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

Zehra Korkusuz **Professor: Jacopo Staiano**

Abstract

This work explores the potential of ModernBERT—a transformer model with an extended token capacity—in the domain of medical text classification using the NBME competition dataset. ModernBERT's architecture supports an 8,192-token context (16 times larger than traditional BERT models), allowing it to holistically process lengthy clinical documents. My experiments demonstrate that ModernBERT achieves competitive performance relative to DeBERTa-v3-large while offering enhanced training stability and generalization. By reimplementing a DeBERTa ensemble model, I achieved top 12% in the leaderboard ranking in the competition. While I was unable to run an ensemble of ModernBERT and DeBERTa, my experiments demonstrate their compatibility for future ensembling. Additionally, my attempts at using pseudo-labeling led to data leakage, suggesting that additional techniques, such as adversarial training, are necessary to make it effective in practice.

1 Introduction

Medical text classification poses significant challenges due to the complexity of clinical language and the sheer volume of patient records. Standard transformer models, including BERT and its domain-specific variants, are often constrained by a fixed token limit that hampers their ability to capture the full context of long clinical narratives. Although models like DeBERTa-v3-large have advanced performance in medical NLP tasks, their limited token capacity restricts comprehensive analysis of extended texts.

In this work, I investigate ModernBERT, a transformer model engineered with an extended token capacity that supports up to 8,192 tokens. This substantial increase enables ModernBERT to analyze complete patient records without truncation, thereby capturing intricate dependencies and clinical nuances that may be lost in traditional models.

Term	Description
Cases/Classes	Clinical condition
Features	Labels
Feature text	Standard Label/Class Text
Patient Story	Doctor's notes about the patient
Annotation text	Text span in the patient notes related to the feature
Location	Location of the related text span classified as given feature

Table 1: Description of terms used in the dataset.

In addition to its architectural benefits, ModernBERT exhibits robust training dynamics, including stable gradient behavior and faster convergence.

The primary contributions of this study are:

- **Extended Context Processing:** Demonstrating how ModernBERT's 16x token capacity improves the holistic analysis of lengthy clinical texts.
- **Performance and Stability:** Comparing ModernBERT against state-of-the-art models like DeBERTa-v3-large, with a focus on training stability and generalization.
- **Advanced Techniques:** Integrating pseudo-labeling and ensembling strategies to enhance classification accuracy, achieving competitive results on the NBME dataset.

These findings highlight the significant potential of extended token capacity models in medical NLP, paving the way for future research in processing and understanding complex clinical data for both classification and retrieval-oriented tasks.

2 Dataset & Task Description

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

The dataset includes:

- 40.000 unique patient notes
- 1,000 annotated patient notes,

- Across 10 clinical cases (10 different cases)
- Each clinical cases have a varying number of features. For example, *Figure 1* displays the features associated with Case 0.
- We aim to predict if given the patient notes and feature, feature (clinical concept) exists in the text

  e.g. "father is diagnosed with .." (family history feature)

Feature Number	Case Number	Feature Text
000	0	Family-history-of-MI-OR-Family-history-of-myocardial-infarction

Table 2: Features (Labels) dataset

Patient Number	Case Number	Patient History
00000	0	17-year-old male, has come to the student health clinic complaining of heart pounding...
00001	0	17-year-old male with recurrent palpitations for the past 3 months lasting about 3 - 4 minutes...

Table 3: Patient Notes Dataset

ID	Case#	Pat#	Feat#	Annotation	Location
00016_000	0	00016	000	dad with recent heart attack	696 724
00016_001	0	00016	001	mom with thyroid disease	668 693
00016_002	0	00016	002	chest pressure	203 217
00016_003	0	00016	003	intermittent episodes, episode	70 91, 176 183

Table 4: Training Data

Each row in the dataset represents a feature (a condition or symptom) associated with a specific patient case. This structure helps link each patient's condition to the relevant case and detailed notes, providing a clear overview of patient health conditions and their locations in the records.

