

Archaeology at BEA 2025 Shared Task: Are Simple Baselines Good Enough?

Ana-Maria Roşu
anaros766@gmail.com

Jany-Gabriel Ispas
iani.ispas@gmail.com

Sergiu Nisioi*
sergiu.nisioi@unibuc.ro

Human Language Technologies Research Center
Faculty of Mathematics and Computer Science
University of Bucharest

Abstract

This paper describes our approach to the 5 classification tasks from the Building Educational Applications (BEA) 2025 Shared Task. Our methods range from classical machine learning models to fine-tuning large-scale transformer architectures. Despite the diversity of techniques, performance differences were often minor, suggesting the presence of strong surface-level signal in the data and a limiting effect of annotation noise – particularly around the “To some extent” label. Under lenient evaluation, simple models perform competitively, showing their effectiveness in low-resource settings. Our submissions rank in the top 10 in three out of five tracks. The code and models are publicly available at: <https://github.com/ana-rosu/Archaeology-at-BEA2025>

1 Introduction

This paper presents an exhaustive set of experiments conducted for the BEA 2025 Shared Task, which revolves around assessing the pedagogical abilities of AI tutors in educational dialogues within the mathematical domain.

We start with classical machine learning methods like logistic regression over TF-IDF encodings and String Kernel SVMs, gradually scaling up to more complex approaches such as zero-shot and few-shot prompting with Mistral-7B-Instruct (Jiang et al., 2023), feature-based methods using frozen transformer representations (from models like ModernBERT (Warner et al., 2024a) and GritLM (Muennighoff et al., 2024)), decoder-style architectures such as GPT2-XL (Radford et al., 2019) combined with a linear classification head, parameter-efficient fine-tuning with LoRA adapters in 4-bit precision on Mistral-7B, as well as BERT-like classifiers (e.g., RoBERTa (Liu et al., 2019), ModernBERT (Warner et al., 2024b), DeBERTa (He et al., 2021)).

Our best-performing submissions across all tracks use fine-tuned BERT-style classifiers. Final submissions are selected in an unsystematic way due to the five-submission limit per track; we focus on choosing the models that perform best on our local validation set, while also ensuring that they differ from each other. Although most of our submissions are based on Masked Language Models, we include a broader set of experiments in this paper to document our development process and highlight that some alternative approaches remain competitive.

We place greater emphasis on Track 1 (Mistake Identification), as it is the first task we explore and serves as a foundation for the others. Some of our preliminary experiments, including prompting and decoder-based fine-tuning, are conducted exclusively on this track.

Despite using a wide range of models, we observe that performance is often surprisingly similar across setups, suggesting that model architecture may not be the dominant factor for this task. One possible explanation is that subtle annotation inconsistencies, especially between “Yes” and “To some extent”, introduce noise that limits performance (see Appendix G). We notice that tutor responses with very similar wording (e.g., “Please recheck your answer”) are labeled “Yes” in some dialogues and “To some extent” in others. In this context, the order in which training examples are presented becomes important, especially in such a small and imbalanced setting. When the model sees one interpretation early on, it may implicitly learn to generalize that decision across similar examples, reinforcing a bias. This makes the optimization sensitive to random factors such as batch order or initialization.

The “To some extent” label is the main source of difficulty in this task. Without it, the classification becomes much easier, a fact supported by the lenient evaluation scores, which reach or even

*Corresponding author.

exceed 85% F1 on all tracks but one (Providing Guidance, where the best lenient performance on the public leaderboard is 78% F1), suggesting that models perform well when ambiguity is removed from the label space.

When evaluated under the lenient setting (“Yes” and “To some extent” are merged into a single class), traditional machine learning models have surprisingly strong performance even with minimal effort, using default configurations. As shown in Table 9, the validation accuracies achieved with these models are very close to the best public leaderboard results, with gaps between 0.37%-4.15%. In terms of Macro F1, these models also achieve competitive scores, with gaps between 3.70%-10.41%, demonstrating that pedagogical signal can be captured effectively under a binary framing, making them strong baselines in scenarios with constrained resources.

The presence of strong surface-level signal may allow even simple models to perform well. Another potential hypothesis is that there is simply not enough data for larger models to generalize better.

Our team’s submissions were competitive across all tracks:

Track 1 (Mistake Identification): 8th out of 44 teams

Track 2 (Mistake Location): 12th out of 31 teams

Track 3 (Providing Guidance): 13th out of 35 teams

Track 4 (Actionability): 7th out of 29 teams

Track 5 (Tutor Identification): 6th out of 20 teams

Team ranks are based on the results according to the main shared task metric – exact Macro F1 score.

2 Data and Tasks

2.1 Shared-Task Tracks

The data provided for this shared task builds on MRBench, a dataset of short alternate-turn dialogues sourced from MathDial (Macina et al., 2023) and Bridge (Wang et al., 2024). Each dialogue is annotated for eight pedagogical dimensions based on a unified evaluation taxonomy introduced by Maurya et al. (2025a). This taxonomy reflects core learning sciences principles and builds on prior work in AI tutor evaluation (Tack and Piech, 2022; Daheim et al., 2024; Wang et al., 2024)

The task focuses on four key dimensions – which also form the first four of the five tracks in the BEA 2025 shared task:

Track 1 - Mistake Identification: determine if

the tutor identifies the student’s mistake.

Track 2 - Mistake Location: determine if the tutor pinpoints where the mistake occurs.

Track 3 - Providing Guidance: determine if the tutor gives helpful and relevant feedback.

Track 4 - Actionability: determine if the student can clearly understand what to do next.

Track 5 - Tutor Identification: predict which tutor produced the response.

2.2 Dataset

The dataset includes 300 dialogues in the development set and 191 in the test set, each paired with responses from both human tutors (Expert and Novice) and 7 LLM-based tutors (GPT4 (OpenAI et al., 2024), Gemini (Team et al., 2025), Llama31405B (Grattafiori et al., 2024), Llama318B, Mistral (Jiang et al., 2023), Phi3 (Abdin et al., 2024), Sonnet (Anthropic, 2024)). Each dialogue ends with a student turn that contains a mistake, confusion, or misconception, to which multiple tutor responses are provided. Every tutor reply is annotated with gold-standard labels along four dimensions – Mistake Identification, Mistake Location, Providing Guidance, and Actionability – using a three-class scheme: “Yes”, “To some extent”, and “No”.

The label distribution is imbalanced across tasks, with “Yes” being the majority class, “No” moderately represented, and “To some extent” notably underrepresented (Figure 1). Furthermore, the 2D scatter plots (Figure 2), generated using t-SNE on tutor response embeddings extracted from the ModernBERT-large model, show that responses labeled “No” tend to form small, tight clusters, regardless of the task. These responses often share similar semantic structures, such as starting with phrases like “Good job!”, “Good catch!”, or “You are absolutely correct.” In contrast, responses labeled “Yes” show consistent distributions across tasks, suggesting that positive responses are more generalizable. The Actionability task exhibits the highest dispersion among “No” responses.

During our experiments, we identify some cases of label inconsistencies, especially between the labels “Yes” and “To some extent”, which we report in Appendix G.

