Medical Notes: Classifying and Grading Medical Examinations using Transformers

Kamaljeet Ghotra
School of Information
University of California, Berkeley
kghotra@berkeley.edu

Nicholas Brathwaite
School of Information
University of California, Berkeley
brathwaiten16@berkeley.edu

Dylan Jin
School of Information
University of California, Berkeley
dylanjin@berkeley.edu

Abstract

Automated Scoring of Clinical Patient Notes

“The pen is mightier than the sword.” This adage is especially true in medicine, where a doctor’s ability to take complete and accurate patient notes is key in diagnosing and treating patients; a tool used more frequently than the scalpel. Physicians make an estimated 80% of their diagnoses based solely on a patient’s medical history.[1] Therefore, an essential part of medical training is learning to be attentive during patient interactions, making careful observations, and taking comprehensive notes. Assessing a medical student’s note-taking ability used to be a part of the United States Medical Licensing Examination Step 2. The exam required students to interact with simulated patients (actors trained to portray specific medical conditions) and write a patient note. However, due to the COVID-19 pandemic and other issues with the Step 2 Clinical Skills exam, USMLE decided to merge parts of the Clinical Skills exam into Step 3.

Introduction

One of the primary issues with the Step 2 Clinical Skills exam was its high cost. Much of the high cost was due to the need for trained physicians to grade the patient notes based on a predetermined set of features for each patient. The physicians scored each note by determining how many correct features a student included in their patient note compared to a theoretical maximum. An automated method of grading the exams would greatly reduce the costs of administering the exam and increase availability.

Objective

The current scoring method is a time and resource-intensive process. Our proposed solutions using Natural Language Processing (NLP) will help overcome the practical barriers in patient note scoring, making the approach more transparent and interpretable. It will also help ease the development and administration of assessments, enabling medical practitioners to explore the full potential of patient notes in revealing information relevant to clinical skills assessment.

A significant challenge is correlating the multitude of ways medical practitioners express the same concepts due to the abundant jargon, abbreviations, and acronyms used. For example, the patient can express the concept of "loss of interest in activities" as "no longer plays tennis." Other challenges include mapping concepts by combining multiple text segments or cases of ambiguous negation such as “no cold intolerance, hair loss, palpitations, or tremor,” which corresponds to the feature “lack of other thyroid symptoms.”[2]

Methods

One solution to the issue of recognizing various representations of the same features is the use of encoder-decoder models. Conceptually, these models work by encoding the meaning behind the original text and then subsequently decoding the meaning using another set of words, depending on the target/objective. Theoretically,

[1] Summerton N; The medical history as a diagnostic technology. Br J Gen Pract. 2008 Apr58(549):273-6. doi: 10.3399/bjgp08X279779.

[2] https://www.kaggle.com/competitions/nbme-score-clinical-patient-notes/overview/evaluation

texts containing similar features would have similar encodings, despite any differences in the specific wording. Text encoding has obvious advantages in the application of grading medical notes.

Data

The datasets we used for the problem statement and a description of the fields in each file are as follows:

Patient_notes.csv
Contains 42,146 note submissions the examinees wrote during the patient encounter (physical exam and interview).

- *pn_num* - A unique identifier for each patient note.
- *case_num* - A unique identifier for the clinical case a patient note represents.
- *pn_history* - The encounter text recorded by the test taker.

Features.csv
Contains the official list of the concepts relevant to each one of the ten unique clinical cases. The number of unique features in each clinical case range from 9 to 18, with an average of 14.3 features per 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
Contains a selection of 1000 notes (100 from each clinical case) and the correct features they correctly contain, as determined by the graders.

- *pn_num* - A unique identifier for each patient note.
- *feature_num* - The feature annotated in this row.
- *case_num* - A unique identifier for each case.
- *annotation* - The features in the patient note matching those indicated in Features.csv. A feature may be indicated multiple times within a single note.
- *location* - Character spans indicate each annotation's location within the note. Multiple spans may be needed to represent an annotation, in which case a semicolon delimits the spans.

Models

The transformer-based encoder-decoder model, introduced by Vaswani et al. in the famous Attention is all you need paper, is currently the de-facto standard encoder-decoder architecture in NLP.[3] Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT) are pre-trained neural networks using the transformer architecture. However, there are some key differences between the two models[4]:

- Training: BERT incorporates masked language modeling (MLM), where a certain percentage of the input tokens are randomly masked. BERT is trained to predict the masked tokens based on the surrounding context. In contrast, GPT is an autoregressive transformer decoder model. Each token is predicted and conditioned on the previous token.

- Bi-directionality: GPT is one-directional, whereas BERT is a bidirectional model that processes text in both directions. This bi-directionality allows BERT to capture more complex relationships between words and better understand the context of a sentence.

