RESEARCH REPORT

Rethinking EHR: Enhancing transformer-based models for reorganized tree-structured Electronic Health Records

Yihan Tang / The University of Hong Kong/ Hong Kong SAR

Supported by the Faculty of Science, The University of Hong Kong

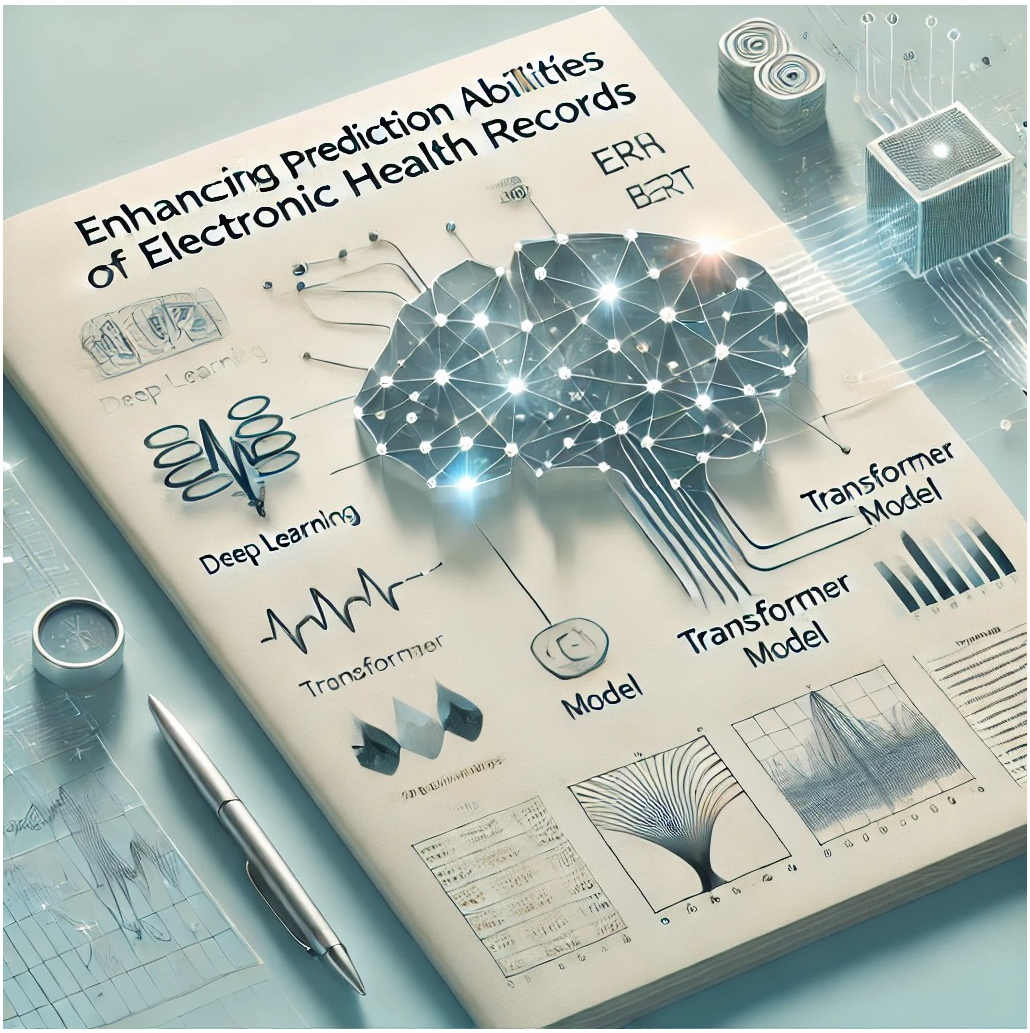
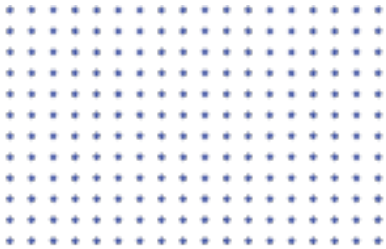


