaluating the output of the model.
* Identify other models that attempted to predict disease trajectories, for discussion.

# Week 7 – Discussion with supervisor [14-20.11.2022]

Main goal identified: outline the process to generate event logs from the MIMIC III database to fit into the algorithm.

* Each patient is a row, and the values are comma separated. We want the data to be chronological in order of diagnosis.
* Consider running issues during diagnosis as primary/secondary – does the database do this?
* An interim step is columns: patient; primary; secondary 1; secondary 2; secondary 3; etc.
* It’s important to note that if a patient dies/is discharged, to predict morbidity – add to the chain of diagnosis.
* Maybe record department names, and anything important in the patient’s journey.

Next steps:

* Gain access to the MIMIC III database.
* Find and follow steps to store database on computer for information/log extraction.
* Review data in the database and consider what needs to be done to create event logs.

# Weeks 8-11 – Analysis and Planning [21.11-18.12-2022]

We want to be able to execute PostgreSQL commands on the database, and a VM is ideal, so we want to follow these steps: <https://github.com/nsh87/mimic-iii-vm>

To do this, we need to install VirtualBox (done) and Vagrant with Ansible capabilities. This is preferably done on a Linux/Unix OS, but I have Windows, so we need to follow these steps to get it to work on a Windows OS: <https://www.azavea.com/blog/2014/10/30/running-vagrant-with-ansible-provisioning-on-windows/>

In order to follow this tutorial, we need to install Cygwin: <https://cygwin.com/install.html> and Babun (I think?): <http://babun.github.io>

I may not need to install Babun as it seems to be outdated, although it may complicate following the tutorial.

Next steps:

* Install Cygwin (and Babun?)
* Install Vagrant and Ansible
* Provision the VM using Vagrant in VirtualBox
* Set up the PostgreSQL environment and import the MIMIC-III database

# Weeks 12-14 – Christmas break [19.12.2022-08.01.2023]

No progress made due to being on holiday over Christmas.

# Week 15-16 – Environment setup [09-22.01.2023]

The goal at this stage is to set up the environment to host and manipulate the data. The MIMIC-III database is a set of large CSV files, so I wanted to handle them using PostgreSQL. I wanted to set up a virtual environment following the steps at <https://github.com/nsh87/mimic-iii-vm> and subsequently, <https://www.azavea.com/blog/2014/10/30/running-vagrant-with-ansible-provisioning-on-windows/> but as I am using a Windows machine the extra steps required to set up the Ansible controller aren’t working.

The errors I’m getting relate to provisioning the VM, and somehow Vagrant loses the path to Ansible due to it not being set up identical to how it would be on a Linux machine. I am unable to install the correct software on the Polly Vacher PCs without IT support, still waiting for a response. I am reluctant to download PostgreSQL and host directly on my machine due to the size of the database.

I also tried hosting the data in Amazon Web Services but I had to pay extra in order to manipulate the data in the CSV files.

I have just emailed my supervisor and will await a response while looking into different ways to access the data.

Next steps:

* Await supervisor’s response and continue to research ways I can host PostgreSQL.

# Week 17 – Discussion with supervisor [23-29.01.2023]

Jenya forwarded my problem to Todd Jones, who suggested using Docker or a cloud provider. Also provided the link: <https://www.prisma.io/dataguide/postgresql/5-ways-to-host-postgresql>

I set up Docker and after following this tutorial (<https://hevodata.com/learn/docker-postgresql/>) I can now handle the database in a virtual environment.

Next steps:

* Extract event logs from the MIMIC-III database. The ideal format would be a column for patient ID, a column containing a string of diagnoses separated by commas, and a column indicating whether the patient was diseased or discharged.

# Week 18 – Extracting Event Logs, version 1 [30.01-05.02-2023]

I decided to load in only the relevant CSV files (patients, diagnoses\_icd, admissions and d\_diagnoses\_icd). It’s easy using SQL commands to drop the irrelevant columns. I can also manipulate the data so I can string all diagnoses for one patient ID into one row by adding a new column for easy primary diagnosis and another for secondary diagnosis.