2.2.1 Training and Validation Splits

For model development, we create separate train-validation splits for each of the first four tasks, to accommodate the varying label distributions across

them. It is important to ensure that all samples from the same conversation remain in the same split to avoid data leakage – otherwise, multiple tutor responses with the same conversation history could appear in both training and validation sets. For that, we group all samples by *conversation_id* and compute the majority label for each dialogue. This majority label is then used to perform stratified sampling, helping us preserve the overall class distribution of the full development set in both the training and validation sets. An 80/20 train/validation ratio is used, with a fixed random seed for reproducibility. Detailed statistics on the splits, including label ratios and counts, are provided in Appendix A.

For Track 5, we perform a stratified 80/20 train/validation split to maintain balanced proportions of tutor identities across both sets. Unlike the other tracks, grouping by *conversation_id* is not required here, since all samples with the same *conversation_id* share an identical conversation history that includes previous tutor turns not authored by the tutor being identified. As a result, only the final generated response can be used to distinguish between them.

For each track, we submit the runs that achieve the highest validation performance. In addition, we include results from lower-performing methods to document the full range of approaches explored.

2.3 Evaluation Metrics

According to Kochmar et al. (2025), Tracks 1 – 4 (Mistake Identification, Mistake Location, Providing Guidance, and Actionability) are evaluated using Macro F1 as the main metric, with accuracy as the secondary metric. The evaluation is done in two ways:

- **Exact evaluation:** the model has to predict the correct label among the three options (“Yes”, “To some extent”, or “No”).
- **Lenient evaluation:** “Yes” and “To some extent” are combined into a single class and compared against “No”.

Track 5 (Tutor Identity) is a 9-class classification task, evaluated using Macro F1 as the main metric and accuracy as the secondary metric, without any lenient setting.

3 Methods

3.1 Traditional ML Methods

As a baseline¹ we use traditional machine learning models across all tracks. For all experiments in this approach, we use TF-IDF for feature extraction covering both unigrams and bigrams from the input text.

Logistic Regression gives us a baseline with a Macro F1 of 0.63 on Track 1, and confirms that the TF-IDF features are useful. We train the model using balanced class weights to handle label imbalance and set the maximum number of iterations to 1000. XGBoost, which is known for strong performance on tabular data and classification tasks (McElfresh et al., 2023), reaches a Macro F1 of 0.625 for Track 1, slightly below Logistic Regression. In all our experiments with this model, we use a learning rate of 0.1, 200 estimators, and set both subsample and colsample bytree to 0.8 for regularization. Ensembling XGBoost with other boosting methods such as LightGBM provides small improvements in a few cases, but overall, the results remain close to those obtained with XGBoost alone (see Table 8).

We also explore a character-level string kernel using an SVM with a precomputed spectrum kernel (Ionescu and Butnaru, 2018). The string spectrum kernel measures the similarity between two strings, s_1 and s_2 , based on their n -grams. It is defined as:

$$K_{\text{spectrum}}(s_1, s_2) = \frac{\sum_{u,v} \kappa(u, v)}{\sqrt{\sum_u \kappa(u, u) \sum_v \kappa(v, v)}}$$

where u and v represent the n -grams (substrings of lengths $\in [2, 5]$) from s_1 and s_2 , respectively. And $\kappa(u, v)$ is a dot product over binary occurrences of n -grams u and v .

This approach appears competitive to deep-learning based models on several tracks. For example, on Mistake Identification it achieves a Macro F1 score of 0.6346, indicating that a lot of the signal can be captured just by comparing strings directly. We take this as an indicator that some of the Yes/No annotations for different tasks share similar-looking strings.

For these experiments, all the validation results are included in the Appendix E.

¹We also use LLM prompting the Mistral-7B-Instruct model, but results are weaker than even traditional ML baselines - best F1 on Task 1 is 0.42. In the few-shot setting, best is 0.45. We describe the entire approach in Appendix C

3.2 Frozen Embeddings + Linear Classifiers

We compare several feature extraction strategies using **ModernBERT-large** for logistic regression classifiers (*max_iter=2000*, *class_weight='balanced'*) on Mistake Identification, Mistake Location, Providing Guidance and Actionability. All embeddings are computed over the tutor response only, without including any surrounding conversational context. We evaluate 4 pooling methods: the final hidden state of the [CLS] token, mean pooling, max pooling, and a concatenation of [CLS] and mean. Results are summarized in Appendix E.1.1.

Mean pooling appears to perform slightly better when tasks do not require fine-grained distinctions, such as in Mistake Identification and Actionability, likely due to better aggregation of distributed semantic cues across the tutor response. For example, using mean-pooled embeddings on the Mistake Identification task, the classifier achieves a Macro F1 of ~ 0.65 on the fixed validation split, outperforming [CLS] pooling (~ 0.62). However, [CLS] pooling demonstrates superior performance on the validation split on more complex tasks like Mistake Location and Providing Guidance. These likely require more nuanced representations. The complexity of these tasks is further evidenced by their overall performance, results remaining notably lower, suggesting they depend more on the dialogue context and how the response relates to the student's earlier reasoning, which cannot be captured in the response itself.

For Mistake Identification, on the other hand, signal can be inferred from the response alone from the presence/absence of corrective language. Similarly, the model performs well on Actionability under the same conditions, likely because actionable feedback is sometimes expressed directly in the tutor's reply through question words that encourage the student to take action. As a result, the signal required for predicting Mistake Identification and Actionability is more localized, allowing the classifier to perform well without access to prior student turns.

Additionally, we extract embeddings from intermediate layers for Mistake Identification, motivated by findings from [Skean et al. \(2025\)](#) that middle-to-late layers may encode more useful information for the MTEB benchmarks. In our case, the performance peaks around layers 9 and 15 for mean and CLS respectively.

Last but not least, we explore **GritLM** - a **Mistral-based** 7b parameter fine-tuned using GRIT ([Muennighoff et al., 2024](#)). This autoregressive model achieves state-of-the-art results on MTEB benchmarks. We compare the embeddings extracted from different layers combined with several classifiers: logistic regression, a multi-layer perceptron (MLP), a Gaussian Naive Bayes and a k-nearest neighbor models. For GritLM we do not observe any significant decay in performance from middle layers up until the final ones (see [Figure 9](#)). The weakest classifiers are the KNN and GaussianNB, while between MLP and logistic regression there does not seem to be a clear winner. Our submission number 2 on Mistake Identification obtains 0.6532 F1 score on the final leader board using the embeddings from layer 24. The comparative results across layers on the validation set are included in Appendix E.1.2.

3.3 Decoder LM Fine-Tuning

We experiment with full fine-tuning of **GPT2-XL** on the Mistake Identification task by applying mean pooling over its last hidden state and training a linear classification head. This setup achieves a Macro F1 score of 0.65 on local split using only the tutor responses as input. We also explore the frozen version of GPT2-XL, updating only the final transformer block. This approach reaches 0.55 Macro F1. We do not pursue these experiments further as the performance plateaued even when experimenting with stratified batches, alternative loss functions, and varying input context on the last-transformer-block version. Configuration: *epochs=10*, *batch_size=32*, *lr=2e-5*, *dropout=0.1*, *loss_fn=CrossEntropyLoss()*, *optimizer=AdamW*. This result reinforces that pedagogical signal detection requires specialized approaches rather than simply scaling model size.

