Score Clinical Patient Notes

Suchitra Pudipeddi
MS in Health Informatics

1. INTRODUCTION

Following the covid-19 epidemic, the world learned that the medical profession is the most vital industry. We would not be able to live without the medical team (doctors, nurses, medical technicians, and cleaning professionals). As a result, we as a team were inspired to work on something that would be valuable to that area and may potentially save time and money.

In the United States, after visiting a patient, a doctor is required to make up a medical note for a variety of reasons, the most essential of which is to maintain track of the patient's medical history and symptom changes. As a result, fellow doctors manually rate the clinical note using a rubric already in place. This takes a large amount of time, as well as financial and human resources. To top it off, all physicians must go through training and be tested on it when they take the USMLE (United States Medical Licensing Examination), which is a necessity before they can begin their medical education.

We will strive to discover an automated method to translate clinical ideas from the test rubric to many ways these concepts are conveyed in the patient clinical note in this project, which will allow a professional doctor to spend less time evaluating the notes of medical students and interns. We will perform three models using the data supplied on Kaggle to examine the effects of such an approach. The job will be divided among the three members (Maria, Rahul, and Li), who will each run a model and then compare the accuracy output we receive in our attempt utilizing joint efforts.

2. ABSTRACT

A patient's notes are the sole way for a doctor to communicate with another doctor or with a patient. When a patient comes to the hospital with a condition, this is the first thing a doctor does. This determines whether a patient's diagnosis is correct; this cannot be determined on the first day of a doctor's career; it takes feedback on their notes from other doctors with extensive experience in writing patient notes, including patient problems, physical examinations, test reports, complaints, possible diagnosis, and follow-up treatment. This takes a long time to learn and access, but it can be enhanced with the help of Natural Language Processing and Machine Learning Algorithms.

The clinical skills examination, in which medical interns interact with patients who have been trained to portray specific clinical diseases and write patient notes, which are then rated or scored by experienced physicians using rubrics that are outlined with respective patient diseases and important concepts, which are referred to as features in this project. The ranking of the patient notes is greater if there are more features discovered. We implement a few algorithms on the given patient history data that can automatically assess or grade a patient's notes based on the annotations and discoveries because this strategy requires a lot of human and financial resources.

3. DATASET

The data was obtained straight from kaggle, with the entire zip file including four files, namely features. csv - features rubric for each clinical case, patient notes.csv - history of patient notes, train.csv, and test.csv We clean the data first and then merge all the files into a single dataset file because the data is so raw. In our dataset, the input is as simple as "patient_history" - Text detailing important information related by the patient during the encounter (physical exam and interview), “feature” - A clinically relevant concept, which are both strings and the output is the start and end point location of the "patient_history" string

pointing out the main feature, in medical terms mapping to clinical terms.

3.1 Important Terms

Clinical Case: The scenario (e.g., symptoms, complaints, concerns) the Standardized Patient presents to the test taker (medical student, resident, or physician). Ten clinical cases are represented in this dataset.
Patient Note: Text detailing important information related by the patient during the encounter (physical exam and interview).
Feature: A clinically relevant concept. A rubric describes the key concepts relevant to each case.

3.2 Training Data

Patient_notes.csv - A collection of about 40,000 Patient Note history portions.
pn_num - A unique identifier for each patient note. case_num - A unique identifier for the clinical case of a patient note. patient_history - The text of the encounter as recorded by the test taker.
features.csv - rubric features (key concepts) for each clinical case feature_num - A unique identifier for each feature.
case_num - A unique identifier for each case. feature_text - A description of the feature.
train.csv - Feature annotations for 1000 of the patient notes. id - Unique identifier for each patient note / feature pair. pn_num - The patient note annotated in this row. feature_num - The feature annotated in this row.
case_num - The case to which this patient note belongs. annotation - The text(s) within a patient note indicating a feature. A feature may be indicated multiple times within a single note. location - Character spans indicating the location of each annotation within the note.

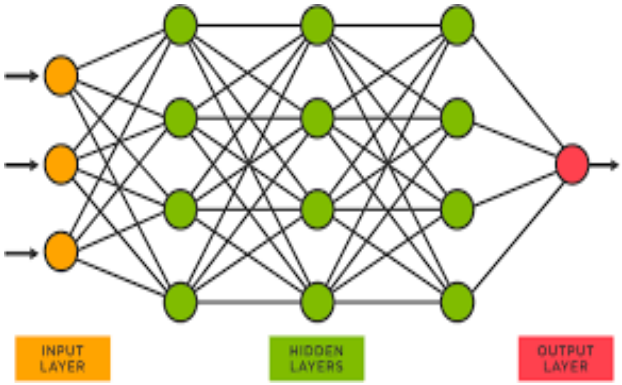
4. BACKGROUND

4.1 Neural Network

A neural network is a deep learning-focused machine learning algorithm. Where it seeks to find ways to think like a human brain in order to solve problems with data loading. It works by using layers of perceptrons to process the desired output after we load data batches into it as input. It consists of three types of layers:

- Input Layer
- Hidden Layers
- Output Layer

The input layer handles the data loaded into it, and then it passes the input through perceptrons into the hidden layers and then through the output layer. Once the perceptron passes through the data into the hidden layer, weights are randomly generated, which will then generate the bias terms values to be able to calculate the probability in the output layer to get the final classification result.



