Team CNLP-NITS-PP at PAN: Advancing Generative AI Detection: Mixture of Experts with Transformer Models

Notebook for PAN at CLEF 2025

First Author[1], Second Author[2]

[1]*Affiliation, Address, City, Country*
[2]*Affiliation, Address, City, Country*

Abstract
Generative Artificial Intelligence (Gen AI) texts are evolving globally, from mundane to significant matters. We humans tend not to know that the texts are written by them, but not by an AI, so we do things like adding our content to the original generated AI texts. This works proposes a new method for the classification of potentially obfuscated text and for the classification of a document collaboratively authored by humans and AI. This work is a part of PAN at CLEF 2025 shared task named Voight-Kampff Generative AI Detection. Our new method explores the integration of Mixture-of-Experts (MoE) architecture with transformer-based language models for text classification. This work involves two tasks: Voight-Kampff AI Detection Sensitivity and Human-AI Collaborative Text Classification. The SoftMoE employs a gating mechanism to dynamically combine expert outputs, while the HardMoE selects a single expert per input. Our experiments tell us that MoE-enhanced models achieve competitive performance.

Keywords
PAN 2025, Gen AI Detection, Mixture-of-Experts, Transformers

1. Introduction

A significant advancement of transformer-based language models [1] has made a great impact in natural language processing (NLP) [2] capabilities, particularly in text classification tasks. These models, such as BERT [3], RoBERTa [4], and DeBERTa [5], have demonstrated remarkable performance across a wide range of benchmarks by capturing deep contextual representations and long-range dependencies in text. However, the computational complexity and resource demands of these models pose challenges for scalability and efficiency as the number of trained parameters increases. This growing computational burden limits their deployment in real-time and resource-constrained environments, such as mobile devices or edge computing platforms. Moreover, the trend of continuously enlarging model architectures to gain marginal improvements in accuracy raises concerns about energy consumption, inference latency, and environmental sustainability. As a result, there is a growing interest in developing lightweight, efficient, and scalable transformer variants or hybrid architectures that can retain high accuracy while significantly reducing computational overhead.

Mixture of Experts (MoE) [6] [7] architectures offer a promising solution by distributing the computational load across multiple specialized sub-networks, or "experts," each of which is responsible for handling different aspects or subsets of the input data. This dynamic allocation of processing tasks allows the model to activate only a small portion of the total parameters during inference, significantly reducing computational overhead while preserving or even enhancing performance.

In this study, we investigate the application of both Soft and Hard MoE frameworks integrated with transformer models, including DistilBERT [8], DeBERTa, ModernBERT [9], XLNet [10], RoBERTa [4], and ALBERT [11], for binary and multi-class text classification on the respective datasets. The Soft MoE dynamically combines expert outputs through a gating mechanism, while the Hard MoE selects a single expert per input, optimizing for computational sparsity. By leveraging the CLS token for classification and visualizing expert routing, we aim to evaluate the effectiveness of these MoE variants

CLEF 2025 – Conference and Labs of the Evaluation Forum, September 9–12, 2025, Madrid, Spain
first@author.com (F. Author); second@author.com (S. Author)

in improving classification performance, expert utilization, and computational efficiency, contributing to the development of scalable and robust NLP systems.

2. Task

Generative AI Detection This is a shared task organised by the PAN 2025 lab [12] on digital text forensics and stylometry. Furtherly, it is divided into two subtasks, Subtask 1 (*Webis*) AI Detection Sensitivity Analysis, Subtask 2 (*MBZUAI*) Fine-grained recognition of human-AI collaborated document.

Subtask 1 AI Detection Sensitivity Analysis for Identifying Unobfuscated and Obfuscated LLM-Generated Text.

Subtask 2 Detailed classification of documents created through human-AI collaboration: For a given document produced by humans and AI systems, assign it to one of these categories: (1) Fully human-written, (2) Human-initiated, then machine-continued, (3) Human-written, then machine-polished, (4) Machine-written, then machine-humanized (obfuscated), (5) Machine-written, then human-edited, (6) Deeply-mixed text sections.