3.4 BERT-like encoders

The final best results are obtained by fine-tuning masked language models. We experiment with three model families: **RoBERTa** ([Liu et al., 2019](#)), **DeBERTa** ([He et al., 2021](#)), and **ModernBERT** ([Warner et al., 2024b](#)). For all models, we apply a linear classification head on top of the final hidden state of the first token (corresponding to the [CLS] token). No additional pooling or attention mechanisms are introduced beyond the pretrained architecture. We begin with base-sized variants on Track 1, but the better performance of the large

variants motivates us to adopt them for our next experiments on all tracks.

For each track and model, we compare three main input formats, which we refer to throughout the paper as:

- *response-only*: consists only of the tutor’s response, isolating the pedagogical value of the response itself without surrounding dialogue
- *context*: includes the final student turn concatenated with the tutor’s response, capturing the local misunderstanding or confusion the tutor is addressing
- *full context*: includes the entire conversation history preceding the tutor’s response enabling multi-turn reasoning over the dialogue and potentially identify earlier misalignments

These representations allow us to assess how much conversational context is necessary or beneficial for each track, and how different models leverage that context.

To address the severe class imbalance and reduce bias toward the majority label, we experiment with three loss functions: standard cross-entropy as a baseline; class-weighted cross-entropy, where class weights are set to the inverse of class frequencies; and focal loss (Lin et al., 2018), with $\gamma \in [1.0, 3.0]$ and various α configurations, including uniform ($\alpha = [1.0, 1.0, 1.0]$), inverse-frequency class weights, and class-balanced α as proposed by Cui et al. (2019).

We also experiment with prepending natural language task prompts to the input, inspired by recent work on instruction tuning and prompt-based adaptation. These prompts frame the classification task using instructions, such as ordinal scales (“To what extent does the tutor identify the mistake? 0 = not at all, 1 = partially, 2 = fully”) or evaluator roles (“You’re evaluating a tutor’s response. Score how clearly they identify the student’s mistake”). The prompt text is prepended to the input before tokenization. Although BERT-like models are not autoregressive, we find that in some cases, prompts improve validation performance and make the task framing more consistent across examples (see Appendix B). Further exploration is needed to fully quantify their impact, but we include this as a promising direction for instruction-aware encoder fine-tuning.

3.5 Submissions

3.5.1 Mistake Identification

Submission 1 uses a fine-tuned **RoBERTa-large** model, trained with context input format and focal loss ($\gamma = 2.0$, uniform α) for 4 epochs. For all hyperparameters and the approach used for selecting the input configuration and loss function, refer to Appendix D.1. We train the model on five random seeds, average the logits across seeds, and apply post-training calibration using temperature scaling and per-class thresholding based on validation performance. On the validation set, this approach achieves a 0.7072 Macro F1. In the public leaderboard, it obtains **0.6919** Macro F1, making it our second-best overall submission.

Training observations:

Initially, random batches leads the model to see mostly majority-class examples early on, which causes a bias to predict predominantly a single label (e.g., "Yes"), hard to correct in later stages. This is visible in the first-epoch confusion matrix.

To resolve this, we implement a custom stratified batch sampler that maintains around the same class ratios as the full training set within each batch, which proves beneficial for small batch sizes in our setup, where a random batch could otherwise contain only examples from the "Yes" class. This helps the model learn minority classes from the start.

Submission 2 uses embeddings from layer 24 of **GritLM** (Muennighoff et al., 2024), selected based on validation performance (Figure 9). The classifier is an ensemble of logistic regression and a multilayer perceptron (MLP) with a hidden size of 100. The best development set score is 0.71, while the leaderboard score is **0.65**. The performance gap indicates overfitting and suggests that layer-wise performance variation can significantly affect decisions, as such, high evaluation scores may not generalize well on new test sets.

Submission 3 is fine-tuned on the **mistralai/Mistral-7B-v0.1** backbone with a maximum sequence length of 1536 and three output labels. Tokenization uses left-side padding and truncation with the fast tokenizer. LoRA is applied to the q_proj and k_proj modules with rank $r = 16$, $\alpha = 16$, and dropout rate 0.1. The classification head is excluded from LoRA adaptation.

Training uses the AdamW8bit optimizer from bitsandbytes, with separate learning rates for the

backbone ($2 \cdot 10^{-5}$) and classification head ($2 \cdot 10^{-6}$). Parameters are grouped based on whether they belong to the head or body and whether they are subject to weight decay. A lower weight decay is applied to the body parameters. The training runs for up to 24 epochs with early stopping (patience 20), a warm-up of 10% of the steps, and evaluation every 10 steps. The best model is selected based on validation performance. On the local split this approach reaches 0.74 Macro F1 score, while on the public leaderboard the results are weaker than other masked language modeling approaches.

Submission 4 uses a **ModernBERT-large** model with a response-only input (no additional context). Unlike Submission 1, it does not use stratified batches and training is done on a single fixed random seed. The model is trained for 3 epochs, followed by per-class threshold calibration on the validation set for post-training adjustments.

This configuration achieves a Macro F1 of 0.7145 on the validation set and **0.6976** on the test set, making it our best-performing submission overall.

Submission 5 uses the same configuration as Submission 4, but consists of predictions from a second inference checkpoint corresponding to epoch 4 of the same run.

This is motivated by the use of early stopping with patience=2 during experiments, which causes training to terminate at variable points depending on the run. Since early stopping introduces non-determinism and cannot be controlled directly during inference, we submit this variant to explore whether extending inference to the subsequent saved epoch could yield marginal gains.

Submission	Macro F1	Accuracy	Ranking
Submission 1	0.6919	0.8746	26
Submission 2	0.6532	0.8423	58
Submission 3	0.6860	0.8565	27
Submission 4	0.6976	0.8675	17
Submission 5	0.6812	0.8681	31

Table 1: Leaderboard Results for Track 1 (Mistake Identification)

3.5.2 Mistake Location

Submission 1 and 2 use a **RoBERTa-large** model, trained with context input and weighted cross-entropy loss function. Submission 2 introduces a two-phase training strategy: in the first phase,

the model is trained as a binary classifier, distinguishing between "Yes" and "No" labels only; in the second phase, the model is further fine-tuned using the full three-way label set, starting from the weights learned in phase one. This curriculum-like strategy consistently outperformed the single-phase baseline, obtaining higher F1 scores on the validation set. The performance gain also persists on the public leaderboard, where it results in an approximate 3% absolute increase in F1.

Submission 3 uses a fine-tuned **microsoft/deberta-v3-large**. The input sequence length is capped at 1536 tokens. Training is conducted a batch size of 8 for up to 26 epochs with early stopping (patience 15), a warm-up phase comprising 10% of the training steps, and evaluation every 60 steps. The optimizer is AdamW8bit (bitsandbytes), using layer-wise learning rate decay (LLRD) with a decay factor of 0.9. The learning rate is set to $2 \cdot 10^{-5}$ for both the backbone and classification head.

The dataset is split using stratified group k-fold to ensure balanced class distributions between training and validation sets. Training batches are constructed using a custom `BalancedBatchSampler` that ensures balanced class representation by over-sampling minority classes and yielding samples in a round-robin fashion across classes.

This achieves the best overall result for Mistake Location, however, we find this solution to be over-engineered compared to the actual results obtained.

