The Tell-Tale Heart: Using ECG Notes to Diagnose Patients

Sanjay Elangovan
UC Berkeley, School of Information
sanjay.elangovan@berkeley.edu

Abstract

Diagnosis prediction based on electrocardiogram (ECG) notes could be a valuable asset to doctors in the critical care unit. In this paper, several embeddings and models are evaluated to see which can best predict a patient's diagnosis. The model which performed best was a simple bag-of-words model, though BERT embeddings pre-trained on the clinical domain performed nearly as well. However, none of the models breached 40% accuracy, likely due to the high degree of similarity among ECG notes. Code for this project can be found at: https://github.com/sanjay-elangovan/PredictingECGDiagnoses

1 Introduction

Over the past few decades, Electronic Health Records have seen increased adoption, resulting in an abundance of long-form clinical data. These are notes written by physicians and nurses, covering every stage of a patient's visit from diagnosis to tests to treatment and follow up consultations.

Physicians are required to make decisions about patients under strong time constraints. In such settings, a system which can analyze notes and infer likely diagnoses could support physicians in making decisions.

Of particular interest are electrocardiogram (ECG) notes. These notes are descriptions of ECG readings, highlighting features of the waveform and potential underlying causes. An example is shown in Figure 1. ECG diagrams are difficult to read and interpretation can be subjective, so a system which can reliably predict common diseases based on ECG notes could prove to be valuable.

2 Background

Similar problems have been tackled by others, and most use the same dataset: Medical Information Mart for Intensive Care (MIMIC-III), which is a freely available database that contains de-identified patient information. This includes caregiver notes from patient intake and discharge for more than 40,000 adult patients who stayed in the critical care units at Beth Israel Deaconess Medical Center between 2001 and 2012.

The paper "What's in a Note? Unpacking Predictive Value in Clinical Note Representations" (Boag et al 2018) compared a few different methods of embedding and modeling notes from MIMIC-III to diagnose patients with the five most common primary diseases. They approached the problem by using a simple Bag of Words as a baseline model. They also built an LSTM which learned from sequences of patient notes, and used word2vec embeddings as an in-between to compare against the other two methods. Diagnosis accuracy for all three methods hovered around 87%, though the LSTM performed slightly better than the others.

More recently, this problem was approached in "Clinical Outcome Prediction from Admission Notes.." (Aken et al 2020). Aken differed from Boag by creating pseudo-intake notes from discharge summaries in MIMIC-III and predicting on more than 1200 different diagnoses. They used BERT models which had been pre-trained on clinical data, including ClinicalBERT and

> Sinus rhythm. Compared to the previous tracing of [**2178-5-28**] findings are as previously noted. In addition, P waves are more peaked, raising consideration of right atrial overload. PR segment deviations are noted which have been present before and could be due to non-specific atrial repolarization effect, although pericardial process is not excluded. Clinical correlation is suggested. TRACING #2

Figure 1: An example ECG reading. Yellow highlights are descriptions of the waveform itself and green highlights contain diagnostic information. The red highlight indicates removed personal identifiers.

BioBERT Base. They found that the Bag of Words approach resulted in 76% accuracy in diagnosing patients, but the pre-trained models diagnosed ~82% of patients accurately. In addition, Aken trained their own BERT model, called CORe, on top of BioBERT weights using the MIMIC-III dataset, medical transcriptions and clinical notes available online, and Wikipedia articles describing diseases. This new model was about 83% accurate in predicting diagnoses.

Most papers require the combination of multiple note types or the creation of a new note type that may not actually be available to physicians at time of diagnosis. To my knowledge, nobody has specialized their model on ECGs. For this project, I will follow the approach set by Boag and classify ECG notes in MIMIC-III to predict the most common diseases in that category of notes. I will use sparse categorical accuracy as a loss metric, and set up Bag of Words as the baseline model.

3 Methods

This project has three stages:

1. Obtaining Data
2. Embedding Data
3. Model building

3.1 Obtaining Data

The MIMIC-III database contains 26 tables, including tables for notes ('NOTEEVENTS') and patient diagnoses ('DIAGNOSES_ICD'). These tables can be linked together using the field "SUBJECT_ID", which is unique for every patient in the dataset. By combining notes with diagnoses, a labeled database can be created.

The MIMIC-III dataset is accessible through Google BigQuery. Appendix A contains some queries which were used for exploratory data analysis and to download data.