- BERT generates all its outputs simultaneously, whereas GPT generates one word at a time.

GPT and BERT are powerful pre-trained NLP models that can be fine-tuned for various NLP tasks, depending on the characteristics of the data being analyzed. We worked on 5 different variations of those models: Roberta, BioBert, BioMed-Roberta, Deberta, and GPT-3.

Model 1 - Roberta:

First, we aimed to develop a model for text fragmentation to predict the start and end of a string within the input text. We fine-tuned the pre-trained Roberta-base model using offset mapping, token_type_ids, and input_ids. We used a learning rate of 1e-5 and a batch size of 8 for the fine-tuning process. The goal was to achieve high precision, recall, and F1 scores on the validation set.

[3] https://huggingface.co/blog/encoder-decoder
[4] https://huggingface.co/docs/transformers/model_summary

Model 2 - Bio-BERT:

We implemented a pre-trained Bio-BERT model to help with the encoding and decoding of medical text within our data. We constructed the model using a large-based BERT model and fine-tuned it with millions of text examples from medical practices. The same model can be used for NER (Named Entity Recognition), QA (Question & Answering), Summarization, and COER (Coreference Resolution). In our project, we focused on cross-referencing the annotations with patient notes to determine the presence of a medical feature within the text. We used a learning rate of 0.01 and a max sequence length of 200 characters. For the model we integrated a dropout layer with a rate of 0.3 and used a dimension size of 200. We decided to use a batch size of 50 over 2 epochs and an early callback to monitor the loss during training.

Results

After successfully loading the model and encoding the text data for the patient's history, we documented the results below:

Batch Size	Learning Rate	Epochs	Dropout Rate	Val Loss	F1 Score
50	0.01	2	0.3	0.619	0.001

The Bio-BERT model showed good baseline results, however due to the time it took to run it was difficult to train it with different parameters.

Model 3 - BioMed-Roberta:

The Biomed-Roberta model is an extension of the Roberta model trained with biomedical information. Similar to the Bio-Bert model, it too was trained using data from PubMed to be used in clinical practices. We used all the same parameters as the Bio-BERT model except for a max sequence length of 100 instead of 200.

Data Preparation

To prepare the data for the text fragmentation task, we used the offset mapping technique to divide the text into smaller fragments without overlap. Offset mapping involves identifying the start and end positions and mapping each token in the fragment to its corresponding position in the original document. We also used token_type_ids to identify the fragments in the original document. Finally, we encoded the input text into input_ids using the vocabulary of the pre-trained model. These encoded input_ids were used to train the model to predict the start and end positions of the text fragments in the original documents.

The other method we used for text fragmentation utilized the location and annotation features to identify the presence of feature text within patient notes. This allowed us to create a separate label to refer to within the binary classification. Then using each model's encoder and tokenizer to compute input ids, token type ids, and attention masks for the original document.

Model Architecture

We used the pre-trained Roberta-base model for fine-tuning. We fine-tuned the model for text classification using the labeled fragments. We added a layer on top of the Roberta-base model to output binary labels for input_ids and offset mapping.

Training

We trained the model using a batch size of 8 and a learning rate of 1e-5. We used the binary cross-entropy loss as the loss function for training. We trained the model for five epochs on the training set. We used the Adam optimizer with default parameters.

Results

After training, the model achieved a precision of 0.7993, recall of 0.6582, and F1 score of 0.7219 on the validation set. The training loss was 0.0049, and the validation loss was 0.0098. These results demonstrate that the model performed well on the text classification task.

LR	BS	F1
9.00E-05	8	0.533
1.00E-05	4	0.579

5.00E-05	4	0.547
1.00E-05	8	0.800

In this project, we fine-tuned the pre-trained Roberta-base model for text fragmentation using offset mapping, token_type_ids, and input_ids. We achieved high precision, recall, and F1 scores on the validation set. These results demonstrate the effectiveness of using pre-trained models for text classification tasks.

Model 4 - Deberta-V3:

Introduction

Decoding enhanced BERT with disentangled attention (DeBERTa) (He et al., 2021) is an improvement over the BERT and RoBERTa (Liu et al., 2019) models using two novel techniques: (1) The disentangled attention mechanism, where each word is represented using two vectors that encode its content and position, respectively, and the attention weights among words are computed using disentangled matrices on their contents and relative positions, respectively; (2) Enhanced mask decoder, which incorporates absolute positions in the decoding layer to predict the masked tokens when pre-training the model. DeBERTa has significantly improved model pre-training efficiency and advanced state-of-the-art performance on many NLU downstream tasks.We tried a different model apart from Roberta-base. The goal of this project was to fine-tune the DeBERTa-v3 model on text fragment problems using the Masked Language Modeling (MLM) technique. The project aimed to improve the performance of the model on text fragment problems.