3 Preprocessing

Several preprocessing steps are required for the data quality:

3.1. Correction of typographical errors in annotations

- "diarrhoe non blooody" → "diarrhoea non bloody"

3.2. Standardization of medical terminology and abbreviations

0.48

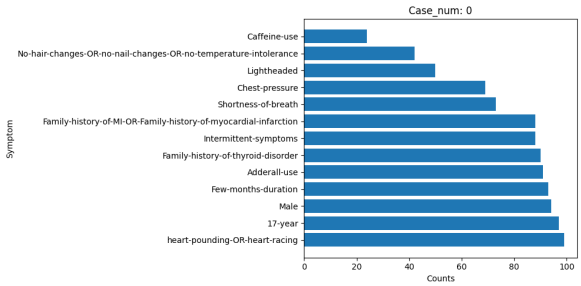

Figure 1: Features (LABELS) associated with Case 0

0.48

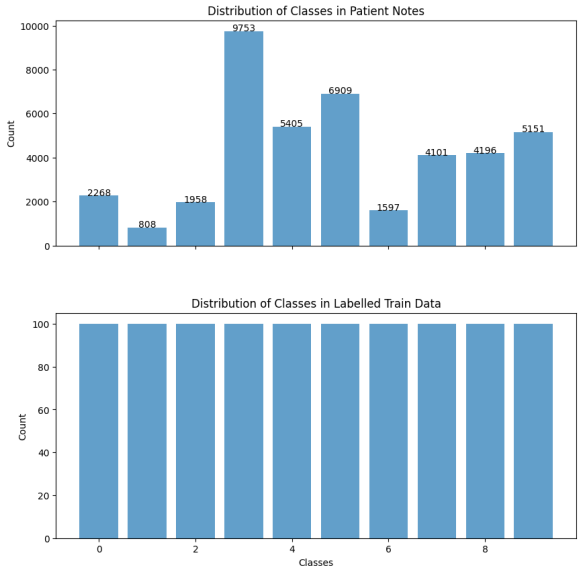

Figure 2: Distribution of cases (0-9) in the patient stories and annotated dataset.

Each case has 100 patient stories, totaling 1000 patient stories across 10 different conditions.

Figure 3: Feature distribution and case distribution in the dataset.

- Common clinical abbreviations standardized:

URI	Upper Respiratory Infection
Hx	History
ROS	Review of Systems
SH	Self-harm
FH	Familial hypercholesterolemia
Meds	Medications
PShx	Past Surgical History
FmHx	Family Medical History

- For a more extensive list, please refer to USMLE Step 2 Clinical Skills (CS) Abbreviations.

3.3. Alignment of character spans with annotated text

- Ensuring character spans precisely match

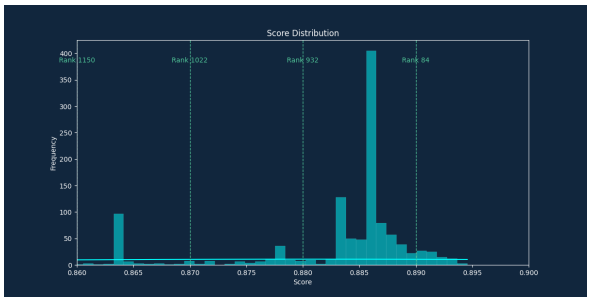

Figure 4: Distribution of the competitor model submissions

text segments

- Correction of the location "259 280" → "stool, with no blood"

3.4. Token-aware truncation and padding strategies

- Implementation of context-preserving truncation for long clinical annotations
- Special handling for partial phrases (e.g., "heart started racing...")

Note: The listed points are crucial for ensuring accurate and consistent annotation. 1st, 3rd and 4th steps are applied and 4th is not required due to the reason that max token size of the model covers the required tokens for our inputs. While working with BERT-like models, skipping preprocessing may also allow us develop more robust models as it resembles the real-world examples.

