Reproducibility Project for CS598 DL4H : Readmission prediction via deep contextual embedding of clinical concepts

Author
eresin2@illinois.edu

Group ID: 77
Paper ID: 232, Difficulty: easy
Code link: https://github.com/rjenya/DL4H/tree/main

1 Introduction

The goal of the replicated work is to predict hospital readmission risk based on EHR data. Excessive hospital readmissions have negative effect on patient lives and healthcare system. Harmful and expensive readmissions can be avoided using accurate prediction of the readmission risk. Deep neural networks can be effective in learning the relationship between such risk and various risk factors automatically derived from the patient's Electronic Health Records without the need of costly feature engineering performed by experts.

2 Scope of reproducibility

This paper introduces a new deep neural network model CONTENT that predicts hospital readmissions and outperforms other modern methods used for similar tasks. Suggested model architecture has higher accuracy than other widely accepted approaches to perform readmission predictions.

2.1 Addressed claims from the original paper

- Introduced model shows good performance in terms of accuracy and ROC-AUC
- Introduced model outperforms GRU model in terms of accuracy and ROC-AUC
- Introduced model outperforms 'Embedding + GRU' model in terms of accuracy and ROC-AUC
- Introduced model outperforms 'Word2Vec + LR' model in terms of accuracy and ROC-AUC
- Introduced model outperforms RETAIN model in terms of accuracy and ROC-AUC

3 Methodology

I use some of the author's code while implementing missing pieces, mostly baseline models. Author's code is using unknown/missing network layer types. I will attempt to re-implement those layers or the whole network - TBD. This experiment is using relatively small dataset, so Mac CPU should be sufficient to train the model.

3.1 Model descriptions

CONTENT model is inspired by language models capturing semantic via latent topics and syntax via RNNs. Each patient is represented by a collection of visits which in turn are represented by a collection of medical events. This work makes an analogy between a patient and a document which is a collection of paragraphs, each a collection of tokens. CONTENT deep model utilizes both local and global patient context from the EHR data using architecture similar to TopicRNN model (1) - hybrid deep learning model that combines topic modeling and RNN. The global context such as general conditions of the patient are captured by the topic model while local context such as short term disease progression is captured by RNN.

CONTENT model is constructed as a combination of the RNN model (GRU) and a Recognition Network (auto-encoding variational Bayes model). RNN captures complex temporal relationship between medical events while Recognition Network is responsible for producing patient's context encoding - patient specific context vector that represents different subtypes (latent topics) of the clinical situations leading to readmission event (global context). Recognition Network is basically a generative model that models distribution of the context vector θ as an inference network using feed-forward neural network.

Original paper's architecture diagram for refer-

ence:
https://doi.org/10.1371/journal.pone.0195024.g002

Denotations:

- $h = \{h^1, h^2, \ldots, h^N\}$ - set of hidden states for all patients coming out pf RNN when N is number of patients. Each patient's p RNN hidden state is $h_p = \{h_1^p, h_2^p, \ldots, h_{Vp}^p\}$ Vp being number of visits for p. Hidden states are computed using GRU:

  $h_t = GRU(v_{t-1}, h_{t-1}; W_v, W_h)$

- θ - context vector coming out of Recognition Network. It's drawn using learnt multivariate Gaussian distribution.

  Given patient representation matrix p:

  $r_1 = ReLU(W_{r_1} p + b_{r_1})$
  $r_2 = ReLU(W_{r_2} r_1 + b_{r_2})$
  $\mu(p) = W_\mu r_2 + b_\mu$
  $log(\sigma(p)) = W_\sigma r_2 + b_\sigma$
  $q(\theta \mid p) = \mathcal{N}(\mu, \sigma^2)$
  $\theta \sim q(\theta \mid p)$

- $\{y^1, y^2, \ldots, y^N\}$ - hospital readmission indicators. Final readmission prediction is made based on combination of RNN hidden states and context vector using logistic regression. $y_t = \sigma(Q^T h_t + B_t^T \theta)$

Learning objective of the training is to maximize log likelihood $log(p(y \mid h, \Theta))$. Since it's impossible to optimize it directly, loss function for Auto-Encoding Variational Bayes (2) is used. So learning objective becomes optimizing $ELBO$ variational objective function - its maximization leads to minimization of the Kullback–Leibler (KL) divergence between learnt distribution and the target one.
$ELBO = E_{q(\theta)}[log(p(y \mid h, \theta, \Theta))] - D_{KL}(q(\theta) \mid\mid p(\theta \mid y))$

In above representation the expectation term is the negative reconstruction loss, whereas the KL divergence term serves as a regularization that encourages the variational posterior to stay close with the prior. The learning process for the CONTENT model is to estimate the optimal values of models parameters - weight matrices for both RNN and inference models - by using stochastic gradient descent (Adam) with back-propagation.

