Achieving Comparable Performance to DeBERTa-v3 with ModernBERT's & Extended Token Capacity in Medical Text Classification

Zehra Korkusuz
Applied NLP @ University of Trento
2024-2025

Abstract

This project experiment with ModernBERT, a transformer model with extended token capacity, in medical text classification using the NBME competition dataset. It shows that ModernBERT, with its 16x larger maximum token size compared to traditional BERT models, achieves comparable performance to DeBERTa-v3-large while offering better training stability and generalization. Also combines pseudo-labeling techniques with ensemble methods, achieving scores within the top 12% of the competition leaderboard. The results highlight the potential of extended token capacity models in medical NLP tasks, particularly for processing lengthy clinical texts as well as retrieval oriented applications such as RAG.

1. Introduction

Medical text classification presents unique challenges due to the complexity of clinical terminology and the length of patient records. While models like DeBERTa-v3 have shown strong performance in this domain, and exceeding domain spesific Clinical and BioBERT variants, their limited token capacity constrains their ability to process longer medical texts holistically. This work explores ModernBERT as an alternative, leveraging its extended token capacity while maintaining competitive performance.

The contributions of this work include:

- A comparison of ModernBERT with state-of-the-art models in medical text classification
- Analysis of the impact of extended token capacity on model performance
- Impact of pseudo-labeling effectiveness in medical domain tasks and data leakage in the training

2. Dataset & Task Description

The NBME dataset comprises clinical patient notes with annotated medical concepts across multiple categories. The dataset includes:

- Approximately 40,000 patient note histories
- 1,000 annotated patient notes across 10 clinical cases
- Feature annotations with corresponding character spans
- A comprehensive rubric of medical features and concepts

2.1. Preprocessing

Several preprocessing steps are required for the data quality:

1. Correction of typographical errors in annotations
2. Standardization of medical terminology and abbreviations
3. Alignment of character spans with annotated text
4. Token-aware truncation and padding strategies

The standardization of the medical terminology and abbreviations (might have some issues with tokenization) is not implemented. 1st, 3rd and 4th are implemented during preprocessing.

3. Model Architecture & Methods

3.1. Model Architectures

ModernBERT

- Maximum token capacity: 16,384 tokens
- Linear attention mechanism: O(n) complexity
- Rotary position embeddings

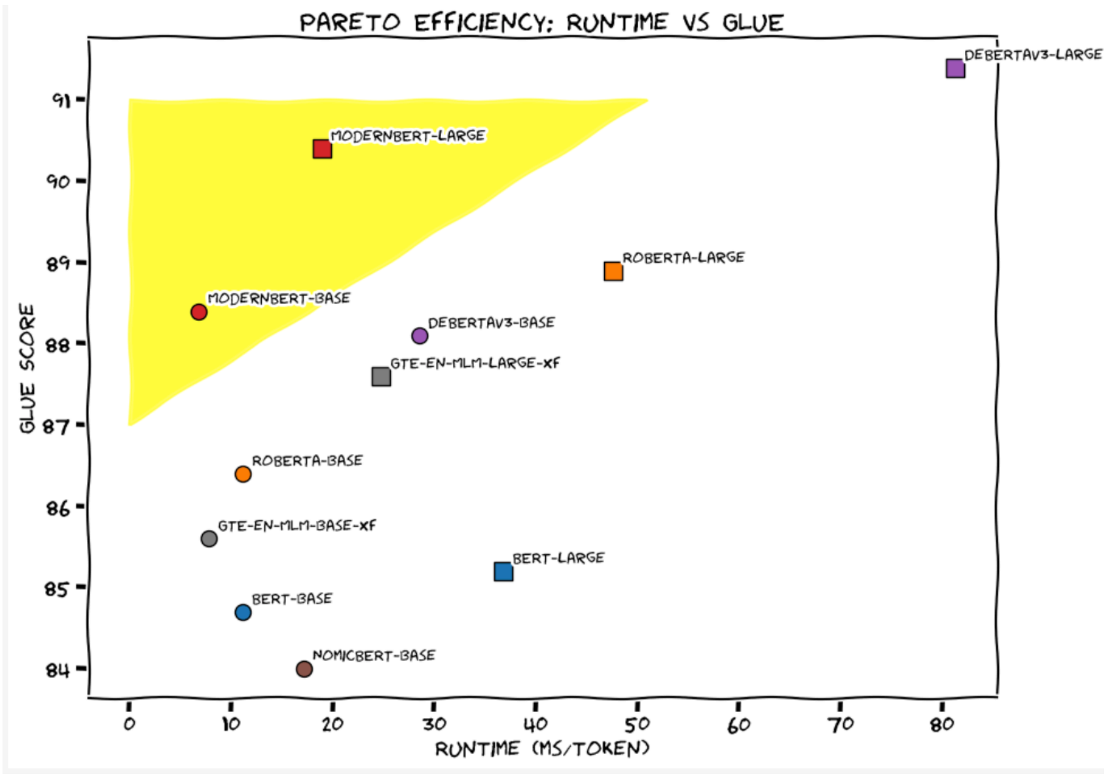


Chart 1: Runtime vs Glue score comparison of the BERT variants[1] (He et al., 2023)