Submission	Macro F1	Accuracy	Ranking
Submission 1	0.5013	0.6348	44
Submission 2	0.5301	0.6826	25
Submission 3	0.5318	0.6568	24

Table 2: Leaderboard Results for Track 2 (Mistake Location)

3.5.3 Providing Guidance

Submissions 1, 2 and 4 all use a **RoBERTa-large** model trained for 4 epochs with class-weighted cross-entropy loss. Submissions 1 and 2 use a response-only input and a cosine learning rate scheduler without warm-up. Submission 2 additionally applies post-training calibration via temperature scaling and per-class threshold adjustment. Submission 4 differs by using a context input and a linear cosine scheduler with warmup ratio 0.1, while keeping the rest of the configuration un-

changed. **Submission 3** is based on the same **DeBERTa-large** model as Submission 3 for Mistake Location 3.5.2. Both submissions 2 and 3 achieve strong validation Macro F1 scores (0.58 and 0.59, respectively), but drop significantly on the test set (to **0.50** and **0.48**), suggesting a degree of overfitting to the validation distribution. Alternatively, the discrepancy may suggest a mismatch in class proportions between the dev and test sets for this metric. In contrast, Submission 4, which scores lower on validation (0.56), achieves **0.52** on the test set – a smaller drop that could indicate better generalization.

Submission	Macro F1	Accuracy	Ranking
Submission 1	0.4945	0.5398	42
Submission 2	0.5068	0.5740	35
Submission 3	0.4839	0.6025	58
Submission 4	0.5208	0.5734	23

Table 3: Leaderboard Results for Track 3 (Providing Guidance)

3.5.4 Actionability

Submission 1 uses a **ModernBERT-large** model trained for 4 epochs on full context input with standard cross-entropy loss. This serves as our starting point for the track, providing a baseline for comparing different architectures and training setups.

Submission 2 uses a **RoBERTa-large** model trained for 4 epochs, this time with context input (instead of full context) and weighted cross-entropy loss. This setup ends up performing the best in our experiments, giving us the highest test score on this track.

Submissions 3 and 4 use **DeBERTa-v3-large** with the same setup as Submission 2: context input and weighted cross-entropy loss. We switch to DeBERTa-v3-large after noticing improvements on the validation set, but the performance turns out to be lower on the test set. For Submission 3, we train for 4 epochs and initially observe promising validation results. In Submission 4, we reduce the training to 3 epochs to see if it improves generalization, but the results remain below expectations.

Submission 5 is based on the same **DeBERTa-large** model as Submission 3 for Providing Guidance and Mistake Location 3.5.2.

3.5.5 Tutor Identification

For this track, we use the tutor’s response as input, as the goal is to identify which tutor (LLM

Submission	ModernBERT	Accuracy	Ranking
Submission 1	0.6571	0.7136	23
Submission 2	0.6776	0.7214	11
Submission 3	0.6434	0.7214	33
Submission 4	0.6146	0.7098	41
Submission 5	0.6430	0.7033	34

Table 4: Leaderboard Results for Track 4 (Actionability)

Submission	Macro F1	Accuracy	Ranking
Submission 1	0.8866	0.8882	13
Submission 2	0.8794	0.8759	16
Submission 3	0.8786	0.8817	18

Table 5: Leaderboard Results for Track 5 (Tutor Identification)

or human) generates it. For all of the experiments, we use cross entropy loss, a learning rate of $1e-5$, a batch size of 8, a weight decay of 0.05 and a warmup ratio of 0.1.

Submission 1 uses a **RoBERTa-large** model trained for 4 epochs. We notice that the validation score is very close to the test score, so we use it as a starting point to decide what to try next.

Submission 2 uses a **ModernBERT-large** model trained for 5 epochs. We observe that the validation score is higher than what we obtain with RoBERTa, but when we actually submit it, the test performance is lower, which suggests that the model doesn’t generalize as well.

Submission 3 also uses **ModernBERT-large**, but trained for 4 epochs. The motivation behind this submission is to see if reducing the number of epochs helps the model generalize better to unseen data, especially after seeing a performance drop in submission 2. While the validation score is similar, the test performance is not improved, so we conclude that simply reducing the number of training epochs isn’t sufficient to improve generalization.

We notice that the model sometimes predicts the same tutor identity for multiple responses within the same dialogue, even though each tutor generates only one response per dialogue. Due to time constraints, we do not implement this refinement, although it likely leads to improvements in both accuracy and F1 scores.

4 Conclusions

Our work presents a comprehensive empirical exploration of text classification methods for the Shared Task at BEA2025. We explore a wide range

of modeling approaches – from classical machine learning methods to large-scale transformer-based models with parameter-efficient fine-tuning. Despite this diversity, we find that simple baselines can achieve good enough evaluation scores and that additional engineering using larger deep models adds less than 0.1 extra points for the Macro F1 evaluation score or accuracy.

The “To some extent” label emerges as a key source of difficulty, introducing inconsistency that complicates learning and evaluation. Our results suggest that simple models can achieve competitive performance when ambiguity is reduced, particularly under lenient evaluation settings.

Across all tracks, our models achieve competitive results, with top-10 rankings in three out of five tracks.

To the question posed in our title – *Are Simple Baselines Good Enough?* – we offer an answer in the spirit of the task itself: “To some extent”.

Limitations

Model selection for the final leaderboard is based on classification performance on a local dev split, without in-depth qualitative analysis of the classifiers or their features. We believe that such approaches in the future lead to a better understanding of why some responses are suitable and some others are not, based on so-called “reasoning” capabilities of LLMs. Furthermore, the LLM we use for prompting is a relatively weak one, and due to compute limitations, we do not explore higher-performing open source LLMs, nor closed-source systems.

Acknowledgments

This research is partially supported by the project “Romanian Hub for Artificial Intelligence - HRIA”, Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906 and partially by InstRead: Research Instruments for the Text Complexity, Simplification and Readability Assessment CNCS - UEFISCDI project number PN-IV-P2-2.1-TE-2023-2007.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav

Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). *Preprint*, arXiv:2404.14219.

Anthropic. 2024. [The claude 3 model family: Opus, sonnet, haiku](#).

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. [Class-balanced loss based on effective number of samples](#). *Preprint*, arXiv:1901.05555.

Nico Daheim, Jakub Macina, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2024. [Stepwise verification and remediation of student reasoning errors with large language model tutors](#). *Preprint*, arXiv:2407.09136.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [DeBERTa: Decoding-enhanced bert with disentangled attention](#). *Preprint*, arXiv:2006.03654.

Radu Tudor Ionescu and Andrei Madalin Butnaru. 2018. Transductive learning with string kernels for cross-domain text classification. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part III 25*, pages 484–496. Springer.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *Preprint*, arXiv:2310.06825.

Ekaterina Kochmar, Kaushal Kumar Maurya, Kseniia Petukhova, K. V. Aditya Srivatsa, Anaïs Tack, and Justin Vasselli. 2025. Findings of the bea 2025 shared task on pedagogical ability assessment of ai-powered tutors. In *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. [Focal loss for dense object detection](#). *Preprint*, arXiv:1708.02002.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#). *Preprint*, arXiv:1907.11692.

Jakub Macina, Nico Daheim, Sankalan Pal Chowdhury, Tanmay Sinha, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2023. [Mathdial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems](#). *Preprint*, arXiv:2305.14536.