Number of parameters - TODO (after I implement the model)

3.2 Data descriptions

The original study was conducted using both real life EHR data and synthetic EHR data simulated from de-identified real dataset. Authors don't provide access to the real life dataset (supposedly because of understandable privacy concerns), but synthetic dataset is made publicly available in the article. As result I am using generated data only in my replication study. Dataset consists of visit/diagnosis information for 3000 patients. I will compare my results against relevant sections of the study that correspond to the synthetic data metrics only. Input EHR dataset contains patients' visits with related ICD-9 diagnosis codes and service location. Recurring visits indicating 'INPATIENT HOSPITAL' as service location within 30 days period are considered 'readmission' label.

3.3 Hyperparameters

CONTENT model:

- the size of the RNN hidden layer is 200.
- Adam's learning rate is 0.001.
- batch size for the shuffled mini-batches is 1

Baseline models:

- use word vector size 100 for Word2Vec in baseline model implementation
- other hyperparameters for baseline models implementation are unknown so picked some reasonable choices.
  - Use hidden layer of size 128, batch size 64, and ADAM learning rate 0.001 in vanilla GRU baseline model implementation since it provides best results.
  - Use hidden layer of size 128, batch size 32, embedding size 256, and ADAM learning rate 0.001 for bi-directional GRU with embedding layer
  - Use hidden layer of size 256, batch size 32, embedding size 256 and ADAM learning rate 0.001 for RETAIN model

3.4 Implementation

- Use modified author's code for EHR data preprocessing https://github.com/rjenya/DL4H/blob/main/Preprocess.ipynb

- Mostly reused author's code for model implementation, but run into issues: original code is utilizing outdated layers of the 'lasagne' neural network library (not found in any existing version). Implemented Gaussian Sample Layer - I believe that was author's intention based on published architecture description. https://github.com/rjenya/DL4H/blob/main/CONTENT.ipynb

- Implemented baseline models - not included in the original project's git repository.
  - Vanilla GRU model, uses one-hot encoding for diagnosis list: https://github.com/rjenya/DL4H/blob/main/Baseline_GRU.ipynb
  - GRU model with embedding layer: https://github.com/rjenya/DL4H/blob/main/Baseline_EmbGRU.ipynb
  - RETAIN model - RNN model with attention mechanism: https://github.com/rjenya/DL4H/blob/main/Baseline_RETAIN.ipynb
  - Word2Vec + LR model - encode visits using Word2Vec model and use Logistic Regression for readmission prediction: https://github.com/rjenya/DL4H/blob/main/Baseline_Word2VecLR.ipynb

3.5 Computational requirements

Training the models (proposed CONTENT model as well as baseline models) is not computationally challenging because of relatively small size of the input dataset (46Mb, 3000 patients).

Training vanilla GRU model on the provided dataset takes 204.7 secs of CPU time and 2699.1 MByte of memory.

Each Epoch of the CONTENT model training takes 500-1000 seconds.

4 Results

My work supports the claims:

- Introduced model shows good performance in terms of accuracy and ROC-AUC

- Introduced model outperforms other methods in readmission prediction in terms of accuracy and ROC-AUC
  - Vanilla GRU model with one-hot encoding
  - 'Embedding + GRU' model
  - 'Word2Vec + LR' model
  - 'RETAIN' model

	acc	roc-auc	precision	recall
CONTENT	.834	.790	.718	.423
GRU	.821	.778	.741	.314
GRU+Emb	.637	.505	.236	.276
RETAIN	.75	.62	.40	.17
W2V+LR	.796	.759	.713	.248

4.1 Result of the proposed CONTENT model

Prediction performance of the replicated CONTENT model is not in agreement with paper's results.