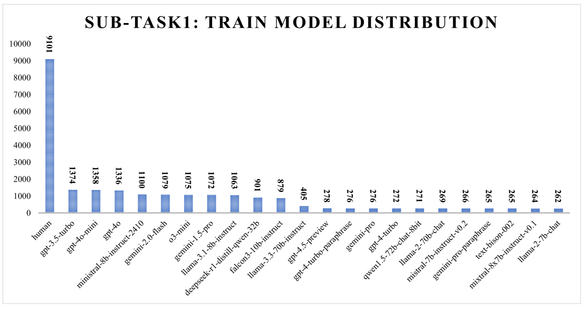
3. Dataset Statistics

This task provides two datasets presenting one for each subtask.

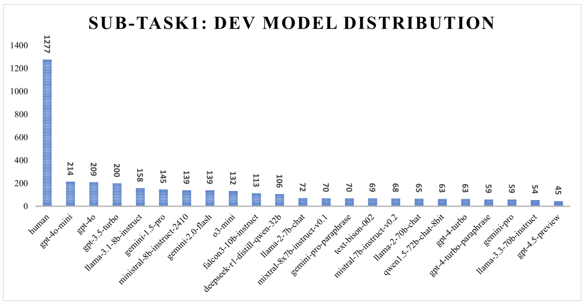
In Subtask 1 dataset is with Human texts and texts from the AI models are *gpt-3.5-turbo, gpt-4o-mini, gpt-4o, ministral-8b-instruct-2410, gemini-2.0-flash, o3-mini, gemini-1.5-pro, llama-3.1-8b-instruct, deepseek-r1-distill-qwen-32b, falcon3-10b-instruct, llama-3.3-70b-instruct, gpt-4.5-preview, gpt-4-turbo-paraphrase, gemini-pro, gpt-4-turbo, qwen1.5-72b-chat-8bit, llama-2-70b-chat, mistral-7b-instruct-v0.2, gemini-pro-paraphrase, text-bison-002, mixtral-8x7b-instruct-v0.1, llama-2-7b-chat.* Model Distribution follows the Figure 1 below.

Split	Human	Machine
Train	9,101	14,606
Dev	1,277	2,312

Table 1
Subtask 1 Dataset split by Human and Machine labels



(a) Distribution of Model in Train Set



(b) Distribution of Model in Dev Set

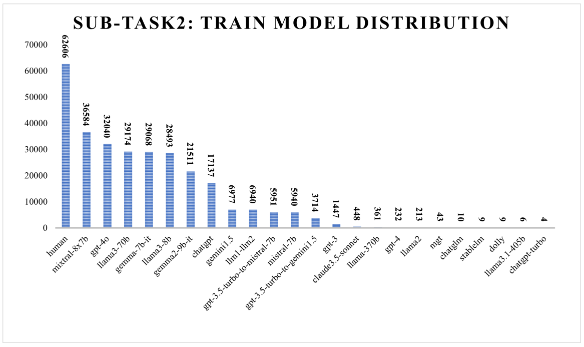
Figure 1: Distributions of Model with count in SubTask 1

In Subtask 2 dataset the dataset is made by the AI texts from the models *mixtral-8x7b, gpt-4o, llama3-70b, gemma-7b-it, llama3-8b, gemma2-9b-it, chatgpt, gemini1.5, llm1-llm2, gpt-3.5-turbo-to-mistral-7b, mistral-7b, gpt-3.5-turbo-to-gemini1.5, gpt-3, claude3.5-sonnet, llama-370b, gpt-4, llama2, mgt, chatglm, stablelm, dolly, llama3.1-405b, chatgpt-turbo.* Model Distribution follows the Figure 2 below.