TABLE OF CONTENTS

INTRODUCTION



With developments in Natural Language Processing (NLP), foundation models such as Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2018) have been widely applied in various fields. Particularly in Electronic Health Records (EHR) under AI for Healthcare, several works (Rasmy et al., 2021; Yang et al., 2023) that build upon the aforementioned models have been published. However, most works involving the usage of Transformer and BERT models are evaluated using The International Classification of Diseases (ICD) codes. The primary goal of ICD in healthcare settings is to provide medical staff with fast and standardized access to billing information. However, ICD codes have long been criticized by medical researchers as lacking ontological components (Lin et al., 2023). This significantly harms the interpretability of ICD-based prediction models, making research results based on them less likely to be applied in clinical settings. Therefore, this research project aims to address the problems brought about by the discrete nature of ICD codes through tree-based models. This project enhances transformer-based models so that they offer more interpretability of tree-structured ICD codes.

METHOD

Transformer-based models

Similar to what a BERT model does in terms of text generation (Devlin et al., 2018), the model to be employed in this project treats ICD codes as input tokens. The model makes predictions of future disease in a similar way as language models predict the most likely word that follows before generating that word.

The pretraining objective of the model employed is based on Masked Language Modeling (MLM), as is done in BERT (Devlin et al., 2018). During pretraining, 15% of the tokens are masked in pretraining, in which 80% of the tokens are replaced with [MASK], 10% of them are substituted with a random token, and the rest 10% of them remain unchanged.

Tree-structured ICD codes

ICD codes are normally structured as #icd code version - #icd code. However, merely using ICD codes at that level sacrifices the rich ontological information contained. This project aims to classify ICD codes in tree structure. Specifically, different trees correspond to the illnesses that different hospital departments deal with. For instance, ICD-10 I21.4 (Non-ST elevation myocardial infarction), and I50.9 (Heart Failure) are both related to the cardiology department while I02 **(**Rheumatic chorea) may need to be treated primarily by neurology experts.

CHALLENGES AND SOLUTIONS

Throughout the process, I have encountered many new problems and I always try to address them as soon as possible.

Challenge 1 A Steep Learning Curve in Natural Language Processing

In the course of two and a half months, I had to develop my Natural Language Processing (NLP) skills from a basic understanding to being capable of training and optimizing large language models (LLM). Prior to this project, I have only been trained on relatively basic NLP tasks during my time at Stanford University.

Solution 1.1 Consistent self-learning

I have been reading academic research papers extensively, ranging from Transformer architecture, and pre-training to Recurrent Neural Networks, and parsing. I have also completed coding assignments published by Stanford University [CS224] (Natural Language Processing with Deep Learning). Besides, I frequently read blogs and posts written by experienced Machine Learning engineers and professors. I have gained great improvements through consistent and active self-learning.

Solution 1.2 Read, test, and write more code

I believe that the best way to enhance coding skills is to spend more time practicing. I have been reading, running, and testing codes for months. The most recent [codespace] I have worked on consists of around 5,000 lines of code. I have read the entire codespace, conducted segment tests, and made modifications as well as comments (to be covered later) to the codespace. I have created a new codespace with my modifications to the original one (please tell me if you need to look at the code).

