Paper ID: ClinicalBERT: Using a Deep Learning Transformer Model to Predict Hospital Readmission. (**Difficulty Level**: undefined)
Youtube Link: https://youtu.be/Xn9gfh4eQ7Q
Github Repo: https://github.com/wdchild/CAMLBERT
BERT Model: https://drive.google.com/drive/folders/1X_oOiKWE5WRebNDAyniafuycQZYkQYXu

Introduction

This project is modeled after an online article by Nwamaka Imasogie entitled "ClinicalBERT: Using a Deep Learning Transformer Model to Predict Hospital Readmission." This article is, in turn, based on a similar study by Kexin Huang, Jaan Altosaar, Rajesh Ranganath entitled "ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission." Though addressing the same issue (hospital readmission), they differ as described below.

Huang et al.'s article utilized a generic version of BERT and trained it using MIMIC-III so as to produce "ClinicalBERT"—a version of BERT performs better when analyzing medical discharge notes. In this way they were able to achieve significant improvements in a variety of measures relative to standard BERT. Their prediction of hospital readmission based on discharge notes was significantly higher than previously established baselines.

Following up on this study, Nwamaka Imasogie took a slightly different approach, using a rather ingenious insight that predicting readmission based on discharge notes was of little practical value given that healthcare practitioners would have no way to provide meaningful interventions to prevent readmission once the patient had left the hospital. Therefore, she utilized the ClinicalBERT that had been pretrained by Huang et al., and added another neural layer, while focusing primarily on second-day and third-day notes *prior to discharge* to see if it would be possible to predict readmission *before* discharge. In this way, doctors and other care givers would still have time to provide meaningful intervention *before* patients left the hospital, so as to prevent readmission.

Scope of reproducibility

Despite numerous attempts to redress the problem and over a month of waiting, I never attained access to MIMIC data. For the project draft, I used MIMIC demo ADMISSIONS data, and was able to test my preprocessing code and run ClinicalBERT on substitute data cobbled together by combining CAML text with the values provided in ADMISSIONS.

For the final project, at the suggestion of TA Zhengbang Wu, I substituted CAML data, which consist of sequences of text and a series diagnostic labels. I processed this data to produce binary labels (abnormal / normal) and used the resulting data for training, validation, and testing. My code largely follows the code base Imasogie provides, though I was forced to make numerous modifications to account for the different data content.

There was no way to attempt meaningful reproducibility given the radically different datasets, and the fact that Imasogie was predicting readmission within 30 days based on 3 million 2- and 3-day clinical notes, whereas I was predicting abnormal vs normal lung

condition based on 400 records CAML data. So instead of trying to reproduce her results, I was forced to test the applicability of (an adjusted version of) her model to a new use case.

Methodology

• *Model description*

The original ClinicalBERT model (Huang et al.) was developed by utilizing BERT and training it on MIMIC-III EHR data and discharge notes using two unsupervised language modeling tasks: masked language modeling (where some portion of input tokens are held out for prediction) and next sentence prediction (where the model tries to predict whether two sentences are consecutive). The input tokens were based on subword units to address the out-of-vocabulary issue expected with EHRs, and the tokens comprised a combination of that token's embedding, a segment embedding that identified the sequence the token was associated with, and a position embedding corresponding to the token's position in the input sequence. A classification token was inserted in front of every input token sequence.

Now, given that BERT and its derivative, ClinicalBERT, are transformers, as one would expect, they contain an attention function. This attention function is applied to the input sequence comprising the input token embeddings, the input comprising a set of queries, keys, and values. The queries, keys, and values are constructed by multiplying learned weights to the input embeddings. The attention function is computed as follows:

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}((\boldsymbol{Q}\boldsymbol{K}^T / \sqrt{\text{d}}) \text{V})$$

where Q is the query, K is the key, V is the value, and d is the dimensionality of the queries, keys, and values.

Imasogie's adaptation of ClinicalBERT applies a final network for fine-tuning Huang et al.'s pretrained ClinicalBERT on 3-day notes and testing on 2-day and 3-day notes. In her approach, she trained BertForSequenceClassification (with bert-base-uncased tokenizer) for only one epoch. For the CAML data, I took essentially the same approach, but experimented with the use of different epochs of training. I also used random shuffling for each epoch.