4 Competition Analysis and Model Performance

The NBME competition leaderboard analysis reveals distinct performance tiers among different model architectures (Figure 1). The distribution shows several key thresholds:

- **Top 1% (Rank 84)**: Achieved by teams using adversarial training techniques, comprehensive cross-validation, and larger model architectures. These solutions typically employed ensemble methods of multiple DeBERTa-v3 models.
- **Top 10% (Rank 932)**: Characterized by teams using DeBERTa-large (v2-v3) models, often with basic ensemble techniques and careful hyperparameter tuning.
- **Mid-range Performance (Rank 1022)**: Solutions using DeBERTa-base (v1) models, showing the importance of model capacity in achieving competitive performance.
- **Lower Performance (Rank 1150)**: Teams using domain-specific models like ClinicalBERT, demonstrating that recent general-purpose architectures can outperform domain-specialized models.

5 ModernBERT: Architectural Advantages

ModernBERT offers several significant advantages over traditional transformer architectures, making it particularly well-suited for medical text classification tasks. The most notable advantages include a **16x larger maximum token size** and improved runtime, thanks to several **architectural design enhancements**. Additionally, it achieves a competitive **GLUE score**, further solidifying its effectiveness in natural language understanding tasks. (**?**)

- **Extended Context Processing**:
  - 8,192 token context length (16x BERT's capacity)
  - Advanced sequence packing and unpadding optimizations
  - Alternating attention mechanism with 128-token local windows
- **Architectural Innovations**:
  - Rotary Positional Embeddings (RoPE) for improved position encoding
  - GeGLU activation layers replacing traditional MLP
  - Additional normalization layer post-embeddings
  - Bias term removal for enhanced parameter efficiency
- **Performance Optimizations**:
  - 2-4x faster inference compared to DeBERTa
  - Hardware-aware architecture design for optimal GPU utilization
  - Three-phase training approach: 1.7T@1024 tokens → 250B@8192 tokens → 50B annealing

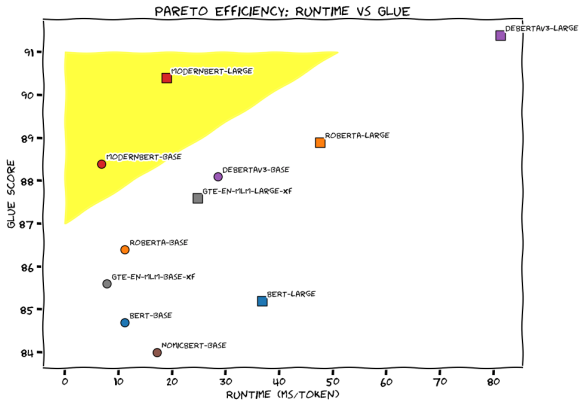

Figure 5: Pareto Efficiency - Runtime vs GLUE Score

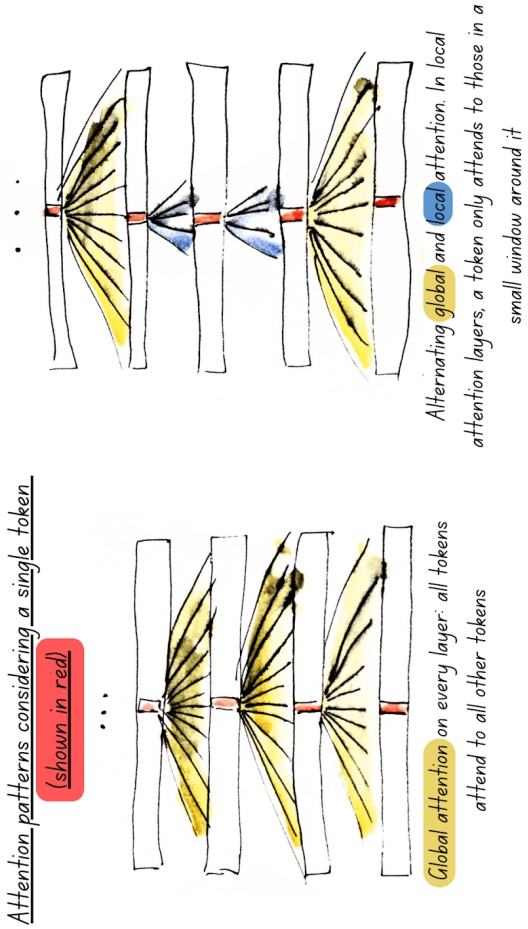

Figure 6: ModernBERT Attention Mechanism. **Alternating Attention** - global attention is applied every 3 layers, while local attention (with a 128-token window) is used in the remaining layers, which is the key factor that accelerates the computations.