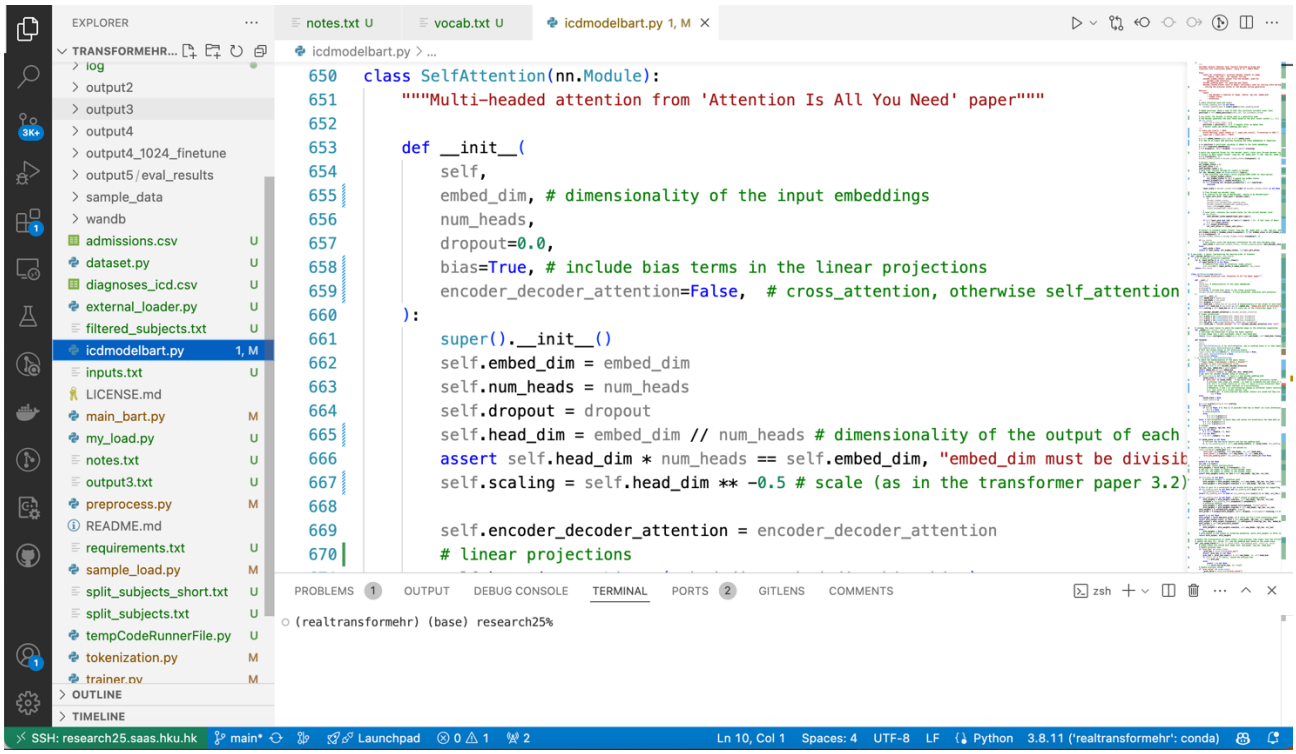


Figure 1. An incomplete and high-level view of the codespace

Challenge 2 Data availability and replicability

At the very beginning of my project, I was troubled by the fact that due to [security and privacy protocols], the [code] is only partially open-sourced. I could not access the majority of data used for pretraining, nor could I obtain the code for pre-processing data. With a significant part of the codespace unavailable, I could not have any results generated. Therefore, it was not immediately clear how I could replicate the code of the original paper.

Solution 2 Ask and innovate

The very first thing I did was to reach out to the author of the codespace, Dr. Yang. Within the security protocols, he kindly provided some .pt files (embedded tokens in PyTorch format) to me. Nevertheless, he declined to offer any further explanation or certain parts of the code that were still missing.