A neural network's layers might be densely connected or locally connected. All nodes or neurons from the previous layer influence the output of each node in the next layer in tightly coupled networks. In contrast, each of the nodes in the locally connected layer is only connected to a small piece of the input from the previous layer. The neighbor values are given a weight in the convolution operation, and the weighted sum of the neighbor values is used to estimate the value of the current input. Convolutional Neural Networks are a sort of neural network that is only connected locally.

4.2 LSTM

Long Short-Term Memory is a kind of a Recurrent Neural Network that has the ability to remember previous values and use it to predict future values. A Recurrent Neural Network consists of three layers: an input layer, an output layer, and a hidden layer. The number of features determine the number of connected nodes from the input layer to the hidden layer, called synopses. These synopses are assigned weights that determine which input may pass

through the hidden layer towards the output, and which don't. And then the node Inputs are added together after they are multiplied by the weights, which is known as the summed activation. They are then transformed via an activation function to define the specific output. During the training process, the weight assignments and errors produced may not produce the best results the first time it is trained, and so, in order to find the optimum values for the errors and weight, the errors are back propagated into the network towards the hidden layer. The procedure is then repeated, known as epochs, and weights are repeatedly adjusted. The hidden layer acts as a storage device that remembers the information captured in previous stages in order to predict the next set of data. The output layer then generates probabilities for each output and selects the one with least error. Each LSTM is a set of cells and gates where data streams are stored for usage when handling future data streams. The gates either allow data to pass through, or allow the cells to forget the data stream. There are three gates in each LSTM that control the flow of data:

- Forget Gate: outputs 0 in order to "forget" data, and outputs 1 in order to let data pass through and allow the option of it to be remembered.
- Memory Gate: decides which data should be remembered.
- Output Gate: decides the yield out of each cell

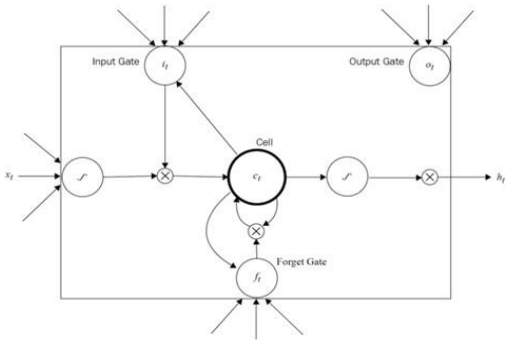


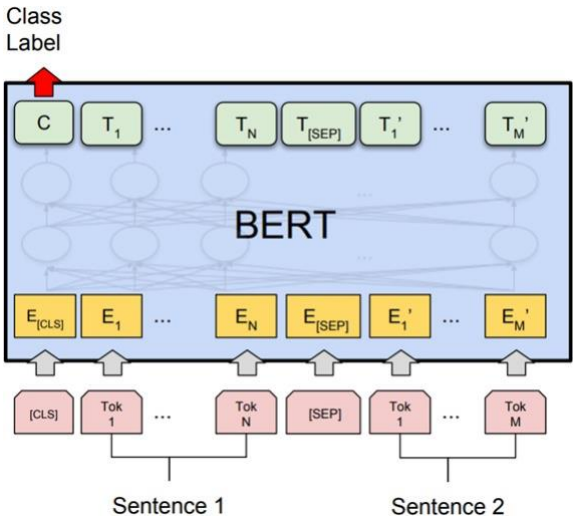
Figure 1

4.3 BERT

The Bidirectional Encoder Representations from Transformers is an important milestone in the NLP world. It inspired many models based on this architecture, and one of the most important models is BERT. BERT stands for Bidirectional Encoder Representations from Transformers. BERT is a pre-trained language model using the encoder part of Transformer architecture.

The input of BERT is a representation of each token. Each token consists of three parts: token embedding, segment embedding and position embedding. Token embedding is the word vector of each word or special character. The segment embedding is to identify different sentences. The position embedding is the location of each word. The final input will be the sum of these three embeddings.

The output of the BERT model has two parts. The first is class token which means the output of the [CLS] token. This class token is to predict classification problems like sentiment classification. The second part is the output of other word tokens. These tokens are used in some token-level tasks like question answering and sequence labeling.



5. APPROACH

Three models are attempted to be implemented: a neural network, an LSTM, and a BERT model. We utilize models from packages and libraries that have been implemented. nltk, pandas, numpy, matplotlib, sklearn, gensim-for word embedding, and pytorch are the libraries we use in this project. We convert the output as a sequence of 0 and 1 since the output is the location of each annotation. The length of the sequence is the same as the input length, the annotated location area is

assigned 1 and the remainder is assigned 0. Then this is a situation similar to Named Entity Regression. To assess the output, we convert it to sequence and compare the actual label with the predicted f1 score of the label. We also print each model's run history, which includes average train loss, average validation loss, and learning rate etc. to compare each model.

