Reproduction of paper 228 for CS598 DLH4H in Spring 2022

Daoxing Wang, Zuliang Weng
{wang264, zwe}@illinois.edu

Group ID: 37, Paper ID: 228
Code link: https://github.com/jeffzlweng/DII-Challenge

1 Introduction

Sepsis is a life-threatening condition and its high cost adds burden to the healthcare system. Studies show that early prediction can significantly improves the survival in sepsis patients. There are a lot of researches on this topic and many models were built with limited success. Traditional models and approaches suffer in two ways: they predict too late and they are hard to interpret. Most of the models were also built in ICU settings rather than the more critical Emergency Department. The authors of this paper built a Long Short-Term Memory (LSTM) based model that not only out-performs the traditional models and, with attention mechanism and global max pooling, they also give the model clinical interpretability.

2 Scope of reproducibility

The authors claimed that their model achieved an average AUC of 0.892, the mean of 0.940 and 0.845 from two use cases. We will be focusing on reproducing the first case (where the AUC is 0.940) because a) authors provided a detailed AUC scores on all the subpopulations. and b) it's the highest score among all the models and cases being tested. If time permits we will also reproduce the ablation study that removes event encoding, time encoding and global max pooling respectively.

2.1 Addressed claims from the original paper

- The model achieves a total AUC of 0.942 in use case 1.
- Proposed model without event embedding produces AUC 0.83 in use case 1.
- Proposed model without time encoding produces AUC of 0.89 in use case 1.
- Proposed model without global max pooling produces AUC of 0.91 in use case 1.

3 Methodology

The authors provided a link to a github repo in the paper. We will fork from that repo and check in any changes that we make to our own repo. This will include any changes that are needed to run the code on our own hardware. Also it's possible that libraries used are out-dated and needed upgrading. Data sets used in buliding the models are also part of the repo so we don't need to acquire that from MMIC website. Both members in the team has a Windows machine and a Mac. None of them has a Nvidia graphic card so Cuda is out of the picture. By checking the size of the dataset we estimate that it won't take significant computing power to train the models.

3.1 Model descriptions

Architecture

- embedding of vocal size and embed size
- Relu with Dropout $p = 0.25$
- Apply max pooling or softmax on visits
- Bidirectional LSTM
- AdaptiveMaxPool1d
- Relu with Dropout $p = 0.25$ with weighted average on master record

Notes: authors claimed in the paper that attention mechanism was used to provide interpretability but we have not found it in the code. Will update this architecture when we have more details.

3.2 Data descriptions

Authors attached a few CSV files in the code base

- vital.csv: vital numbers at different time points for each patient

- label.csv: 100 labels for each patient. 0 is no sepsis and 1 is sepsis
- master.csv: 100 patient info
- feature_index_dict.json: mapping between feature text and numerical value
- feature_value_order_dict.json: normorlized feature values in 0..1 range
- group_index_dict.json and index_group_dict.json: mapping and inverse mapping between feature index to feature group
- index_feature_list.json: a list of available features but it's never used in the code.

3.3 Hyperparameters

In the code, a set of hyperparameters is provided by the authors. We have not found any other meaningful hyperparameter searching logic. The only grid-search we found was a loop through a predefined seeds, which don't make much sense. They just used the training data to train and the validation dataset to test.

- Learning rate: 0.0001
- Embedding vocab size: 2031
- Embed size: 256
- Hidden size: 256
- Number of layers: 2
- Dropout Rate: 0.25

3.4 Implementation

We will be building the models on top of the code base provided by the authors. The link to the git repo is: https://github.com/jeffzlweng/DII-Challenge.

Code changes

- Convert to Python 3 because Pytorch does not support python 2.7 in Windows system.
- Add key checking logic to handle edge cases in data generation pipeline.
- Correct some documentation errors.
- Fine-tune grid-search logic to include more hyperparameters.

3.5 Computational requirements

It takes 10 minutes to train best models for task 1 on a machine with the following setup:

- Processor Intel(R) Core(TM) i78700 CPU @ 3.20GHz, 3192 Mhz, 6 Core(s), 12 Logical Processor(s)
- Installed Physical Memory (RAM) 32.0 GB
- No GPU is used

4 Results

Our focus for drafting phase is to verify the AUC of 0.94 for task 1. Our testing produces similar results compared with authors' claim.

4.1 Result 1

The result in case 1 display an AUC score of 0.9444, accuracy of 0.83, F-1 score of 0.8, recall of 1.0, and precision of 0.667.

4.2 Result 2

The result in case 2 display an AUC score of 0.9421, accuracy of 0.7557, F-1 score of 0.7787, recall of 0.9406, and precision of 0.6643.

4.3 Analysis

For result 1 presented in 4.1, the AUC score is very close to authors' result. One thing particularly interesting is that we have a recall rate equal to 1, which means that we successfully select out all patient that will have sepsis condition. In this particular use case, false positive is more acceptable compare with false negative as the later post a significant risk on patients and could delay patients' treatment.

For result 2 presented in 4.2, the AUC score is even higher than authors' result. This is unlikely so we should investigate further.

4.4 Future Plan

There are a few concerns we have regarding the reproduction of this paper.

- The result for task 2 is better than authors claimed. This is uncommon so we want to confirm that.
- Fine-tune grid-search logic to include more hyperparameters.
- The ablation study for case 1.
- Test the model with more data sets.

- If time permits, use GAN or generative models to generate more data for testing.
- Present our code and findings via a YouTube video.

5 Discussion

TODO

5.1 What was easy

TODO

5.2 What was difficult

TODO

5.3 Recommendations for reproducibility

TODO

6 Communication with original authors

TODO

References