There are several nuances to consider when working with the MIMIC-III dataset:

1. Note structure
2. Quantity of diagnoses
3. Multiple diagnoses per patient
4. Data cleanup

3.1.1 Note structure

There are several categories of notes, such as *Nursing/Other*, *ECG*, *Nutrition*, and *Discharge Summary*. Some categories, like *Radiology* are follow a defined format, while others are mostly freeform, like *Nursing* notes.

Note lengths also vary significantly, with some being only a few words to others being hundreds of words. This is an important consideration while generating embeddings for notes. BERT, for example, is limited to 512 tokens without special processing.

For this project, I will focus only on *ECG* notes. This average note length is 209 words long, with notes ranging from 1 word to 1149 words. However, the bottom 75% of notes are under 264 words long.

3.1.2 Quantity of Diagnoses

Diagnoses are recorded using the International Classification of Diseases, Ninth Revision (ICD-9). This classification set contains more than 15,000 codes for diseases and disorders, of which 6,984 are represented in the MIMIC-III dataset.

As a result of the wide range of codes, the number of notes per diagnosis can be very small, making it difficult for the model to identify unique features for disease. One approach to reducing the number of possible diagnoses is to look at the disease category rather than the disease itself. For example, ICD-9 codes for both burns and fractures fall under the category *Injury and Poisoning*. An example of how data can be collected using this approach is shown in Appendix A.

Diagnoses in ECG notes mostly fall under *Diseases of the Circulatory System*. A few diseases are prevalent in this category, shown in Table 1. Since the most common three codes appear more frequently than the rest, I will ask my model to diagnose patients with either the top three most common diseases or output that the disease is something else entirely.

3.1.3 Multiple diagnoses per patient

Most patients come to the critical care unit with multiple health issues, which translates to multiple ICD-9 diagnoses for each patient. As a matter of

fact, only 187 patients in the whole dataset have been diagnosed with a single ICD-9 code.

This presents an issue when training models because each data point has multiple labels. However, this problem is easily resolved by using the "SEQ_NUM" field in 'DIAGNOSES_ICD'. This field correlates with the relevance of a diagnosis – For example, a diagnoses with SEQ_NUM=1 is more relevant than a diagnoses with SEQ_NUM=5. The dataset can thus be limited to diagnoses with SEQ_NUM=1.

3.1.4 Data Cleanup

Several steps can be taken to remove distracting tokens from the data. Patient identifiers, as shown in red highlighting in Figure 1, can be removed using a REGEX match. In addition, text in the dataset can be lowercased, and special characters like "!#*&", can be removed. Some special characters are left in – For example, dashes remain in the dataset because some terms like "A-V" should be treated as one word rather than two. After distracting characters are removed, some notes are totally blank. These notes can now be removed.

The last step in data cleanup is to balance the dataset. The top three diseases all have slightly different counts, so it helps to balance the dataset by ensuring that each label has the same number of counts. After balancing the dataset, each of the four labels has 17,045 examples, for a total of 68,180 rows in the dataset.

3.2 Embedding Data

An essential step in building the model is picking an appropriate embedding to capture meaning from the sentence inputs. Below are the embedding methods used in this project:

1. TF-IDF
2. spaCy
3. BERT Models (ClinicalBERT, CORe)

3.2.1 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure that multiples together how many times a word appears in a document and the inverse document frequency of the word across all the whole dataset. In the process, however, TF-IDF strips documents of all context. This embedding will be used in the baseline model since it requires no pre-training and is not specific to clinical datasets.

Disease Name	ICD-9 Code	Record Count
Coronary atherosclerosis of native coronary artery	41401	23053
Unspecified septicemia	0389	21496
Subendocardial infarction, initial episode of care	41071	19140
Acute respiratory failure	51881	14120
...	...	...
Cortex (cerebral) laceration	85121	1
Closed fracture of vault of skull	80030	1

Table 1: ICD-9 Codes in ECG notes with SEQ_NUM=1

3.2.2 spaCy

spaCy is an easy-to-use library for natural language processing published by MIT. The library contains GloVe vectors trained on Common Crawl, which is a repository of web crawl data. spaCy embeddings will provide an interesting comparison as they are not trained on a clinical dataset but may still be able to draw meaning from the dataset. spaCy returns both pooled outputs and sequential outputs, with pooled outputs having the shape (number of notes, 300), and the sequence outputs having the shape (number of notes, number of words/note, 300).

3.2.3 BERT Models (ClincialBERT & CORe)

ClinicalBERT and CORe are both public language models based on BERT, but pre-trained on additional clinical domain specific data. ClinicalBERT was pre-trained on 100,000 random clinical notes, and CORe was trained on a wide corpus of clinical text. I'd expect these models to do better than TF-IDF because they understand context, and better than spaCy since they are trained on clinical datasets.