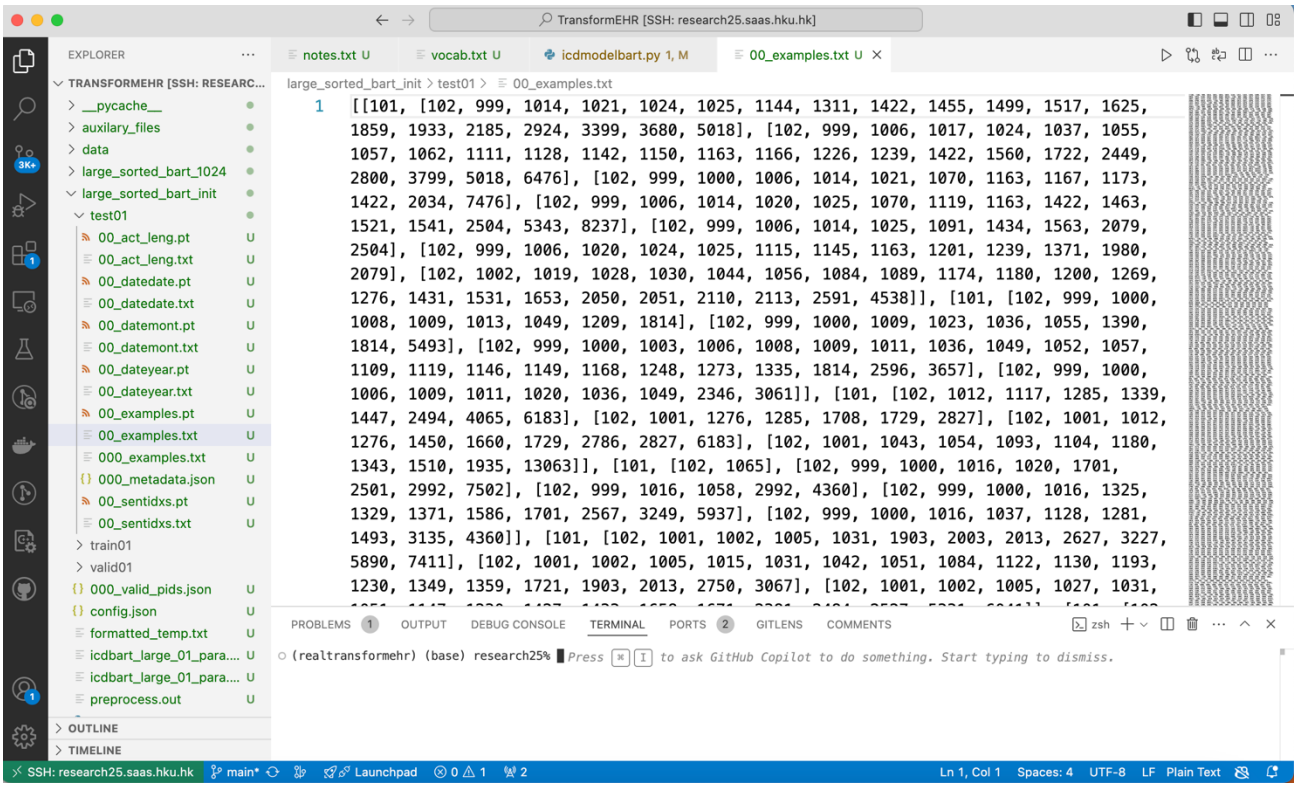


Figure 2. Files that lack any related explanation, awaiting reverse-engineering

Therefore, I had to complete the codespace, including almost the entire data pre-processing pipeline and some modifications to the underlying Transformer architecture. It turned out to be time-consuming as I needed to reverse-engineer the exact data format and deal with a lot of incompatibilities. First, I needed to read through the .pt files, matching them with details of implementations in the [original paper]. Then, I needed to figure out the exact preprocessing steps. Even after completing those two steps, there remained the issue of incompatibilities. It turned out that my data inputs were longer than the original ones, causing dimensionality problems.

I consulted PyTorch forums, PhD candidates, and even GPT models to solve the problem. Through this process, I believe that my debugging skills and my understanding of the language models' details have been greatly enhanced.