Objectives

To train and fine-tune the DeBERTa-v3 model using the MLM technique on the data and to use the pre-trained model for downstream tasks.To evaluate the performance of the fine-tuned model on text fragment problem.

Methodology

The DeBERTa-v3 model was chosen as the base model for this project. It was fine-tuned on the data using the MLM technique. The pre-trained model was then used for downstream tasks. We used the following hyperparameters for the pre-trained model:

Model	Masked Token Prob	Train BS	Val BS	LR	loss	Epochs
mlm-deberta-v3	0.3	8	4	1.00E-05	1.65	5
mlm-deberta-v3	0.2	4	4	0.01	1.65	5

Fine-tuned the Deberta-v3 pre-trained model with below parameters and loss, f1.

Batch Size	Learning Rate	Epochs	Dropout Rate	Val Loss	F1 Score
8	5e-4	5	0.1	0.0816	0.001
4	9e-4	3	0.1	0.005	0.021
4	1e-5	10	0.1	0.094	0.001

Results

After fine-tuning the model using the above hyperparameters, we got the following loss and metrics for the pre-trained model:

Batch Size	Learning Rate	Epochs	Dropout Rate	Val Loss	F1 Score
8	5e-4	5	0.1	0.0816	0.001
4	9e-4	3	0.1	0.005	0.021
4	1e-5	10	0.1	0.094	0.001

During the training the metrics for the fine-tuned model are worse with F1 score nearly zero. For future work, we can continue looking at the attention matrix and logits to see which tokens are causing the score to be lower. For now, this model is unsuitable for scoring patient notes.

Model 5 - GPT-3:

GPT-3 was developed by OpenAI and released in June 2020. It is a neural network-based model trained on massive amounts of text data (45 terabytes). GPT-3 functions by generating contextually relevant responses to various prompts. The release of ChatGPT in November

2022, based on GPT-3.5, sparked worldwide interest due to its ease of use and impressive performance in question-answering compared to popular search engines.

The specific GPT-3 engine used in this project is Text-Davinci-003. Davinci is a variant of GPT-3.5 that OpenAI released in July 2021. It can generate more complex text passages with greater fluency than the original GPT-3 model[5].

Methodology

We compared two techniques to exploit GPT-3's unique features:

First, we followed a more traditional approach to grading each patient note. We used the GPT model to parse the text of each note and determine which of the expected features are present in the text via a prompt. We scored each note, dividing the feature count Davinci provided by the maximum possible number of features. These scores were then compared to the expected scores, calculated from the labeled data provided by the official graders.

For our baseline, we used an untuned model of GPT using Davinci. The average error achieved was 0.36.

We used a subset of 10 cases for initial testing and tuning. GPT tuning focused on prompt engineering to get GPT-3 to match features in our test set. After experimentation, we found that a one-shot prompt asking for the count provided the best results. Once satisfied with our prompt, we set the max tokens to 1. Due to time and cost constraints due to the relatively large dataset, we focused on getting the count of matching features. Future work, if permitted, would include a full listing of matched features for each patient note and calculating F1 scores.

Our second test was to leverage the generative ability of GPT to create "ideal" patient notes containing all of the expected features. We compared each examinee's note to the corresponding generated notes to generate a score. We used BERT to tokenize the texts and create embeddings. Subsequently, PyTorch computed the cosine similarity between the embeddings.

Results

The results of our GPT Feature Scoring and Cosine Similarity with GPT-generated "ideal" notes are shown in Figures 1 and 2.

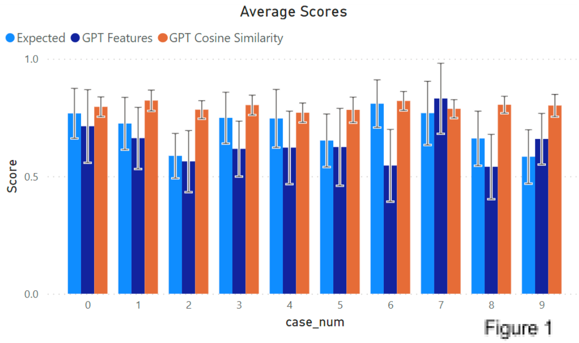
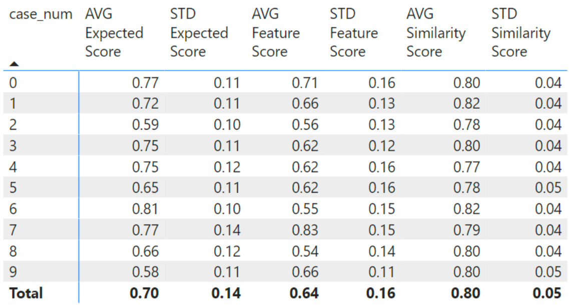


