Learning Latent Space Representations to Predict Patient Outcomes: Model Development and Validation evaluation

Michael Potts and Hannah Fowler
mjpotts2/hfowler42 @illinois.edu

Presentation link: https://youtu.be/olfaCbhQItE
Code link: https://github.com/michael-j-potts/DLH

1 Introduction

The main purpose of the paper we are reproducing is to create prediction models to predict patient mortality by modeling complex relations within EHR data. The models will perform the main contribution of the paper, to use relations between clinical data such as International Classification of Diseases codes, medications, and laboratory components to learn latent features. Those latent features will be used to predict patient mortality and ablations will be used to identify which features are most important in the predictions. The paper achieves this through the use of a correlational neural network (modified auto encoder) which takes all three inputs instead of the standard one (Typically ICD codes).

2 Scope of reproducibility

We are testing the following claims from the paper:

- Claim 1: A LSTM predictive model with a correlational neural network integrated to create relationships between clinical data will have a higher accuracy than a RNN model.
- Claim 2: A LSTM predictive model with a correlational neural network integrated to create relationships between clinical data will have a higher accuracy than a logistic regression model.
- Claim 3: A correlational LSTM predictive model is superior to commonly used sequence data models in representing different types of EHR data as encounter vectors.

3 Methodology

3.1 Model descriptions

The paper uses several different models and ablations, they developed the following models:

- Logistic regression
- RETAIN neural network
- TaRE-TAIN (first) - only ICD codes
- TaRETAIN (previous) - only ICD codes
- LSTM - only ICD codes
- CLOUT - auto encoder only
- CLOUT - Latent space only
- CLOUT - Simple concatenation
- CLOUT - auto encoder concatenation
- CLOUT - latent space concatenation

*TaRETAIN = time-aware RETAIN model. CLOUT is the only model described in detail within our paper, and will be our focus for the sake of the draft. CLOUT intakes ICD codes, Medications, and Laboratory results which are first translated into a latent space layer and an embedding layer simultaneously. From there, we apply ReLU to the embedding layer and then concatenate both the latent space layer and ReLU'd embedding layer. This creates our encounter vector. From there, we separate our encounter vector into individual encounters, which each run through an LSTM. Each LSTM result is applied to an attention weighted sum as well as an attention value (which is applied to the attention weighted sum as well. Finally, the result is passed through a hidden layer and sigmoid function. We then determine whether or not the patient survived. Each of the described models in the paper has been implemented and tested for function and optimization.

3.2 Data descriptions

We are using the MIMIC-III data set that we obtained access to from PhysioNet by completing a CITI training course on working with healthcare data for research. Further, we have implemented

our own data cleaning and preprocessing to prepare our data for our model. This included significant work in understanding the given MIMIC-III data set, as well understanding what is required by our model. Currently we are using the PATIENTS, DIAGNOSES ICD, PRESCRIPTIONS, LABEVENTS, ADMISSIONS, and ICUSTAYS data sets from MIMIC III in our implementation. 7535 patients were retained as these patients have had two or more ICU admissions during the given time frame. Our data was also split into training, validation and testing sets according to the authors original specifications of 70% (5275), 10% (753), and 20% (1509) respectively.

3.3 Hyperparameters

The baseline hyperparameters used by the authors were a learning rate of 0.01, a batch size of 50, and a dropout rate of 0.05 for each of the RNN models. Experimenting with different learning rates, we found a learning rate of 0.03 had comparatively worse results to the baseline: Each model dropped by 0.02% to 0.6%. Further, a learning rate of 0.007 had some improvements in the non-latent space models, but slightly worse performance in the latent space models: an improvement of 0.4% for the rnn concat model, while seeing either no change or a worsening of up to 1.5% found for the latent AE model. For this reason, we have chosen to continue with the baseline rate of 0.01

3.4 Implementation

The authors of the paper we are reproducing did not provide their code but we were able to find the repository of their CLOUT model on GitHub (Subendhu, 2018). We reused the base models provided by the author, and the author provided processmimic.py files, but implemented our own converter file, and changed a significant amount of the original code to correct depreciated/broken dependencies and old code implementations no longer used. We successfully reproduced the authors results while also adding in our own ablations to the models. Looking now at the models, we have a logistic regression model, two RNN models (ICD only, and concatenation of ICDs, meds, and labs), two auto encoder models (one latent space only, and one with latent space and concatenation), and two correlational auto encoder models (one latent space only, and one with latent space and concatenation).