- Architecture specifications:
    - Hidden size: 768
    - Number of attention heads: 12
    - Number of layers: 12
    - Intermediate size: 3072

DeBERTa-v3-large

- 24 transformer layers
- Hidden size: 1024
- Attention heads: 16
- Maximum sequence length: 512
- Disentangled attention matrices
- Virtual adversarial training

DeBERTa-v3-base

- 12 transformer layers
- Hidden size: 768
- Attention heads: 12
- Enhanced mask decoder
- Gradient checkpointing support

ClinicalBERT[2]

- BERT-base architecture
- Clinical vocabulary extensions
- Domain-specific pre-training

- Maximum sequence length: 512

3.2 Pseudo-labeling Strategy

Our pseudo-labeling approach utilized DeBERTa-v3-large following these steps:

1. Train the initial model on annotated data
2. Generate predictions on unlabeled data
3. Select high-confidence predictions
4. Integrate selected predictions into training data

3.3 Training Optimization

To address computational constraints, models trained with mixed precision and gradient checpoints using PyTorch framework.

Memory Optimization:

- Gradient accumulation (4 steps)
- Gradient checkpointing
- Mixed precision (bfloat16)

Bfloat16 is commonly used in ML/DL training to speed up the processes by reducing the memory storage.

[1] https://huggingface.co/blog/modernbert

[2] Haven't trained it yet it was used in the competition and in the ensemble models

Also there were some operational variations in the latest versions of the pytorch model loading and modern bert tokenization.

For instance in ModernBERT, is_token_type_ids are omitted.

Model	N-fold	Epoch	Accuracy	Pseudo
DeBERTa-v3-base	5	5	86%	train.csv
DeBERTa-v3-large	5	5	88%	train.csv
DeBERTa-v3-large	1	4	97%	train.csv + pseudo.csv
ModernBERT-base	1	4	80%	train.csv
ModernBERT-base	1	4	95%	train.csv + pseudo.csv

Table 1: Model Performances with and without pseudolabels included

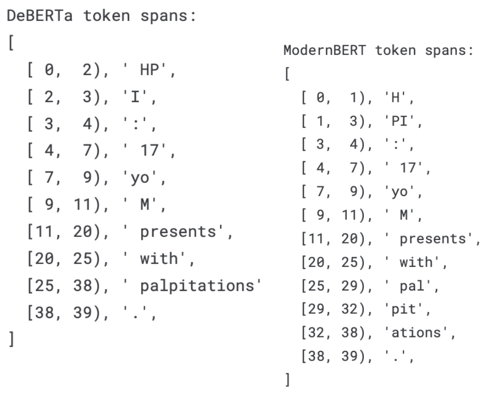
Key observations include:

1. Compared the logging output of DeBERTa v1 base model DeBERTa-v3 model has more stable training yet converges slower. DeBERa v1 model had more fluctations in the gradients during each epoch.
2. DeBERTa-v3-base model achieves 86% accuracy only utilizing the training dataset. Even though in the competition it barely makes the top 50% of the leaderboard among the 1400+ competitors, the winning solutions has achieved 89% accuracy and most of them used adversarial training techniques, ensemble models as well as utilizing pseudolabels while managing data leakage by not using averages directly in ensembles.

3. Generated pseudolabels using DeBERTa-v3-large model considering that high quality (accurate pseudolabels) would be more efficient in the inference time but turns out it also leads to more data leakage. Even though we didn't use the ground truth and it is not officially a data leakage, it is a problem in terms of generalization to unseen samples.
4. ModernBERT provides faster training time, faster convergence and stable gradients during the training as well as larger context size.
5. ModernBERT also had less data leakage in the training with pseudolabels compared to DeBERTa but it could be because how we differently trained both models using pseudolabels.
   - DeBERTa-v3-large model was already trained with train dataset. So, we uploaded the checkpoints - which were the result of best model at each fold, and then at each fold, we used this checkpoint to finetune the model with pseudolabels which were also grouped with K-Fold sampling.
   - ModernBERT with pseudolabel is trained by concatenating the train dataset with pseudolabels.

It could be also due to difference latent space representation in the initial model vs finetuned model helping with better generalization.