Next steps:

* Evaluate this approach to extracting event logs.

# Week 19 – Evaluation [06-12.02.2023]

The problem with the current method of using SQL to format the database is the output has a great number of columns, instead of one for all primary diagnoses and another for all secondary diagnoses. Also, it’s difficult to check for missing values and explore our data in SQL in comparison to Python.

Considering I will be handling this data in Python, I have decided I should also clean the data in Python. At the beginning of this project in September, I didn’t know about pandas in Python, but I have since learned how to manipulate data in the form of DataFrames. I will attempt this approach instead.

Next steps:

* Attempt extracting event logs in a Python environment with pandas.
* Evaluate whether this method produces logs in the ideal format, in comparison with the SQL results.

# Week 20 – Extracting Event Logs, version 2 [13-19.02.2023]

Repeating the steps I took in the first version of extracting event logs in SQL, now in Python, is simple using pandas. The data is also easy to get into the desired format – all primary diagnoses in one column, comma-separated. It is also easy to append the event flag on the end of the string, indicating deceased/discharged.

I am not yet sure how to handle the secondary diagnoses in consideration of the program I will be passing these event logs to… For now, I am saving only the primary diagnoses to the CSV file output and I will discuss with my supervisor how to handle secondary diagnoses.

Next steps:

* Input the event logs into the provided code and produce a machine learning model trained on the logs.

# Weeks 21-22 – Implementation [20.02-05.03.2023]

I altered the ‘NextEventPrediction – model1(unidirectional).py’ file from <https://github.com/dijahanga/DL_Approach_To_Process_Mining/tree/master/Code> in order to handle the event logs. I didn’t need the section that created a string of comma-separated values, for instance, as the logs were already in this format.

The program took 6 days to complete fitting the model with the logs…. I’m not sure I want to refit the model if the accuracy isn’t good. I know I will need to when I introduced secondary diagnoses….

Next steps:

* Load the saved model into a new file and evaluate it using some machine learning evaluation metrics. I could use loss and accuracy metrics like in the original code.
* Research other ways I could evaluate the model.
* Consider how to incorporate secondary diagnoses to the model.
* Consider how to reduce training time.

# Week 23 –Evaluation and discussion with supervisor [06-12.03.2023]

My main issue with the current state of this project is how long the model takes to train. It’s likely because of the size of my data. In order to combat this, I should reduce its dimensionality.

I contacted my supervisor to update where I was at. I asked previously how I could include secondary logs. Previously, she suggested we move forward with primary diagnoses and we will both consider a future approach. After consideration, my supervisor concluded that this approach isn’t appropriate as even with repetitive admissions, patients get the same diagnosis. She offered a few options on how we could proceed:

* Changing the task from disease trajectories to disease pathways and producing event sequences such as in <https://iopscience.iop.org/article/10.1088/1742-6596/971/1/012008/pdf>
* Changing the task to clustering analysis to identify common cases of comorbidities from both primary and secondary diagnoses. Following on from this, I could calculate the frequency of co-occurrence for each appearing pair of diagnoses; or focusing on primary only, see how often each combination appears in the database, in any order.

I decided to venture into clustering analysis, as this is an area that I feel more confident in following modules I have had this past term. I will also calculate the frequency of co-occurrence.

Next steps:

* Research approaches to this new problem.
* Consider the best clustering algorithm to achieve the result.

# Week 24 – Research [13-19.03.2023]

Some papers I found that achieve similar results:

<https://ieeexplore.ieee.org/abstract/document/7727984?casa_token=ZwAFNnHzhQMAAAAA:t0uBQPi-JygwDSk_llDro4dIble2Pfroccm2c-u-DFjM6jvoDIP0dy9Y56b_K6GjKIrdWbwJPQ> - this explains well its process of clustering and the decisions made, although too advanced for the given timeframe I have left.