Replicated model's prediction performance is reported after 6 epochs.

	acc	roc-auc	precision	recall
Paper	0.717	0.689	-	-
Repl.	0.834	0.790	0.718	0.423

I mostly reused author's code excluding missing lasagne layer that I replaced with my implementation of the Gaussian Sample Layer - based on my understanding of the model's architecture. I don't have an explanation why my results are so much better.

4.2 Result of the baseline model 1- GRU

Prediction performance of the implemented basic GRU model is not in agreement with paper's results.

Replicated model's prediction performance is reported after 6 epochs.

	acc	roc-auc	precision	recall
Paper	0.714	0.688	-	-
Repl.	0.821	0.778	0.741	0.314

There is no information on hyperparameters used by authors for basic GRU model implementation. That can explain difference in results.

4.3 Result of the baseline model 2- GRU with Embedding

Prediction performance of the implemented model is not in agreement with paper's results.

Replicated model's prediction performance is reported after 6 epochs.

	acc	roc-auc	precision	recall
Paper	0.712	0.684	-	-
Repl.	0.637	0.505	0.236	0.276

There is no information on actual implementation of this baseline model. Paper actually mentions 'GRU+Word2Vec' model that I implemented as bi-directional GRU with Embedding layer under assumption that it's a comparable interpretation. There is probably a disconnect between my implementation and author's implementation that causes big discrepancy in end results. This model performs much worse than GRU with one-hot encoding, probably because low-dimensional embedding doesn't capture the information well.

4.4 Result of the baseline model 3- Word2Vec + LR

Prediction performance of the implemented model is not in agreement with paper's results.

Replicated model's prediction performance is reported after 6 epochs.

	acc	roc-auc	precision	recall
Paper	0.623	0.604	-	-
Repl.	0.796	0.759	0.713	0.248

There is no information on actual implementation of this baseline model. I implemented it based on my understanding of what it means. That can explain difference in results.

4.5 Result of the baseline model 4- RETAIN

Prediction performance of the implemented RETAIN model is not in agreement with paper's results.

Replicated model's prediction performance is reported after 6 epochs.

	acc	roc-auc	precision	recall
Paper	0.731	0.692	-	-
Repl.	0.75	0.62	0.40	0.17

There is no information on hyperparameters used by authors in model implementation. That can explain difference in results.

4.6 Additional results not present in the original paper

In my additional experiment I assess the sensitivity of the model to its hyperparameters. Model seems to have steady performance that varies only slightly with different hyperparameters choice.

hidden size	emb size	topics	roc-auc	acc
150	50	50	0.799	0.836
300	200	50	0.801	0.835
100	100	100	0.801	0.836

5 Discussion

The original paper was reproducible in its main claim - proposed model did demonstrate results better than other methods - baseline models, but my results for all the implemented models differed drastically from the published results.

Overall, the evidence I got from running the code supports the claims of the paper. My approach has some shortcomings though:

- I didn't implement Med2Vec based baseline model - didn't find suitable library.
- I had to make some guesses in regards to lasagne layers that were used in author's implementation but are missing in the library.
- I had to make some educated guesses in regards to implementation of the baseline models - not well defined by the paper.

5.1 What was easy

Implementing baseline models was relatively easy because we've covered most of them in class. The only problem is that there seems to be some discrepancy between my implementation and author's - results are pretty different.

5.2 What was difficult

The most difficult part was to get author's code to run. The code was broken because of library mismatch - maybe just different library version (didn't find related layers in older versions either) or author just used privately enhanced library. I had to make some assumption regarding author's intention and implement accordingly. Paper was somewhat vague as source of truth.

5.3 Recommendations for reproducibility

Recommendations to the original authors for improved reproducibility:

- I am afraid there are bugs in the provided code - I couldn't make it work with batch size other than 1, while having batch size 1 doesn't make much sense to me

- Provided code is not well organized and some crucial portions are missing
- Provided code is not well documented
- Baseline models are not well defined

6 Communication with original authors

No communication with original authors - article doesn't expose contact information.

References

References

[1] Dieng AB, Wang C, Gao J, Paisley J. TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency. International Conference On Learning Representations. 2017;.

[2] Kingma DP, Welling M. Auto-Encoding Variational Bayes. CoRR. 2013;abs/1312.6114.