3.5 Computational requirements

For this project, we are using a computer with a i7-7700k CPU, GTX 1070 GPU, 16gbs of RAM, and a 500gb solid state drive. Each model took between 1 minute and 7.5 minuets to run with the average run time being 4 minutes and 30 seconds. Each model also used a different number of runs and epochs within those runs. *The authors originally used only 1 epoch per run for AE cl and CAE cl. Increasing the number of epochs to 5, resulted in the same AUC value, but increased training time from roughly 4 minutes and 30 seconds to 21 minutes and 30 seconds. Next, the baseline models made use of size_average = False which sums the results in a minibatch, while reduction = 'mean' averages the results. size_average is now depreciated and is equivalent to reduction = 'sum' when set to false.

Model	total time	runs	epochs
Logistic Reg.	1 min. 3 sec.	10	1
RNN ICD	3 min. 1 sec.	10	5
RNN c	7 min. 21 sec.	10	5
AE l	5 min. 29 sec.	10	5
CAE l	5 min 15 sec.	10	5
AE cl	4 min. 26 sec.	10	1
CAE cl	4 min. 37 sec.	10	1

* c = concatenate, l = latent

4 Results

Our results were similar in scope to the authors results with a few exceptions: Upon updating our code to python3, we did see worsening results for our latent only models, but a slight improvement in the concatenated RNN model. Experimenting with the learning rate by increasing or decreasing it slightly from its 0.01 baseline also resulted in poorer outcomes. Adding additional epochs to the concatenated, latent models resulted in no change surprisingly. Experimenting with using different reduction statements within the loss function resulted in better results compared to the baseline. Each model (all except for logistic regression) that used reduction = 'mean' found between a -0.1% to 3% increase in performance. The models that benefited from this change the most were specifically the latent models, which further provide evidence that latent correlational networks do appear to have better performance than RNNs. The results given below include this alteration.

Model	AUC 1	std	AUC 2	std
Log. Reg.	81.4%	0	-	-
RNN ICD	83.0%	0.23%	82.9%	0.32%
RNN c	87.8%	0.29%	87.8%	0.27%
AE l	78.5%	0.23%	79.1%	0.40%
CAE l	78.3%	0.34%	81.3%	0.31%
AE cl	87.8%	0.20%	88.1%	0.18%
CAE cl	87.2%	0.33%	88.2%	0.32%

*c = concatenate, l = latent, 1 = sum, 2 = mean

4.1 Result 1

Evaluating the papers first claim, our results were unable to prove this claim with the baseline reduction = sum but were able to more significantly prove it for reduction = mean. By making many slight alterations to the authors code, we found that converting to python3 worsened the performance of the latent models. However, changing the reduction statement results in results more in line with the authors original claims that, yes, latent correlational networks do appear to be more accurate then RETAIN RNNs.

4.2 Result 2

Evaluating the papers second claim, it is clear that the correlational LSTM (CLOUT) model is superior to logistic regression for modeling complex relationships found in EHR data. With an improvement in predictive capacity of 6.8% (CAE cl AUC 2) and 6.7% (AE cl AUC 2), we can conclude that latent space modeling using LSTMs is clearly more effective than logistic regression.

4.3 Result 3

Overall, it can be concluded that, yes, the correlational LSTM model is superior to the other types of models explored in this paper. Although concatenated only LSTMs do appear to perform quite well, it is clear that the reduction = mean alteration had a strong enough effect on the CLOUT model.

4.4 Additional Ablations - hospital mortality

Only changing the mortality flags, we decided to evaluate the difference in results between the authors original implementation (patients who died during the available mimic time period), by evaluating patients who died in hospital. To implement this, we needed to create new code to gather patient deaths found under the Admissions file hospexpflag, then alter each model to accept an input argument to choose to use the given hospital labels in place of the regular labels. Further we also applied reduction = mean to this model and did fine slight improvements to each of the latent models in particular.