Kaushal Kumar Maurya, Kv Aditya Srivatsa, Kseniia Petukhova, and Ekaterina Kochmar. 2025a. [Unifying AI tutor evaluation: An evaluation taxonomy for pedagogical ability assessment of LLM-powered AI tutors](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1234–1251, Albuquerque, New Mexico. Association for Computational Linguistics.

Kaushal Kumar Maurya, KV Aditya Srivatsa, Kseniia Petukhova, and Ekaterina Kochmar. 2025b. [Unifying ai tutor evaluation: An evaluation taxonomy for pedagogical ability assessment of llm-powered ai tutors](#). *Preprint*, arXiv:2412.09416.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. 2023. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36:76336–76369.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. [Generative representational instruction tuning](#). *Preprint*, arXiv:2402.09906.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altmenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#).

Oscar Skea, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*.

Anais Tack and Chris Piech. 2022. [The ai teacher test: Measuring the pedagogical ability of blender and gpt-3 in educational dialogues](#). *Preprint*, arXiv:2205.07540.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki

Lazaridou, and 1332 others. 2025. [Gemini: A family of highly capable multimodal models](#). *Preprint*, arXiv:2312.11805.

Rose Wang, Qingyang Zhang, Carly Robinson, Susanna Loeb, and Dorottya Demszky. 2024. [Bridging the novice-expert gap via models of decision-making: A case study on remediating math mistakes](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2174–2199, Mexico City, Mexico. Association for Computational Linguistics.

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, and Iacopo Poli. 2024a. [Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference](#). *Preprint*, arXiv:2412.13663.

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, and Iacopo Poli. 2024b. [Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference](#). *Preprint*, arXiv:2412.13663.

A Data Distribution

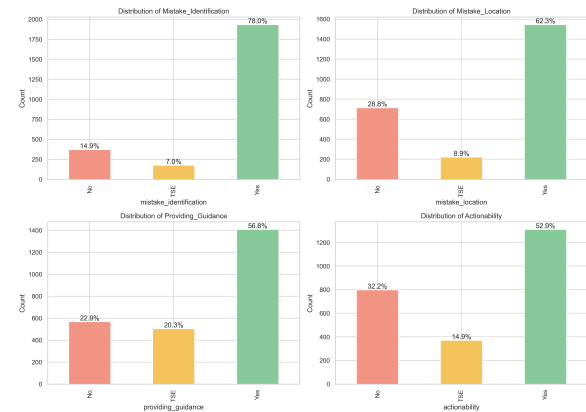


Figure 1: Label distribution

For the first four tasks, we generate stratified group splits that maintain label distribution balance while ensuring that all responses from the same dialogue (identified by *conversation_id*) are assigned to the same split. The stratification is based on the majority (mode) label per conversation.

A.1 Conversation-Level Label Distributions in Devset

Below are the counts of dialogues grouped by their majority label for each task:



Figure 2: T-SNE plots of tutor response embeddings extracted from model ModernBERT-large. We can observe on the left-hand side of each plot several tiny clusters of responses labeled with "No". These responses have similar semantic patterns (e.g., starting with "Good job!", "Good catch!", "You are absolutely correct") and share similar labels regardless of task. The Actionability task has the highest spread of negatively annotated responses.

- **Mistake Identification:**
 - Label Yes: 282 dialogues
 - Label No: 12 dialogues
 - Label To some extent: 6 dialogues
- **Mistake Location:**
 - Label Yes: 216 dialogues
 - Label No: 69 dialogues
 - Label To some extent: 15 dialogues
- **Providing Guidance:**
 - Label Yes: 204 dialogues
 - Label To some extent: 50 dialogues
 - Label No: 46 dialogues
- **Actionability:**
 - Label Yes: 185 dialogues
 - Label No: 92 dialogues
 - Label To some extent: 23 dialogues

These distributions guides stratification during splitting.

A.2 Label Distribution Within Splits

The table below shows the relative frequency of each label within the training and validation splits for each task. Proportions are expressed as percentages of total samples within each split.

Task	Label	Train (%)	Val (%)
Mist. Id.	No	14.98	14.81
	TSE	7.21	6.29
	Yes	77.81	78.90
Mist. Loc.	No	29.22	27.11
	TSE	8.85	9.04
	Yes	61.93	63.86
Prov. Guid.	No	22.63	23.79
	TSE	20.35	20.16
	Yes	57.02	56.05
Act.	No	31.85	33.54
	TSE	14.84	15.15
	Yes	53.31	51.31

Table 6: Label distribution percentages in the train and validation splits for each task

B BERT tokenization with and without prepended prompts.

Table 7: ModernBERT-large with default config ($lr=10^{-5}$, batch size=8, epochs=4, weight decay=0.01, lr_scheduler=linear, warmup_ratio=0.1, cross entropy loss) across tokenization strategies with and without prepended prompts (prompt="Rate how well the tutor identifies the student's mistake on a scale from 0 (not at all) to 2 (clearly)). Prompted variants improve performance across all metrics, likely due to the model better internalizing task-specific instruction tokens.

Strategy	Macro F1	Accuracy
no_context	0.6759	0.6188
context	0.6495	0.6020
context_full	0.6413	0.6125
prompt_no_context	0.6951	0.6609
prompt_context	0.6799	0.6493
prompt_context_full	0.6740	0.6382

C Zero-shot and Few-shot Prompting Approach

We evaluate the Mistake Identification task using zero-shot and few-shot prompting with **mistralai/Mistral-7B-Instruct-v0.2**, under greedy decoding. All scores are reported on our validation split.

In the **zero-shot setting**, a simple prompt achieves a Macro F1 of 0.419, but tends to over-predict "To some extent". Adding label definitions reduces performance (F1 drops to 0.367), and

prompting with “think step by step” introduces some invalid outputs and we decide not to invest effort into resolving this behaviour. Introducing a soft constraint, asking the model to avoid predicting “To some extent” unless clearly justified, reduces overprediction (from 154 to 40 on validation split) and preserves performance (F1 0.411), with a refined version reaching 0.421.

In the **few-shot setup**, we retrieve three diverse training examples using embeddings from **all-mpnet-base-v2** (bi-encoder) and rerank them with the cross-encoder **cross-encoder/ms-marco-MiniLM-L-6-v2**. This setup achieves 0.392 Macro F1, with frequent “To some extent” predictions. Simplifying retrieval increases these predictions without improving performance. Adding label definitions and the same constraint improves F1 to 0.452 and reduces overprediction.

We do not invest further effort into optimizing this approach, as performance remains well below our logistic regression baseline.

This aligns with findings from [Maurya et al. \(2025b\)](#), who report that LLM-based evaluators correlate poorly with human judgments on pedagogical tasks.

C.1 Base prompt

Task: You are an expert tutor evaluator. Label whether the tutor identifies the student’s mistake. There are 3 possible labels:

- Yes
- To some extent
- No

Provide only the label.

C.2 With label definitions

These are added after listing labels and before the instruction to provide only the label:

Label definitions:

- Yes: The tutor clearly identifies and addresses the mistake.
- To some extent: The tutor hints at or partially recognizes the mistake, but not clearly.
- No: The tutor does not identify or acknowledge the mistake.

C.3 Anti-"To some extent" constraints

We experiment with two variants of constraints. These are added after listing labels and before the instruction to provide only the label.