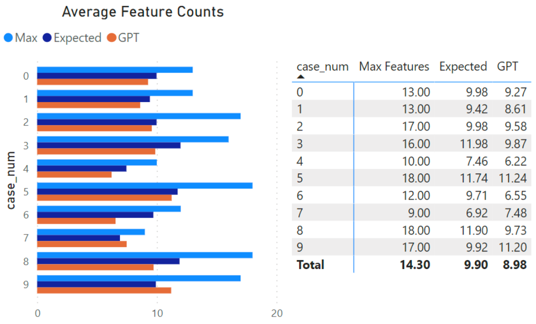
Figure 1



case_num	AVG Expected Score	STD Expected Score	AVG Feature Score	STD Feature Score	AVG Similarity Score	STD Similarity Score
0	0.77	0.11	0.71	0.16	0.80	0.04
1	0.72	0.11	0.66	0.13	0.82	0.04
2	0.59	0.10	0.56	0.13	0.78	0.04
3	0.75	0.11	0.62	0.12	0.80	0.04
4	0.75	0.12	0.62	0.16	0.77	0.04
5	0.65	0.11	0.62	0.16	0.78	0.05
6	0.81	0.10	0.55	0.15	0.82	0.04
7	0.77	0.14	0.83	0.15	0.79	0.04
8	0.66	0.12	0.54	0.14	0.80	0.04
9	0.58	0.11	0.66	0.11	0.80	0.05
Total	**0.70**	**0.14**	**0.64**	**0.16**	**0.80**	**0.05**

Figure 2

As seen in Figure 1, the GPT Feature scores match reasonably well with the Expected Scores except for Case 6. We also saw this trend in the average feature counts from Figure 3. Although the average error across all cases was 0.22, this amounts to just an average of one feature difference from the expected count and a 39% increase in performance over our baseline. It is also well within the standard deviations seen in Figure 1.



case_num	Max Features	Expected	GPT
0	13.00	9.98	9.27
1	13.00	9.42	8.61
2	17.00	9.98	9.58
3	16.00	11.98	9.87
4	10.00	7.46	6.22
5	18.00	11.74	11.24
6	12.00	9.71	6.55
7	9.00	6.92	7.48
8	18.00	11.90	9.73
9	17.00	9.92	11.20
Total	**14.30**	**9.90**	**8.98**

Figure 3

From the results, we see that GPT tends to undercount the features. The results suggest that human graders can extract more features

[5] https://platform.openai.com/docs/models/overview

from the notes. Looking at the outputs from GPT, we see some of the common mismatches were frequently due to the use of shorthand, such as “M” for “male” and gross misspellings that the grader caught, but GPT missed.

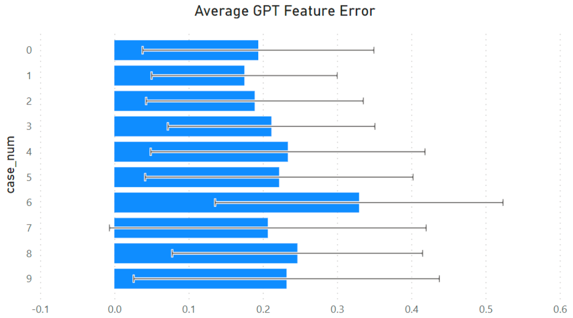


Figure 4

Overall the results are very promising, especially considering that this used a pre-trained model without additional training from this dataset.

Although not directly comparable to the feature-based scores, the cosine similarity data showed an extremely weak correlation of 0.1 with the expected scores. The cosine similarity values were generally consistent across all cases, as Figures 1 and 2 show.

Cosine similarity did not indicate whether an examinee’s note would score high or low, making it unsuitable for this task. Using GPT to generate patient notes as a gold standard for comparison and grading still has potential but will require additional research. Future work comparing the generated notes and examinee notes could use state-of-the-art two-tower models and tools like Google’s Scalable Nearest Neighbors (ScaNN).

Conclusion

To grade patient notes, we found the BERT-based BioMed-Roberta and GPT-3 based Davinci models to be reliable alternatives for determining the number of features present within an examinee’s submission. The advantage of the GPT model was no additional specialized training was necessary, unlike the BERT model, and the process was entirely autonomous. The cost of implementation is trivial compared to the cost in time and capital for a trained physician to grade the patient notes. Therefore, GPT-3 would be our ideal recommendation for a low-cost, high-throughput solution for grading the USMLE Step 2 Clinical Skills exam.