<https://medinform.jmir.org/2022/5/e35422> - much more similar to this project, with ICD10 codes but leans more toward a comorbidity network than disease trajectory prediction.

<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0032141> - again, in the same vein. In this, they use additional features like age and sex.

<https://onlinelibrary.wiley.com/doi/full/10.1002/sim.5426?casa_token=nhBUVAP3Dy0AAAAA%3AWiEtBOPHRPv2NYHYOSyOo9wFQduyOlJInmwif3H0TozGNiWxu9xOpT_EeukdY0XfEAuD5O3XEWOqZYo> - detailed explanations on measurements of comorbidity.

Other notable papers:

I have seen K-means mentioned a few times, and this is one I am confident in working with. I should also consider reducing the dimensionality of the dataset, considering its performance in the last implementation….

Next steps:

* Meet with supervisor and discuss my findings and my ideas for an approach.
* Begin an implementation of clustering analysis.
* Consider what form the input will need to be in in order to achieve the desired end result.

# Week 25 – Discussion with supervisor [20-26.03.2023]

Presenting my plan to my supervisor, we have concluded that the best approach to the next implementation is as follows:

* Compare the k-means algorithm, a hierarchical algorithm such as agglomerative, and the M-algorithm implemented in
* Colour-code the instances by discharged/died to determine whether certain disease combinations and/or more complex profiles (people with lots of secondary diagnoses) increase chances to die. (I may also represent primary/secondary diagnoses with different shapes).

My new project title becomes: ‘**Mining co-morbidity patterns and associations with health outcomes from an intensive care unit registry’**

This should be a simple implementation. I need to re-write my literature review and intro as I am no longer focusing on disease trajectories.

I will need to re-process the database to have binary primary-secondary diagnosis pairs for each row. I need to retain the patient and admission IDs and the event flag, too.

Next steps:

* Begin to build up my report again.
* Research and implement the three algorithms.
* Re-process the database to get the input in an ideal format.
* Evaluate and compare the algorithms.

# Weeks 26-27 – Implementation [27.03-09.04.2023]

Reworked the event log extraction algorithm to have a resulting DataFrame in the format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SUBJECT\_ID | EVENT\_FLAG | HADM\_ID | PRIMARY\_DIAGNOSIS | SECONDARY\_DIAGNOSIS |

Achieved by splitting the list of secondary diagnoses into a new row for each individual diagnosis, the other four columns containing the same information.

Can then use this to perform clustering analysis on primary-secondary diagnosis pairs. Provided all codes are numeric, encoding is not needed. PCA is needed to reduce dimensionality of data.

Perform clustering analysis using scikit-learn’s k-Means algorithm, and also an implementation of the M-algorithm (and subsequently, K-algorithm) such as in <https://link.springer.com/article/10.1007/s10115-021-01623-y> on a list of values of k (number of clusters).

Plotted the results, coloured by cluster. Additionally represent those deceased with a ‘o’ (0) and those discharged with a ‘x’ (1) (or is it the other way round? Must review meaning of event flag column). Tried using silhouette score to evaluate but run out of memory due to large set size. Can use calinski-harabanz score and SSE, but need to consider whether these are effective evaluation metrics or if there is something better.

Hierarchical algorithm not yet implemented – if there is time after the report is mostly finished and the code works, I can implement it. Otherwise, it is not as important (only ideal), as I will still have two algorithms to compare.

Next steps:

* Refine the scatter plots – additionally review the event flag column’s meaning.
* Decide on appropriate evaluation metrics – use of SSE/C-H/relative risk/some other cost function or metric?
* Produce tables of results, perhaps plot any numeric scores on a line plot for visualisation?
* Evaluate, comparing the k-means and m-algorithm results, the appropriate number of clusters.
* Begin to compose the discussion of results for the report.

# Week 28 – Evaluation [10-16.04.2023]

# Week 29 – Presentation Preparing [17-23.04.2023]

# Week 30 – Final reviews [24-30.04.2023]