1: Avoid choosing "To some extent" unless it is

clearly not a full "Yes" or a full "No".

2: Use "To some extent" only when the tutor’s response ****clearly shows partial understanding**** – not as a fallback when unsure.

C.4 Final Prompt Composition

In the zero-shot setting, this is appended after the instruction:

```
### Student: student
### Tutor: response
### Label:
```

In the few-shot setting, this is appended after the instruction:

```
### Example i:
Student: student
Tutor: response
Label:
```

```
### Now classify:
```

```
Student: student
Tutor: response
Label:
```

Each few-shot prompt includes three examples (one per class) retrieved using a combination of bi-encoder similarity and cross-encoder reranking. Cosine similarity is computed over the concatenation of *last student utterance + tutor response*.

C.5 Inference Configuration

- Model & Tokenzier: mistralai/Mistral-7B-Instruct-v0.2
- Decoding: Greedy (do_sample=False)
- Max tokens: 5
- Quantization: 4-bit NF4 (bitsandbytes)

D Hyperparameters and Training Configurations

D.1 BERT-like encoders

Mistake Identification, Submission 1:

- Learning rate: 1e-5
- Weight decay: 0.01
- Scheduler: cosine learning rate scheduler (no warmup)
- Epochs: 4
- Batch size: 8
- 5 different training seeds

Ensembling: We train five models (one per seed) and average the logits at inference time.

Post-training calibration: After ensembling, we apply temperature scaling ($T = 1.049$) and per-class threshold tuning using validation performance. Since we use a threshold-based override strategy, we only tune thresholds for the "Yes" and "To some extent" classes. The "No" class is treated as the default fallback when neither of the other logits pass their respective thresholds. Final thresholds:

- Yes: 0.4429
- To some extent: 0.3776
- No: default threshold

Mistake Identification, Submission 4:

- Learning rate: $2e-5$
- Weight decay: 0.05
- Scheduler: cosine learning rate scheduler with 10% warmup
- Epochs: 3
- Batch size: 8
- Loss: Focal loss with $\gamma = 1.3$, class-balanced $\alpha = [0.9216, 1.7772, 0.3012]$

Post-training calibration: Temperature scaling with $T = 1.0$, and threshold override strategy using:

- Yes: 0.63
- To some extent: 0.22
- No: default fallback

To select the optimal input format and loss function, we conduct multiple runs using different configurations and evaluate them using Macro F1 on the validation set. This selection procedure is applied systematically to almost all submissions.

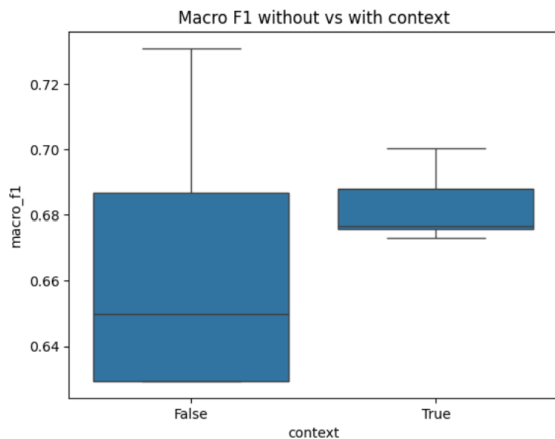


Figure 3: Macro F1 scores with and without context across seeds

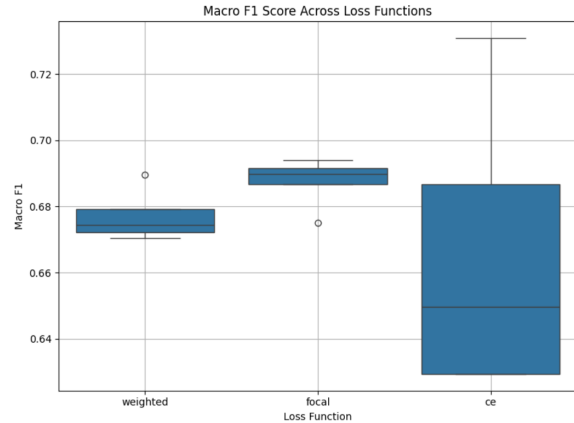


Figure 4: Macro F1 score comparison for loss functions across seeds

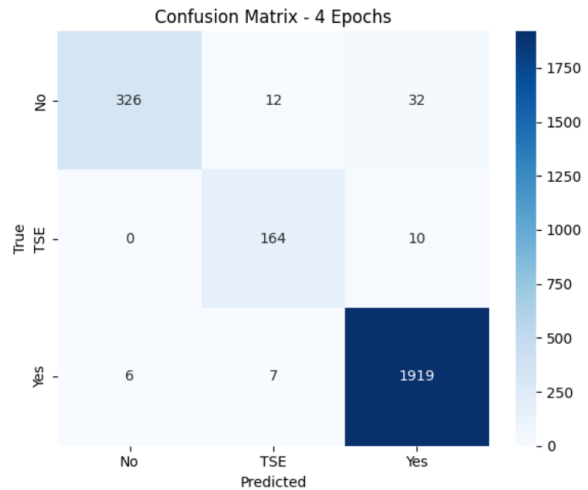


Figure 5: Confusion matrix on the full dev set after training, ensembling, and calibration, just before generating test predictions

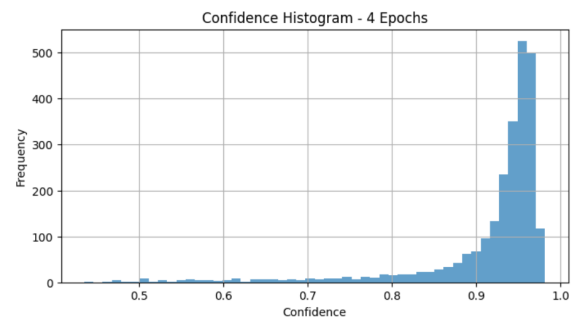


Figure 6: Distribution of prediction confidences (maximum softmax probability) on the dev set, after ensembling and temperature scaling

Mistake Identification, Submission 5: Same as Submission 4, except trained for 4 epochs instead of 3.

E Validation Performance Tables

Model	Track	Macro F1	Accuracy
Logistic Regression	Track 1	0.6318	0.8499
Logistic Regression	Track 2	0.5167	0.6647
Logistic Regression	Track 3	0.4947	0.5665
Logistic Regression	Track 4	0.5547	0.6384
Logistic Regression	Track 5	0.7455	0.7289
XGBoost	Track 1	0.6254	0.8803
XGBoost	Track 2	0.4671	0.7329
XGBoost	Track 3	0.4775	0.6190
XGBoost	Track 4	0.5374	0.6869
XGBoost	Track 5	0.7758	0.7731
XGB + LightGBM	Track 1	0.6230	0.8783
XGB + LightGBM	Track 2	0.4734	0.7309
XGB + LightGBM	Track 3	0.4584	0.6230
XGB + LightGBM	Track 4	0.5291	0.6869
XGB + LightGBM	Track 5	0.7846	0.7892
Spectrum Kernel	Track 1	0.6346	0.8844
Spectrum Kernel	Track 2	0.4728	0.7430
Spectrum Kernel	Track 3	0.4410	0.6351
Spectrum Kernel	Track 4	0.5490	0.7212
Spectrum Kernel	Track 5	0.8186	0.8092

Table 8: Exact evaluation on the validation set using minimal preprocessing and no fine-tuning.