6. Each model has different tokenizations strategy, so we need to be mindful of it at the ensembling, requiring post processing, particularly in considering of whitespaces because ModernBERT uses similar tokenization[3]



```
DeBERTa token spans:
[
  [ 0,  2), ' HP',
  [ 2,  3), 'I',
  [ 3,  4), ':',
  [ 4,  7), ' 17',
  [ 7,  9), 'yo',
  [ 9, 11), ' M',
  [11, 20), ' presents',
  [20, 25), ' with',
  [25, 38), ' palpitations'
  [38, 39), '.',
]
```

```
ModernBERT token spans:
[
  [ 0,  1), 'H',
  [ 1,  3), 'PI',
  [ 3,  4), ':',
  [ 4,  7), ' 17',
  [ 7,  9), 'yo',
  [ 9, 11), ' M',
  [11, 20), ' presents',
  [20, 25), ' with',
  [25, 29), ' pal',
  [29, 32), 'pit',
  [32, 38), 'ations',
  [38, 39), '.',
]
```

So, ModernBERT has similar tokenization approach to Roberta
Yet ModernBERT also consider whitespaces as Deberta

i.e. [4, 7), ' 17' vs [5, 7) for the text chunk' 17',

[3] https://www.kaggle.com/code/zehrakorkusuz/nbme-roberta-deberta-modernbert-tokenization

Conclusion

ModernBERT provides faster training time, faster convergence and stable gradients during the training as well as larger context size making it a suitable model to handle cases more holistically in analyzing the documents in the medical domain.

Future Work

In this project, in order to classify the medical feature spans in the text

<start > **pn_history** CLS **feature_text** <end> used. Using ModernBERT, It would also allow further approaches such as adding description of the medical feature since appropriate iron values can have variations in the meaning in different groups such as different demographics groups.

Resources

1. **Project Code Repository**: For training notebooks, outputs, and evaluation details, check out the codebase on GitHub: https://github.com/zehrakorkusuz/Applied_NLP

Kaggle Notebooks:

2. **Tokenization Differences**: Details on tokenization differences that need to be taken into account during ensemble, including RoBERTa, DeBERTa, and ModernBERT tokenization: https://www.kaggle.com/code/zehrakorkusuz/nbme-roberta-deberta-modernbert-tokenization
3. **DeBERTa v3 Base Baseline Train**: A baseline implementation for DeBERTa v3 base model: https://www.kaggle.com/code/zehrakorkusuz/nbme-deberta-base-baseline-train
4. **Training DeBERTa v3 Large**: A notebook focused on training the DeBERTa v3 large model: https://www.kaggle.com/code/zehrakorkusuz/training-deberta-v3-large

References

Kaggle Discussion on NBME Score Clinical Patient Notes. (2021). Retrieved from https://www.kaggle.com/competitions/nbme-score-clinical-patient-notes/discussion/323156

Yasufumi Nakama. (2021). *NBME DeBERTa Base Baseline Train*. Retrieved from https://www.kaggle.com/code/yasufuminakama/nbme-deberta-base-baseline-train

Microsoft. (2021). *DeBERTa v3 Base*. Hugging Face. Retrieved from https://huggingface.co/microsoft/deberta-v3-base

Hugging Face. (2021). *Modern BERT*. Retrieved from https://huggingface.co/blog/modernbert

Neos960518. (2021). *Ensembling DeBERTa Models (Bronze Medal Top 8)*. Retrieved from https://www.kaggle.com/code/neos960518/ensembling-deberta-models-bronze-medal-top-8

A. Appendix: Evaluation & Micro F1[4]

This competition is evaluated by a micro-averaged F1 score.

For each instance, we predict a set of character spans. A **character span** is a pair of indexes representing a range of characters within a text. A span `i j` represents the characters with indices `i` through `j`, inclusive of `i` and exclusive of `j`. In Python notation, a span `i j` is equivalent to a slice `i:j`.

For each instance there is a collection of ground-truth spans and a collection of predicted spans. The spans we delimit with a semicolon, like: `0 3; 5 9`.

We score each character index as:

- TP if it is within both a ground-truth and a prediction,
- FN if it is within a ground-truth but not a prediction, and,
- FP if it is within a prediction but not a ground truth.

Finally, we compute an overall F1 score from the TPs, FNs, and FPs aggregated across all instances. Example

Suppose we have an instance:

ground-truth	prediction
0 3; 3 5	2 5; 7 9; 2 3

These spans give the sets of indices:

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

ground-truth	prediction
0 1 2 3 4	2 3 4 7 8

We therefore compute:

- TP = size of {2, 3, 4} = 3
- FN = size of {0, 1} = 2
- FP = size of {7, 8} = 2

Repeat for all instances, collect the TPs, FNs, and FPs, and compute the final F1 score.