Similar to spaCy embeddings, BERT models return both pooled and sequential embeddings. The pooled embeddings are of the shape (number of notes, 768) while the sequential outputs are (number of notes, number of words/note, 768).

One approach to using BERT is to calculate the embeddings and feeding them into a model. Another approach is fine-tuning BERT on the dataset before passing the embeddings through a

feed-forward layer for classification. Both approaches were explored in this project.

3.3 Model Building

Several different model options are available to leverage the outputs of the embeddings:

1. Fully Connected Network
2. LSTM
3. CNN
4. BERT

3.3.1 Fully Connected Network

In a fully connected network, each node in a layer is connected to the nodes in the next layer. The inputs to a Fully Connected Network are 2-dimensional. Fully connected networks were generated for the following embeddings:

- TF-IDF
- spaCy Pooled
- ClincalBERT (pooled, not fine-tuned)
- CORe (pooled, not fine-tuned)

A common problem with fully connected networks is overfitting. In identifying the most optimal model, fully connected networks were generated with varying numbers of epochs, L1 and L2 regularization, additional layers, and using different activation functions (ReLU, TanH, Sigmoid) for the final layer.

3.3.2 LSTM

A Long Short-Term Memory (LSTM) network is a type of recurrent neural network which has feedback connections. The following embeddings were used for the LSTM model:

- spaCy (sequential)
- ClinicalBERT (sequential)
- CORe (sequential)

LSTMs were tested with multiple LSTM layers, differing numbers of units in the LSTM layer, and by adding fully connected layers before the LSTM layers.

3.3.3 CNN

A Convolutional Neural Network is a model often used for sentence classification tasks, where different filters are used to identify different features in the text. The same embeddings that were used for LSTM were also tested with CNN. Optimal CNNs were found by varying the number and sizes of filters and kernels.

3.3.4 BERT

The outputs of BERT can be fine-tuned on training data and passed into a fully connected layer in order to classify the text. The following two embeddings were fine-tuned:

- ClinicalBERT (fine-tuned)
- CORe (fine-tuned)

4 Results and Discussion

Table 2 contains the results from the best embedding and model combinations. Model summaries for the top models per embedding are provided in Appendix B.

The best combination was the baseline model which used TF-IDF embeddings, with both fine-tuned BERT models a few points less accurate. None of the models breach 40% accuracy.

4.1 Literature Comparison

None of the models performed well when compared against the literature. Boag et al 2018 was able to achieve 82% accuracy with a Bag of Words approach, while Aken et al 2021 reached

Embedding	Best Performing Model	Total Trainable Parameters	Accuracy
TF-IDF	**Fully Connected**	**10,407,004**	**38.92%**
spaCy (pooled)	Fully Connected	1,204	24.38%
spaCy (sequential**)**	**CNN**	**44,266**	**33.59%**
ClinicalBERT (fine-tuned)	**BERT**	**108,313,348**	**37.40%**
ClinicalBERT (pooled)	Fully Connected	3,076	25.45%
ClinicalBERT (sequential)	LSTM	66,484	35.22%
CORe (fine-tuned)	**BERT**	**108,313,348**	**37.20%**
CORe (pooled)	Fully Connected	3,076	25.03%
CORe (sequential)	LSTM	66,484	35.85%

Table 2: Model accuracies and parameters. Best models per embedding are bolded.

84% accuracy with CORe and 82% accuracy with ClinicalBERT. The major difference between this project and theirs was the scope of data. Aken created pseudo-intake notes from the discharge summary, and Boag combined several different types of notes. This project focused purely on ECG notes. The low accuracy is in part due to how similar ECG notes are to one another. In the entire dataset of ~68,000 documents, there are only 4200 unique words, and many phrases are repeated for different diagnoses. There aren't many words that are distinct to a diagnosis, so training is difficult even with additional context from BERT.

Another issue could have been the "catch-all" category for the model. Since this category is just a random sample from every other type of diagnosis, there are likely few shared characteristics among notes in the category.

A third issue may just be size of training data. The dataset contained ~68,000 documents, but that means there are only ~17,000 examples per diagnosis.

4.2 Hyperparameter Tuning

Each variation of the fully connected model presented different advantages and disadvantages. Keeping all other hyperparameters constant, adding neurons to the fully connected layer increased train accuracy but decreased validation accuracy. Similarly, adding epochs increased train accuracy, but validation accuracy stagnated after a certain number of epochs. The additional neurons and epochs contributed to overfitting, where the model was beginning to memorize the training set without generalizing. One way to combat overfitting is by adding regularization. L1 & L2 regularization on the kernel decreased overfitting, but L2 regularization alone decreased accuracy of the model without decreasing overfitting.