6 Experiments

This section outlines the key strategies and optimization techniques used in the experiments, along with the evaluation of various models.

6.1 Pseudo-labeling Strategy

This pseudo-labeling approach utilized the DeBERTa-v3-large model we trained and followed these steps:

6.1. Train the initial model on annotated data.

6.2. Generate predictions on unlabeled data.

6.3. Select high-confidence predictions.

6.4. Integrate selected predictions into the training data.

Note that while we have started with the notebooks available on Kaggle, we had to make changes to run with the current PyTorch version which is compatible with ModernBERT and new transformers library. It requires several operational updates such as PyTorch model loading and ModernBERT tokenization, such as the omission of `is_token_type_ids` in ModernBERT.

6.2 Training Optimization

To address computational constraints, we trained the models using mixed precision and gradient checkpointing within the PyTorch framework. The specific memory optimization techniques applied are as follows:

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

Bfloat16 is commonly used in machine learning/deep learning training to speed up the processes by reducing memory storage and improving computational efficiency.

6.3 Model Evaluation

The following table summarizes the model performance based on different training setups, including the use of pseudo-labeling.

The table shows the performance of different DeBERTa and ModernBERT models, with pseudo-labeling applied to some of the models. We observe 90% in accuracy when pseudo-labeling is

Model	N-fold	Epoch	Accuracy
DeBERTa-v3-base	5	5	86%
DeBERTa-v3-large	5	5	88%
DeBERTa-v3-large (with pseudo-labeling)	1	4	97%
ModernBERT-base	1	4	80%
ModernBERT-base (with pseudo-labeling)	1	4	95%

Table 5: Model performance with and without pseudo-labeling.

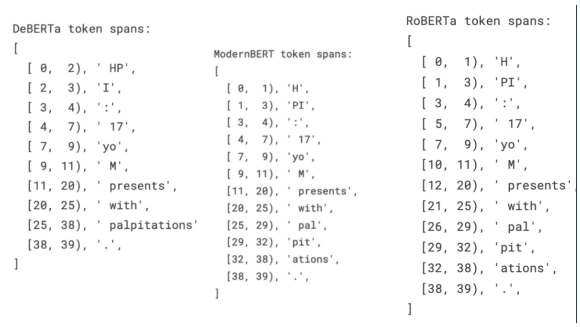

Figure 7: Tokenization by BERT Variants

applied to DeBERTa-v3-large and ModernBERT-base models. But this is due to data leakage and doesn't generalize well. Pseudolabeling requires a better approach including selection of confident responses, number of labeled dataset as well as use of it during prediction, such as instead of averaging at k-fold, utilizing a different approach or adversarial training strategies.

6.4 Tokenization Strategy and Compatibility

Each model utilizes a different tokenization strategy, and it is important to be mindful of these differences, especially when performing ensembling. One key consideration is the handling of whitespaces during tokenization, which can affect how text chunks are tokenized across models.

- **Tokenization Differences & Ensembling:** ModernBERT employs a tokenization approach similar to Roberta, but with one key distinction—it treats whitespaces similarly to DeBERTa. For example, when tokenizing the text chunk " 17", ModernBERT and DeBERTa would represent it as [4, 7) and [5, 7), respectively, with whitespace considered as part of the token.

- **Whitespace Handling in Ensembling:** Because ModernBERT considers whitespaces in a similar manner to DeBERTa, it does not require additional steps when performing ensembling with DeBERTa models. This compatibility simplifies the post-processing of model predictions, as it ensures that tokenization boundaries are consistent across models, avoiding any misalignments in the ensemble process.

- **Post-processing Considerations:** While tokenization differences are generally minimal between ModernBERT and DeBERTa, it is still important to perform post-processing in the ensembling phase to ensure that any subtle tokenization discrepancies (especially in whitespace handling) do not impact the final predictions.