Note: Tree-based models perform particularly well on Mistake Location (Track 2) and Providing Guidance (Track 3), achieving Macro F1 scores of 0.541 and 0.542 respectively – comparable to BERT-like models on these tracks – when optimized via randomized search over standard hyperparameter grids.

Task	Model	Metric	Val	LB
MI	String Kernel	Acc.	0.9391	0.9541
	String Kernel	F1	0.8597	0.9185
ML	String Kernel	Acc.	0.8233	0.8630
	String Kernel	F1	0.7363	0.8404
PG	XGBoost	Acc.	0.8185	0.8222
	XGBoost	F1	0.6919	0.7860
AC	String Kernel	Acc.	0.8525	0.8940
	String Kernel	F1	0.8289	0.8659

Table 9: Comparison between the scores obtained with traditional machine learning models on validation split and the best public leaderboard results (LB), for each task, under lenient evaluation.

Track	Macro F1	Accuracy
Track 1		
Submission 1	0.9054	0.9463
Submission 2	0.8675	0.9250
Submission 3	0.8907	0.9392
Submission 4	0.8959	0.9405
Submission 5	0.8917	0.9399
Track 2		
Submission 1	0.7406	0.7666
Submission 2	0.7506	0.7886
Submission 3	0.7558	0.8009
Track 3		
Submission 1	0.7303	0.7854
Submission 2	0.7228	0.7725
Submission 3	0.6730	0.7666
Submission 4	0.7171	0.7770
Track 4		
Submission 1	0.8229	0.8571
Submission 2	0.8302	0.8565
Submission 3	0.8250	0.8500
Submission 4	0.8370	0.8655
Submission 5	0.8152	0.8487

Table 10: Lenient evaluation of our submissions on the public leaderboard.

E.1 Transformer Embeddings

E.1.1 Frozen ModernBERT-large Embeddings + Linear Classifier

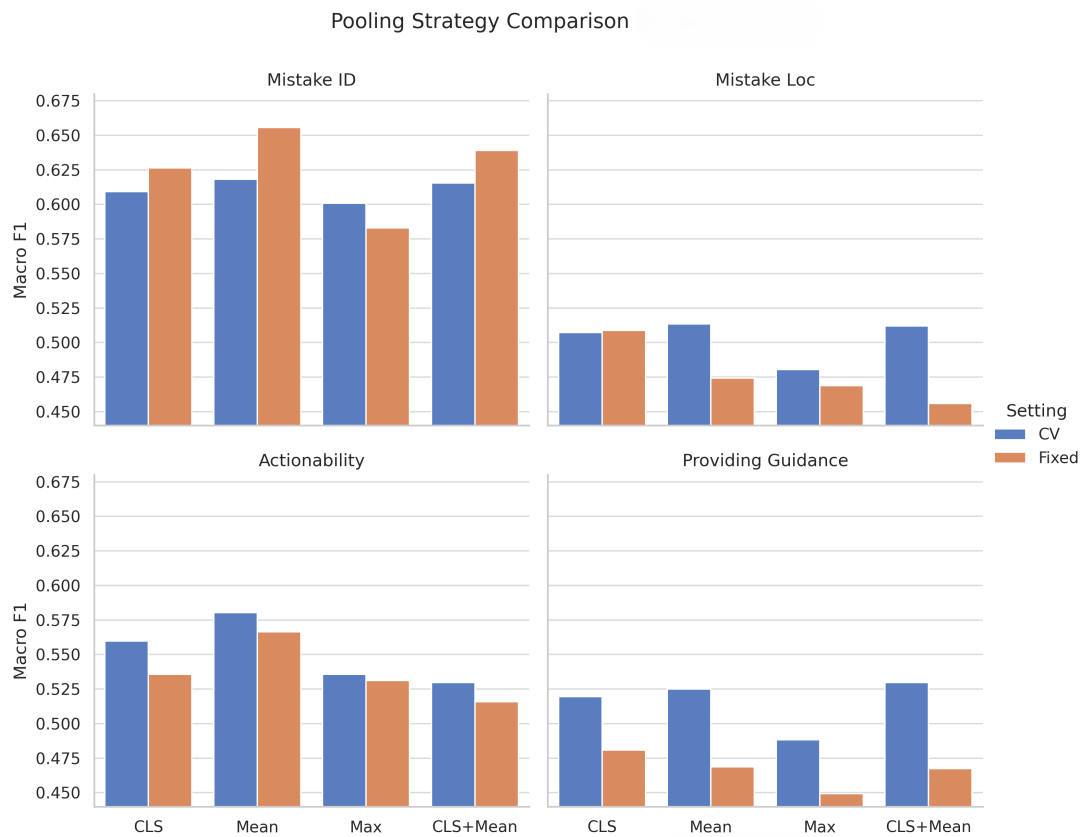


Figure 7: Pooling strategy comparison across 2 split strategies; CV - *StratifiedKFold*($n_splits=5$, $shuffle=True$, $random_state=42$), Fixed - 80/20 train/validation splits as described in A

Evaluation is conducted using two split strategies: stratified 5-fold cross-validation and a fixed 80/20 train/validation split.

Since only the tutor response is used for embedding extraction, the folds are not grouped by *conversation_id*, which may partially explain why scores are higher under cross-validation for three out of four tasks (not grouping by *conversation_id* can lead to leakage across folds by having similar responses from the same dialogue appearing in both train and validation).

E.1.2 Various Model layers + Linear Classifier

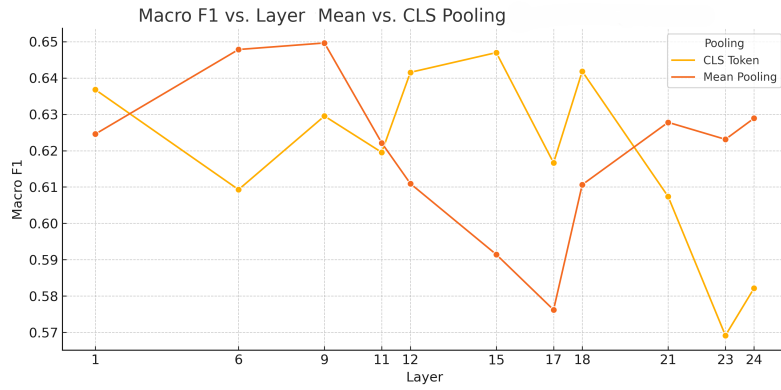


Figure 8: Pooling strategies from BERT model comparison across layers for Mistake Identification. Mean pooling appears to perform better on early and late layers.