In the LSTM model, neither stacking LSTM layers nor adjusting the number of units in the LSTM changed the accuracy very much. Both changes do decrease the loss, though it does take longer to train the model with additional complexity.

For the CNN, a few different kernel sizes and filters were compared, but the accuracy and loss for ClinicalBERT and CORe didn't change much. For the spaCy model, adding larger filters and additional kernels improved the accuracy of the model. The additional complexity in the model allows the network to identify new combinations of patterns in the text.

4.3 Model Comparisons

In general, the accuracy of a model improves as the number of parameters increases. This is apparent when comparing the pooled models against the sequential models. The pooled models have only a few thousand parameters, and accuracies for those models hover around random chance.

CORe and ClinicalBERT performed comparably well, which makes sense since they were trained on similar data. The fine-tuned BERT models outperformed the pooled and sequential outputs, but performed slightly worse than the simple TF-IDF model. This seems to imply that the context captured by BERT is not that relevant to diagnosing patients using ECG notes.

All three sequential models performed comparably (34-35% accuracy). These models use frozen weights to embed the text, so the similar accuracies make sense. Since the BERT models were pre-trained on clinical data, they did slightly outperform spaCy. The spaCy embeddings worked best with the CNN. The layers in the CNN likely allowed the model to find relationships in the text and understand context better.

5 Conclusion

Modeling ECG notes is challenging because many notes are similar in content. Through this project, it was found that the best way to predict diagnoses is to use a TF-IDF model, though fine-tuned pre-trained BERT models perform similarly well.

One improvement could be to take advantage of the sequential nature of ECG notes. For example, in Figure 1 there are several references to previous tracings. This time-ordering was ignored in this project, but an RNN which treats each note as a sequential input could be a more accurate way to interpret the data. Some challenges with this approach would be capturing the meaning of the entire note in a single embedding, and ensuring that the same number of tracings exist across the dataset.

References

Boag, Willie et al. “What's in a Note? Unpacking Predictive Value in Clinical Note Representations.” AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science vol. 2017 26-34. 18 May. 2018

Aken, Betty et al. “Clinical Outcome Prediction from Admission Notes using Self-Supervised Knowledge Integration.” arXiv:2102.04110 [cs]. 8 Feb 2021.

Huang, Kexin et al. “ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission.” arxiv:1904.05342 [cs]. 29 Nov 2020.

Hashir M, Sawhney R. Towards unstructured mortality prediction with free-text clinical notes. J Biomed Inform. 2020 Aug;108:103489. doi: 10.1016/j.jbi.2020.103489. Epub 2020 Jun 25. PMID: 32592755.

A BigQuery Commands

Identifying the most common category of notes for each diagnosis type:

```
SELECT
CASE
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(0)') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^1[0123])')) THEN
    "Infectious And Parasitic Diseases"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(1[456789])') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^2[0123])')) THEN
    "Neoplasms"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(2[4567])'))
    THEN "Endocrine, Nutritional And Metabolic Diseases, And
    Immunity Disorders"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(2[8])')) THEN
    "Diseases Of The Blood And Blood-Forming Organs"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(2[9])') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^3[01])')) THEN
    "Mental Disorders"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(3[2345678])'))
    THEN "Diseases Of The Nervous System And Sense Organs"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(3[9])') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^4[012345])')) THEN
    "Diseases Of The Circulatory System"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(4[6789])') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^5[01])')) THEN
    "Diseases Of The Respiratory System"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(5[234567])'))
    THEN "Diseases Of The Digestive System"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(5[89])') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^6[012])')) THEN
    "Diseases Of The Genitourinary System"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(6[34567])'))
    THEN "Complications Of Pregnancy, Childbirth, And The
    Puerperium"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(6[89])') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^7[0])')) THEN
    "Diseases Of The Skin And Subcutaneous Tissue"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(7[123])') )
    THEN "Diseases Of The Musculoskeletal System And
    Connective Tissue"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(7[45])') ) THEN
    "Congenital Anomalies"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(7[67])') ) THEN
    "Certain Conditions Originating In The Perinatal Period"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(7[89])') ) THEN
    "Symptoms, Signs, And Ill-Defined Conditions"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^(8)') OR
    REGEXP_CONTAINS(ICD9_CODE, r'(^9)')) THEN "Injury
    And Poisoning"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^([vV])') ) THEN
    "Supplementary Classification Of Factors Influencing Health
    Status And Contact With Health Services"
 WHEN (REGEXP_CONTAINS(ICD9_CODE, r'^([eE])') ) THEN
    "Supplementary Classification Of External Causes Of Injury And
    Poisoning"
END
    AS ICD9_CATEGORY,
    CATEGORY AS NOTE_TYPE,
    COUNT(TEXT) AS NUM_RECORDS,
FROM
    `physionet-data.mimiciii_notes.noteevents` AS notes
LEFT JOIN
    `physionet-data.mimiciii_clinical.diagnoses_icd` AS diags
USING
    (SUBJECT_ID)
WHERE
    ISERROR IS NULL
GROUP BY
    ICD9_CATEGORY,
    NOTE_TYPE
ORDER BY
    ICD9_CATEGORY,
    NUM_RECORDS DESC,
     NOTE_TYPE
```