• *Data description*

With MIMIC-III unavailable to me,[1] I had to make a late dataset substitution. I used the data from the "CAML" HW assignment. This data originated from the Indiana chest X-ray dataset, and consisted of three fields: a report ID, a Text description of the condition of the lung, and Label, consisting of a multi-hot vector representing the following diseases: normal, cardiomegaly, scoliosis / degenerative, fractures bone, pleural effusion, thickening,

[1] This project was supposed to use Medical Information Mart for Intensive Care III (MIMIC-III), a dataset hosted for free on PhysioNet. MIMIC-III has nearly 30 tables containing a wide range of information on patients and their demographic information, diagnoses, lab events, and the like. I describe this dataset in more detail in the project draft, but it is irrelevant for the project final, and will not be described here.

pneumothorax, hernia hiatal, calcinosis, emphysema / pulmonary emphysema, pneumonia / infiltrate / consolidation ,pulmonary edema, pulmonary atelectasis, cicatrix, opacity, nodule / mass, airspace disease, hypoinflation / hyperdistention, catheters indwelling / surgical instruments / tube inserted /, medical device, and other. To more closely mimic the binary nature of Imosagie's data, I utilized only the first flag, representing "normal" (1) and "abnormal" (0), eliminating the other flags during the preprocessing stage. In preprocessing, the text data was normalized by eliminating capitalization and punctuation and removing stopwords.

• *Implementation:* **https://github.com/wdchild/CAMLBERT**

Implementation of the code involved a preprocessing step, a training step, a validation step, and a test step. I split the train, validation, and test sets using an 80:10:10 split. The data for the splits was selected in randomly to avoid any possible bias originating from the original ordering of data.

• *Ablation (Comparative Techniques)*

For comparative classical NLP modeling techniques, I tried using both Scikit Learn's CountVectorizer and TfidfVectorizer were both utilized. Imosagie states that applied the same NLP preprocessing for both classical NLP ML and BERT-based ML, and so I applied the same essential preprocessing. Even though stopwords and other noise were removed, the original word ordering was otherwise preserved.

• *Parameters*

Because ClinicalBERT is pretrained, model tuning is much faster, and so for most runs, I simply ran one epoch of training. The initial hyperparameters used for the ClinicalBERTmodel were as follows (see Results for variations in the parameter settings).

Hyperparameter	Value
attention_probs_dropout_prob	0.1
hidden_act	gelu
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
max_position_embeddings	512
num_attention_heads	12
num_hidden_layers	12
type_vocab_size	2

Hyperparameter	Value
vocab_size	30522
train_batch_size	32

For the loss function, to train the original ClinicalBERT, Huang et al. used Adam. In the article I was following, however, AdamW (a variant of Adam that uses a different learning rate for each parameter) was used. The loss function for ClinicalBERT was BCELoss, with a sigmoid function used for the final classification. 5-fold cross-validation was employed. Only one epoch was employed initially, but I experimented with the number of epochs to see if that made any difference.

Evaluation was performed using AUROC, AUPRC, and RP80 (see Results below). In Imasogie's article, probabilities were computed as follows:

$$P(\text{readmission} = 1 \mid h_{patient}) = (P_{nmax} + P_{nmean} * n/c) / (1 + n/c), \text{ where}$$

$h_{patient}$ is the implicit representation computed by clinical BERT based on the patient's notes
P_{nmax} = maximum probability predicted by a subsequences
P_{nmean} = mean probability across all subsequences

However, after much deliberation on why I was getting such bad results, I realized that this was because these P values were totally inapplicable to the CAML dataset. I retooled the vote-score function and as a result the values were much improved (see Results below). In the retooled version, the prediction was the direct output of the value produced by the softmax (equivalent to sigmoid since there are only two classes). Even with these changes, I was not getting the values I expected. After many days of no progress, I realized that the polarity of the predictions was actually reversed (see results below).

- *Computational requirements*

Although I had hoped to install PyTorch and Transformers on an Apple MacBook Air with M1 chip (the procedure for this workaround is described online), the Transformers install failed. Therefore, I initially ran a Jupyter notebook file on a 2012 MacBook Pro with 2.7 GHz i7 processor and with upgraded 2TB SSD HD and 16GB of memory. Epoch training on this machine was glacial—an initial run with one epoch of training took 2 hours, and a subsequent run with 8 epochs took 15 hours.

Because this computer was so slow, I ran subsequent runs on Google Colab. At first I used the free version of Colab, but this too was slow, so I eventually opted for the "pro level" utilizing 27.3 GB of available RAM (the exact amount differed depending on the run) using a GPU (NVIDIA-SMI 460.32.03 Driver Version: 460.32.03 CUDA Version: 11.2).

With so little data (only 4000 records total, and < 1000 in the test data), runtimes were far far faster on Colab Pro. Sample runtimes are given below.