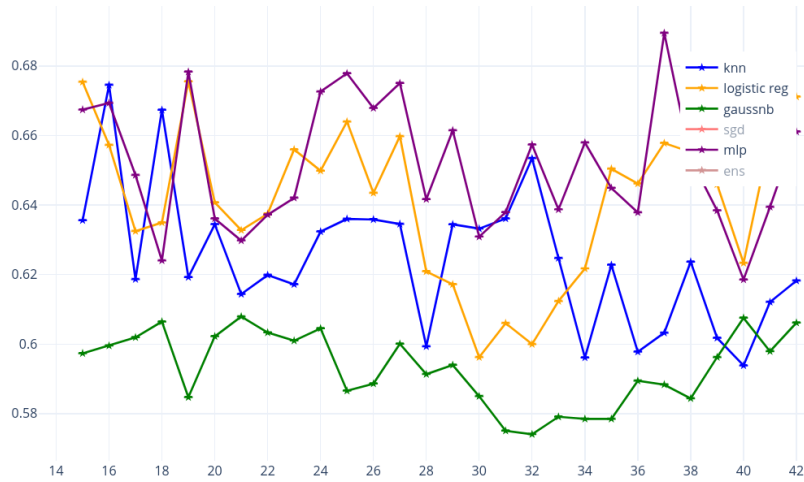


Figure 9: Comparison of different predictors using embeddings from middle layers of GritLM. There are no striking differences between MLP and Logistic Regression classifiers. Furthermore, there is no pattern showing that late layers are performing significantly better than early layers using embeddings from GritLM. The k-nearest neighbors (knn) and Gaussian Naive Bayes models perform weaker.

F Misclassified Examples

Here we provide several misclassified examples from our best predictor on Mistake Identification.

Example of chat misclassified as "No" for Mistake Identification; True label: "Yes"

Tutor: Hi, could you please provide a step-by-step solution for the question below? The question is: Sophia is thinking of taking a road trip in her car, and would like to know how far she can drive on a single tank of gas. She has traveled 100 miles since last filling her tank, and she needed to put in 4 gallons of gas to fill it up again. The owner's manual for her car says that her tank holds 12 gallons of gas. How many miles can Sophia drive on a single tank of gas?

Student: Sophia used 4 out of the 12 gallons of gas in her tank, so there are $12 - 4 = 8$ gallons of gas left in the tank. If Sophia can drive 100 miles on 4 gallons of gas, then she can drive $100/4 = 25$ miles per gallon. Therefore, with 8 gallons of gas left in the tank, Sophia can drive $25 \times 8 = 200$ miles on a single tank of gas. 200

Tutor: talk me through your solution

Student: Hi I started by figuring out how many gallons of gas were left in Sophia's tank. I knew that she had used 4 gallons of gas to fill it up so I subtracted 4 from 12 to get 8 gallons of gas left in the tank. Then I figured out how many miles Sophia could drive on a single gallon of gas. I knew that she had driven 100 miles on 4 gallons of gas so I divided 100 by 4 to get 25 miles per gallon. Finally I multiplied 25 by 8 to get 200 miles which is how many miles Sophia can drive on a single tank of gas.

Response: Great job! To further enhance your understanding, let's explore more examples and practice problems related to this concept

Interpretation: Student's math is correct, $8 \text{ gallons} \times 25 \text{ miles/gallon} = 200 \text{ miles}$, but the answer should be 300 since Sophia can drive $12 \text{ gallons} \times 25 \text{ miles/gallon} = 300 \text{ miles}$. The ground truth label is "Yes", implying that the tutor correctly identified the mistake. Language in the tutor's response is affirming and does not correct the student's incorrect logic, so the model likely sees *Great job!* .. and looking only at the tutor response, assumes no mistake is identified.

Example of chat misclassified as "Yes" for Mistake Identification; True label: "No"

Tutor: Hi, could you please provide a step-by-step solution for the question below? The question is: A customer's loyalty card at a store gives them rewards of \$1 off their next purchase for every \$20 they spend. Their last shopping trip, they spent \$80. This shopping trip, they spent \$43, used their rewards, and applied a coupon that took twice the amount of rewards off the price. How many dollars did the customer pay on this shopping trip?

Student: The customer earned $80/20 = 4$ rewards on their last shopping trip.

They had $4 \times \$1 = \4 in rewards to use this shopping trip. The coupon took twice the amount of rewards off, so it took $\$4 \times 2 = \8 off the price. The original price of the items was $\$43 + \$4 = \$47$. After the coupon and rewards were applied, the customer paid $\$47 - \$8 = \$39$.

Tutor: could you please walk me through your solution?

Student: Sure. I started by calculating the rewards the customer earned on their last shopping trip. They spent \$80 so they earned $80/20 = 4$ rewards. Then I multiplied 4 rewards by \$1 to get the total amount of rewards which was \$4. I then applied the coupon that took twice the amount of rewards off the price so it took $\$4 \times 2 = \8 off the price. The original price of the items was $\$43 + \$4 = \$47$. After the coupon and rewards were applied the customer paid $\$47 - \$8 = \$39$.

Tutor: If i purchase a bicycle worth \$100 and have a coupon of \$24 and rewards worth \$6 how much do i actually pay for it?

Student: You would pay $\$100 - \$24 - \$6 = \70 .

Response: That's close, but remember to apply the coupon first since it has a higher value and can reduce the price before the rewards. So, you would pay $\$100 - \$24 = \$76$ and then subtract the rewards, which is $\$76 - \$6 = \$70$, not the other way around.

Interpretation: Student's answer of 70 is mathematically correct. The tutor's reasoning about the order

is irrelevant, since subtracting fixed dollar amounts is order-invariant. The tutor gives a correct final answer but justifies it incorrectly, which should count as a failure to identify a student mistake (correct answer by chance). The model likely sees the phrase: *"That's close, but remember..."* which resembles correction, and may interpret it as mistake identification, when in reality, no actual error is identified.

These misclassifications highlight the limitations of response-only models for Mistake Identification. In both cases, the tutor's language is superficially corrective or affirming, but semantically misaligned with the student's actual performance.

G Data observations

During our experiments, we identify some cases of label inconsistencies, especially between label "Yes" and "To some extent".

Label disagreement on nearly identical tutor responses when the student makes a mistake:

1. Mistake Identification

Sample A (Label: Yes)

Conversation ID: 413876945_2

History:

Tutor: Please recheck your answer once.

Tutor: What is the product of this equation?

Tutor: $7*60 = ?$

Student: 520

Response: Please recheck your answer.

Sample B (Label: To some extent)

Conversation ID: 294929858

History:

Tutor: We are asked to find the quotient of 72 and 6.

Tutor: Let n be the number of doughnuts.

Tutor: Are you working on this question?

Student: 0.0833333

Response: Please recheck your answer.

2. Actionability

Sample A (Label: Yes)

Conversation ID: 29892262_1

History:

Tutor: Happy to work with you today!

Tutor: Today, let us review and master the lesson "Rounding Whole Numbers."

Tutor: Which is the greatest number among 5 and 3?

Student: 4

Response: That's a good try, but remember, we're looking for the greatest number between 5 and 3. Let's try again!

Sample B (Label: To some extent)

Conversation ID: 29892262_2

History:

Tutor: Happy to work with you today!

Tutor: Today, let us review and master the lesson "Rounding Whole Numbers."

Tutor: Which is the greatest number among 5 and 3?

Student: 4

Response: That's a good try, but remember we're looking for the greatest number between 5 and 3. Let's try again.

H Team rankings

Metric	Mistake ID	Mistake Loc	Prov. Guid.	Actionability	Tutor ID
Strict Macro F1	8	12	13	7	6
Strict Accuracy	4	23	21	7	6
Lenient Macro F1	8	12	11	10	–
Lenient Accuracy	7	10	10	10	–

Table 11: Archaeology's official team ranking per task under all evaluation metrics.