Collecting all primary diagnoses from ECG notes, renumbering them from 0-4, filtering out blank notes

```
SELECT DISTINCT
    SUBJECT_ID,
    CASE
     WHEN ICD9_CODE = "4019" THEN 0
     WHEN ICD9_CODE = "4280" THEN 1
     WHEN ICD9_CODE = "41401" THEN 2
     WHEN ICD9_CODE = "42731" THEN 3
     ELSE 4
    END AS DIAGNOSIS_CODE,
    TEXT
FROM
    `physionet-data.mimiciii_notes.noteevents` AS notes
LEFT JOIN
    `physionet-data.mimiciii_clinical.diagnoses_icd` AS diags
USING
    (SUBJECT_ID)
WHERE
    ISERROR IS NULL
    AND SEQ_NUM = 1
    AND CATEGORY = "ECG"
    AND (REGEXP_CONTAINS(ICD9_CODE, r'^(3[9])') OR
REGEXP_CONTAINS(ICD9_CODE, r'(^4[012345])'))
    AND NOT (REGEXP_CONTAINS(TEXT, r'(see corresponding
office note for interpretation)'))
ORDER BY DIAGNOSIS_CODE DESC
```

B Model Summaries

TF-IDF:

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_1 (InputLayer)        [(None, 202, 300)]        0         []

 conv1d (Conv1D)             (None, 200, 2)            1802      ['input_1[0][0]']

 conv1d_1 (Conv1D)           (None, 199, 10)           12010     ['input_1[0][0]']

 conv1d_2 (Conv1D)           (None, 198, 20)           30020     ['input_1[0][0]']

 global_max_pooling1d (GlobalMa  (None, 2)             0         ['conv1d[0][0]']
 xPooling1D)

 global_max_pooling1d_1 (Global  (None, 10)            0         ['conv1d_1[0][0]']
 MaxPooling1D)

 global_max_pooling1d_2 (Global  (None, 20)            0         ['conv1d_2[0][0]']
 MaxPooling1D)

 concatenate (Concatenate)   (None, 32)                0         ['global_max_pooling1d[0][0]',
                                                                  'global_max_pooling1d_1[0][0]',
                                                                  'global_max_pooling1d_2[0][0]']

 dropout (Dropout)           (None, 32)                0         ['concatenate[0][0]']

 dense (Dense)               (None, 10)                330       ['dropout[0][0]']

 dense_1 (Dense)             (None, 4)                 44        ['dense[0][0]']

 dense_2 (Dense)             (None, 4)                 20        ['dense_1[0][0]']

=================================================================================================
Total params: 44,226
Trainable params: 44,226
Non-trainable params: 0
_________________________________________________________________________________________________
```

spaCy Embeddings:

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 4200)]            0

 dense (Dense)               (None, 2000)              8402000

 dropout (Dropout)           (None, 2000)              0

 dense_1 (Dense)             (None, 1000)              2001000

 dense_2 (Dense)             (None, 4)                 4004

=================================================================
Total params: 10,407,004
Trainable params: 10,407,004
Non-trainable params: 0
_________________________________________________________________
```

ClinicalBERT:

```
BertClassifier(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(28996, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
```

BERT Encoding Layers (removed for brevity)

```
      .
      .
      .
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.5, inplace=False)
  (linear): Linear(in_features=768, out_features=4, bias=True)
  (softmax): Softmax(dim=1)
)
```

CORe:

```
BertClassifier(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(28996, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
```

BERT Encoding Layers (removed for brevity)

```
      .
      .
      .
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.5, inplace=False)
  (linear): Linear(in_features=768, out_features=4, bias=True)
  (softmax): Softmax(dim=1)
)
```