To begin, we generate a dataset that solely contains the patient notes, characteristics, and annotation locations. We combine features into the patient notes to the corresponding feature id's because the features are in a different file. Now that we have a data set with the needed inputs, we turn the outputs into the sequence shown above. The annotation of the words, or the important words from the patient notes, is to be highlighted or returned after evaluating the entire phrase, and indeed the column location consists of a pair of integers, start and end point of each important feature located in the patient notes phrase in string format. The output is labeled as 0s and 1s based on the importance given in the location column. The label is first considered a fixed length for all labels. We tokenize the whole patient notes and features columns, merge them together into the same column, adding the features into the patient notes, each label must have a fixed length of equal to the maximum number of tokens in the updated patient notes column, which is the input to our model. The locations are stored with 1's and the rest of the data is padded with 0's so the output labels always generate the same length of sequence. We split the data into 4 folds, each consisting of a randomly equal number of data values.

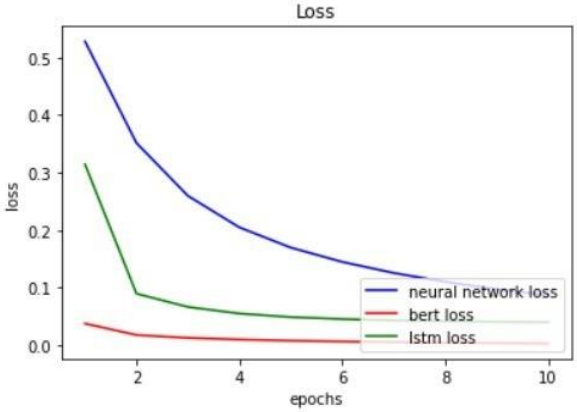
We then created a Neural Network model, feeding features to the model, generating a sequence with fixed output length, which is then converted into a set labels, using the sigmoid function, since the labels consist of 0's and 1's and later converted the label sequence into a set of data points (start and end point of the important terms in the input phrase). Following this, we trained and evaluated the LSTM and BERT model.

6. RESULTS

We train three models on the dataset, that is neural network, lstm and BERT. For the neural network model, we put an embedding layer at the beginning of the neural network. For the lstm model, we just use the normal lstm and add two linear layers at the end of the lstm model so that we can predict the class of each token. For the BERT model, we also add a linear model at the end of BERT original output.

We train the model for 10 epochs, using BCEWithLogitsLoss in pytorch library, the optimizer we use is AdamW and the learning rate is 1e-5. The results we get are shown below.

First we calculate the loss. Instead of using the result obtained from BCEWithLogitsLoss, we make an update to the print loss, that is, to use average sentence loss as the output. We can draw a graph showing the loss of three models below.



From the graph above, we can see that with the epoch increases, the loss decreases. And the loss of the BERT model is the lowest while the loss of the neural network model is the highest. So from the loss, we can conclude that the BERT model is better than LSTM and the neural network is the worst model.

Another thing we calculate to evaluate the model is the precision, recall and f1 score of the sentence. The table is shown below.

	precision	recall	f1 score
neural network	0.001	0.033	0.013
lstm	0.044	0.0002	0.0003
Bert	0.636	0.728	0.679

From the table below, we can say that the Bert model is relatively better than the other two

models. It can predict some right locations. However, the other two models weren't as good because they cannot predict any correct values. The reason may be because this is a very long sentence and both the neural network and Lstm cannot learn long term dependencies in this sentence and thus get a very bad result.

7. CONCLUSION

The results prove that, Basic Neural Network, LSTM models do not show as accurate results as the BERT Model. We prefer considering the BERT model to be the best among the others because it shows an error of around 0.6 and even takes the least time to train its model which is around 6 minutes on GPU, Neural Network and LSTM takes more than 6 minutes considering the number of epochs.

Since, lot of data cleaning, organizing and preprocessing with predicting the locations of the annotated terms in the patient notes was done in this project, there are multiple approaches that can be done to improve the accuracy and better performance of the models. We can improve batch layers, change values of hidden values and hidden layers, compare the results and choose the best among them.

As the whole project is based on real time patient notes analysis, a single mistake can cause a major impact. Since this is a kaggle competition challenge project, there are various approaches in which few of them are cited in the references below.

8. REFERENCES

1. https://www.kaggle.com/competitions/nbme-score-clinical-patient-notes
2. https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html
3. https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/
4. https://medium.com/nwamaka-imasogie/neural-networks-word-embeddings-8ec8b3845b2e
5. https://arxiv.org/abs/1810.04805
6. https://nlp.stanford.edu/seminar/details/jdevlin.pdf
7. https://medium.com/read-a-paper/bert-read-a-paper-811b836141e9
8. https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html