Hardware	No. Training Epochs	Runtime / Epoch	No. GPUs
Macbook Pro 2012 (upgraded) 2.7Ghz CPU, 2TB SSD	1 / 8	1.9 h / 1.8 h	N/A
Google Colab Pro, 27.3 GB RAM, NVIDIA-SMI 460.32.03, CUDA v11.2	1 / 5 / 8	approx 42s	1

- Metrics

For the ClinicalBERT model, I computed an Area Under the Precision-Recall Curve (AUPRC) and Area Under the Receiver Operating Characteristic (AUROC). In addition, loss was computed and graphed during the training step. I also computed the evaluation accuracy, evaluation loss, and training loss, and recall at a precision of 80% (RP80).

- Hypotheses

In the original article, Imasogie believed it would be possible to predict readmission based on 2-day and 3-day notes. This proved to be the case using her dataset. She achieved the following values:

Metric	24 - 48 hr notes	48 - 72 hr notes
RP80	38.69%	38.02%
eval accuracy	62.56%	62.18%
eval loss	63.03%	63.05%

In the case of CAML data, I had no hypothesis to speak of. I suspected that it might be possible to perform some kind of learning through BERT as well as through classical NLP techniques, though a basic logistic regression failed with both bag-of-words and TFIDF vectorization. I was hoping that the pretrained BERT model would be better than classical ML.

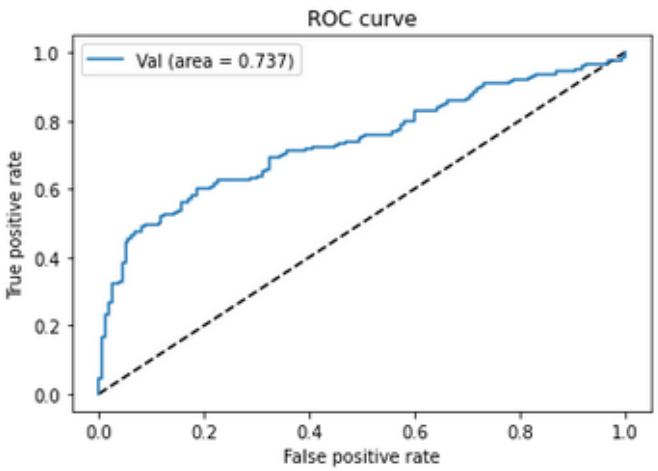
- Results: CAMLBERT – the application of ClinicalBERT to CAML data

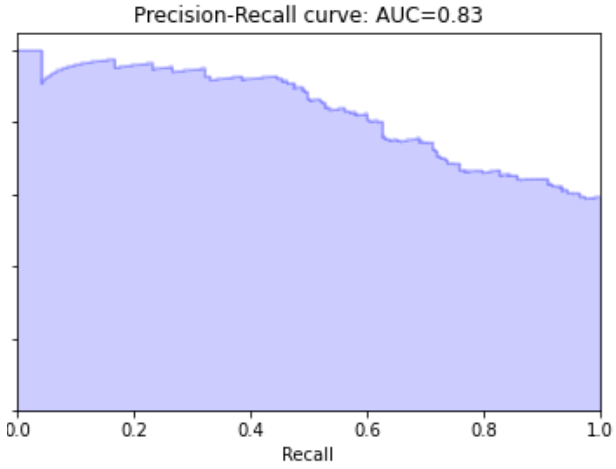
Initially, the results attained when applying ClinicalBERT were terrible (see "Original Polarity" in the table below). To address the issue, I tried a variety of learning rates. On their own, that had no effect. I then considered that the polarity of the labels might be the issue. I tried reversing the labels in the vote computation, but that too had no effect. However, when I reversed the labels for all the data (0 → 1 and 1 → 0), that resulted in an immediate improvement in the AUROC and AUPRC curves.

Metric	Original Polarity	Reversed Polarity
training loss	0.72	0.68

Metric	Original Polarity	Reversed Polarity
RP80	0.00	0.63
eval accuracy	0.39	0.61
eval accuracy rev logits	0.61	0.39
eval loss	0.73	0.67
AUROC	0.26	0.74
AUPRC	0.28	0.83

This led to the startling conclusion that ClinicalBERT actually expects "negative" events to be





1 and "positive" events to be 0. I can find no other explanation for this result. Note that theoretically one can either change the initial labeling, or one can reverse the values of the thresholds. I found the latter approach harder to get to work.

Communication with Authors

I communicated with Nwamake Imosagie via LinkedIn on two occasions. The first was to compliment her on her very interesting article, and to ask her to fix the link to her Jupyter Notebook code. The second was to ask about details on the second article she had written. We are now in communication via LinkedIn. She was able to provide the link to code for her BERT notebook, though she did not have time to provide a link for the original NLP notebook (she is on maternity leave).

I also communicated with Kexin Huang, one of the authors of the original ClinicalBERT article. He provided me with an updated link to his repo. I decided not to use his code, however, because I did not have access to the MIMIC III data. Instead I opted for the pre-trained model resulting from his study which Imosagie used.