Model	AUC 1	std	AUC 2	std
Log. Reg.	0.8531	0	-	-
RNN ICD	86.0%	0.34%	86.4%	0.48%
RNN c	94.9%	0.32%	94.9%	0.15%
AE l	86.2%	0.41%	86.5%	0.43%
CAE l	87.5%	0.21%	89.8%	0.33%
AE cl	94.4%	0.16%	94.4%	0.22%
CAE cl	94.2%	0.36%	94.3%	0.23%

*c = concatenate, l = latent, 1 = sum, 2 = mean

5 Discussion

Overall, we did find this paper to be reproducible, but there were unexpected changes to our results when updating to python3. We did find an overall worsening of performance when switching to python3 by a small amount (0.1% - 0.5%) with only one model imrpoving (RNN concatenate). These changes were not significant, but important to comment on regardless. Taking a look at the original implementation, we were able to duplicate the authors results with some minor variation. This variation is in part due to the update to python3, but also due to the natural variation of rerunning models. Looking next at our hospital only mortality patients, we can see a great improvement in accuracy across all the given models. The average improvement across all models was a staggering 6.36%. This result should also be as reliable as the baseline implementations as none of the training, or core model is changing. Only the models ability to predict if a patient will die in the hospital after being in the ICU vs the baseline models prediction of dying anytime within the MIMIC timeframe after being in the ICU.

5.1 What was easy

The easy parts of this implementation were gathering the MIMIC dataset, communicating with the authors and running the authors models after having completed the preprocessing and conversion. The models did contain depreciated dependencies which were sometimes difficult to properly alter, but for the most part, the models contain a similar base code, so a change to one model, will be similar/the same as a change to another.

5.2 What was difficult

Originally, we implemented our own mimic preprocessing with good results, but slight differences from the authors data set. Some changes we did not expect, were that the authors used whether a patient died at all, while our original implementation focused on if a patient died in hospital. These types of differences originally took a significant amount of time to implement, as we tried to aim for the authors set dataset. The most difficult part of implementing this paper was first trying to preprocess the data to match the authors, and then trying to create a conversion file that properly fit the authors embedding input. The authors code does not have comments, so implementing the preprocessing and the converter file independently was difficult. At times, it felt like trying to solve a puzzle without being able to see what the edges looked like. Further, updating the code to python3 proved to be more challenging than initially expected as it not only broke several more dependencies, but also altered the model results. Another concern would be the time to run the models completely. While not extraordinarily time consuming, a full run (with our ablations) takes about 1 hour or more depending on what computer resources are available (slower speed when multitasking).

5.3 Recommendations for reproducibility

To assist in making this paper more reproducable, commenting the code would be very helpful. Further, we would recommend using numpy or pandas for the data preprocessing as our original preprocessing seemed to take far less time than the authors version of preprocessing. For implementing the models, the authors left in a lot of unused code that was commented out. We would recommend either deleting it, or adding a comment as to why it was commented out.

6 Communication with original authors

We were able to communicate multiple times with the original authors who provided us with one of the preprocessing scripts they used (as our initial preprocessing results differed from theirs). We also asked why they specifically chose to implement the CLOUT model and what the train of though behind it was. Their response was that they "had previously used this multi-view model to encode documents from different languages but which contain the same content." They felt that ""it could work well with healthcare data since the different feature sets tell the same story of the patient's health in different ways." Beyond this, the authors did give us some hints about how to approach implementing the required files for CAE and AE (./CAEEntries.¡ICDs, meds, labs¿). Specifically *"The cae entries files are just the respective entries flattened out. So for icd, you just take all the entries in one visit in one line. Patients with multiple admissions just have multiple lines and you can even use patients with just one admission here since the goal is to train the correlation Al autoencoder. We create such files for medications and labs as well such that each line like contains all features from one visit to use for the autoencoders."*

References

Rongali, Subendhu, Rose, Adam J., McManus, David D., Bajracharya, Adarsha S., Kapoor, Alok, Granillo, Edgard, Yu, Hong (2020). Learning Latent Space Representations to Predict Patient Outcomes: Model Development and Validation. Journal of Medical Internet Research, 22(3):e16374. doi: 10.2196/16374

Subendhu19, (2018, 11). CLOUT. Github. https://github.com/subendhu19/CLOUT