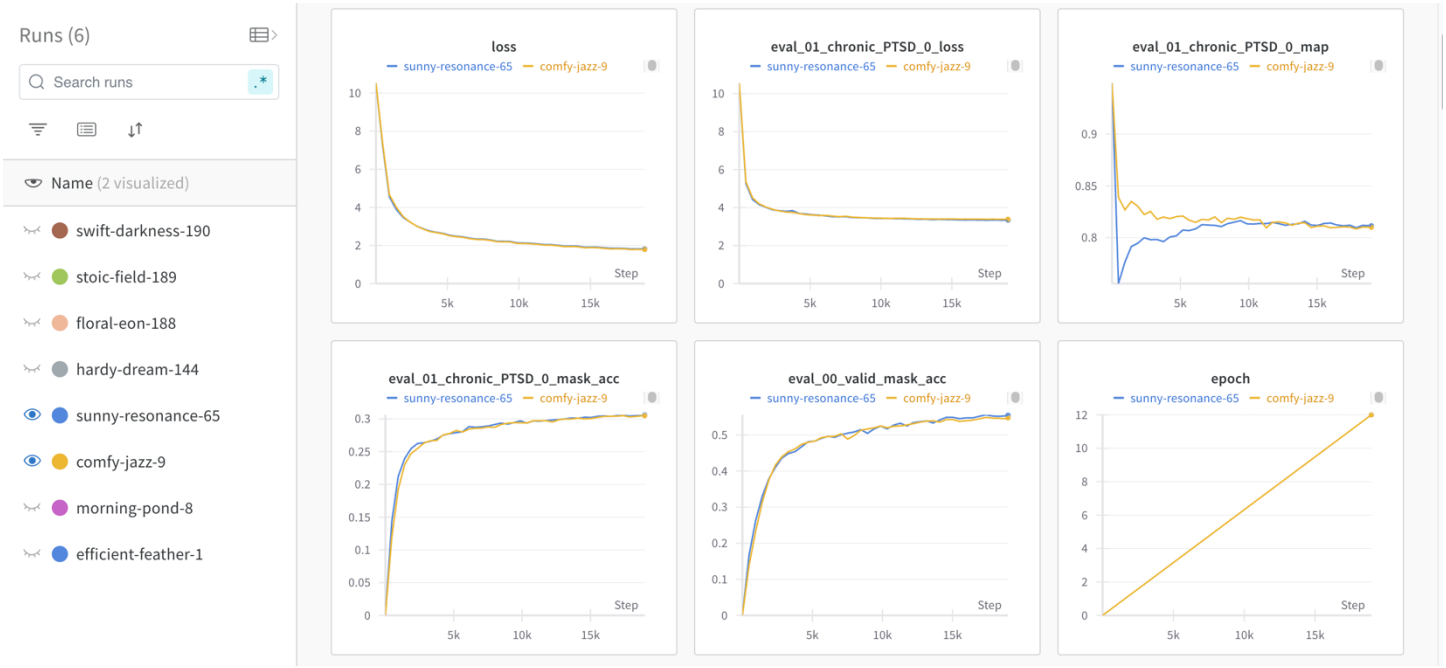


Figure 3. A brief overview of successful pre-training

Challenge 3 Lack of prior experience in leading a research project

Previously, I have never led an academic research project (though I have led several projects at the University of Hong Kong). Consequently, almost every problem I have faced up until now is brand new to me. Sometimes I felt that I was at a loss how to address the difficulties. Furthermore, I found it hard to evaluate my performance.

Solution 3 Ask questions; Ask for feedback

All the way during my research experience (and in fact, all the way through my past 8 years), I have been asking people questions. I have been fortunate to be advised by a kind professor (Professor Lequan Yu) and a helpful PhD candidate (Mr. Fuying Wang) in MedAI Lab at the University of Hong Kong. I have also benefited a lot from talking to other PhD candidates in my lab, outside my lab, as well as various contributors in different forums. It is through my conversation with them that I directly or indirectly find new ways of dealing with difficulties.

I also actively ask for feedback from others. Although I am not required to report to anyone in my lab, I constantly write a weekly report and show it to my supervisors. From their feedback, I get to know potential areas of improvement and share joy of progress with them.