6.5 Additional Notes

The following points highlight key aspects and observations during the experimentation process:

- **Training Dataset:** All models were trained using the 'train.csv' dataset, which contains the labeled data for model training.

- **Effectiveness of Pseudo-labeling:** Pseudo-labeling has proven particularly effective in enhancing model accuracy. However, its use required additional steps outlined in the previous section, such as the careful selection of high-confidence predictions and their integration into the training data which were not completed in this project.

- **Impact of PyTorch and ModernBERT Variations:** The operational variations between different versions of PyTorch and the tokenization process in ModernBERT did not significantly affect the final model performance except the fasttokenizer improving the runtime.

- **Kaggle Notebooks Contribution:** Unique contributions and code developments can be found on the Kaggle Notebooks. Each notebook showcases the changes made, with added or removed lines of code.

- **Training Stability:** In terms of gradient stability, ModernBERT showed more stable gradients compared to DeBERTa-v1 and DeBERTa-v3-large , despite starting with the lowest accuracy at the first epoch, exhibited the most stable training progress, with accuracy improving consistently in each epoch.

- **Model Behavior:** Models like ModernBERT demonstrate a gradual yet steady improve-

ment in training accuracy, which may be attributed to its robust architecture designed to handle complex data over time as well as use of a large corpus in the training. In contrast, while DeBERTa-v3-large showed good performance early on, it did not exhibit the same level of consistent improvement as ModernBERT.

- **Further Experiments and Fine-tuning:** Additional fine-tuning and experimentation with different learning rates, batch sizes, and other hyperparameters could further optimize performance, particularly for models like ModernBERT and DeBERTa-v3 are used but haven't for each run.

7 Conclusion

ModernBERT demonstrates several key advantages, including faster training times, faster convergence, and stable gradients throughout the training process. Additionally, its ability to handle larger context sizes makes it an ideal model for analyzing medical documents holistically. These attributes enable ModernBERT to effectively capture complex relationships and dependencies within medical text, making it particularly well-suited for medical document analysis tasks.

8 Future Work

In this project, we used the token structure `<start> pn_history CLS feature_text <end>` to classify medical feature spans within the text. Moving forward, the use of ModernBERT could be expanded to explore additional approaches, such as incorporating descriptions of medical features. For instance, medical terms like iron values can vary in meaning across different demographic groups. By considering these variations, we can further enhance the model's understanding and classification accuracy, making it more robust and context-aware for diverse patient populations.

9 Resources

- **Project Code Repository:** For training notebooks, outputs, and evaluation details, check out the codebase on GitHub: https://github.com/zehrakorkusuz/Applied_NLP
- **Kaggle Notebooks:**
  - **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
  - **DeBERTa v3 Base Baseline Train:** A baseline implementation for DeBERTa v3 base model: https://www.kaggle.com/code/zehrakorkusuz/nbme-deberta-base-baseline-train
  - **Training DeBERTa v3 Large:** A notebook focused on training the DeBERTa v3 large model: https://www.kaggle.com/code/zehrakorkusuz/training-deberta-v3-large

10 References

- Kaggle Discussion. (2021). NBME Score Clinical Patient Notes. 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
- Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, Iacopo Poli. (2024). Smarter, Better, Faster, Longer: A

Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. Retrieved from https://doi.org/10.48550/arXiv.2412.13663

A Appendix: Evaluation & Micro F1

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 indices 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 (True Positive):** if it is within both a ground-truth and a prediction,
- **FN (False Negative):** if it is within a ground-truth but not a prediction, and
- **FP (False Positive):** 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 with the following ground-truth and prediction spans:

Ground-truth	Prediction
$0\ 3; 3\ 5$	$2\ 5; 7\ 9; 2\ 3$

These spans give the sets of indices:

Ground-truth	Prediction
$0, 1, 2, 3, 4$	$2, 3, 4, 7, 8$

We compute:

- **TP = 3** (size of 2, 3, 4),
- **FN = 2** (size of 0, 1),
- **FP = 2** (size of 7, 8).

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