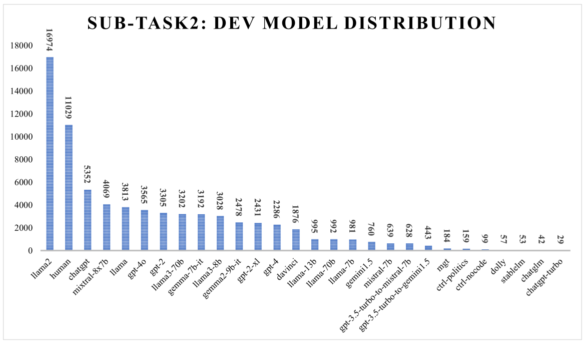
Further dataset textual analysis of both the subtasks are given in the appendix.

Split	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Train	95,398	91,232	75,269	14,910	10,740	1,368
Dev	37,170	12,330	12,289	10,137	510	225

Table 2
Subtask 2 Dataset distribution across different text generation classes, where Class 1: Human-written, then machine-polished, Class 2: Machine-written, then machine-humanized, Class 3: Fully human-written, Class 4: Deeply-mixed text; where some parts are written by a human and some are generated by a machine, Class 5: Human-initiated, then machine-continued, and Class 6: Machine-written, then human-edited



(a) Distribution of Model in Train Set



(b) Distribution of Model in Dev Set

Figure 2: Distributions of Model with count in SubTask 2

4. System Description

Our proposed system integrates a Mixture-of-Experts (MoE) architecture with transformer-based language models to enhance binary and multi-class text classification performance. We implemented two variants of MoE, namely SoftMoE and HardMoE, with 2 (*for subtask 1*) and 6 (*for subtask 2*) experts to a diverse set of pre-trained transformer models.

4.1. System Architecture

The system is built with a transformer-based backbone added with an MoE layer for classification. The transformer backbones include DistilBERT, RoBERTa, ALBERT, XLNet, DeBERTa, and ModernBERT, all base models.

Model	Layers	Hidden	Heads	Parameters
distilbert-base-uncased	6	768	12	66B
roberta-base	12	768	12	125B
albert-base-v2	12	768	12	11B
xlnet-base-cased	12	768	12	110B
deberta-v3-base	12	768	12	86B
modernbert-base	22	768	12	2T

Table 3
Model architecture comparison

4.2. MoE Layer

In our approach, we utilize two distinct types of Mixture of Experts (MoE) classifiers: HardMoE and SoftMoE. The HardMoE classifier operates using a discrete gating mechanism, where a lightweight linear gating network takes the CLS token output from the transformer layer, denoted as `Transformer(x)[:,0,:]`, and transforms it into a set of expert scores using the equation: $\mathbf{g} = \mathbf{W}g\ \mathbf{h}\texttt{CLS} + \mathbf{b}_g$. The expert associated with the highest score is chosen using an `argmax` operation, ensuring that only a single expert is activated per input. The selected expert processes the input and produces a prediction via a softmax layer. Additionally, the raw gating scores can be utilized for computing auxiliary losses during training.

In contrast, the SoftMoE classifier relies on a continuous, probabilistic gating mechanism. Instead of selecting just one expert, the gating network generates a score for each expert, which is then normalized using the softmax function to produce a set of attention-like weights. These weights are used to compute a weighted combination of all expert outputs, allowing the model to leverage information from all experts simultaneously. The core distinction between HardMoE and SoftMoE lies in this gating strategy: while HardMoE enforces a strict "winner-takes-all" approach, SoftMoE softly blends contributions from all available experts. The forward pass logic for both architectures, including the flow of data and the classification process, is detailed in Algorithm 0.

Soft MoE: Consists of 2 or 6 expert linear layers, each mapping the 768-dimensional CLS token to 2 or 6 output classes. A gating network (a linear layer followed by a softmax) computes weights for each expert, producing a weighted sum of expert outputs. Gate weights are stored for visualization.

Hard MoE: Similar to Soft MoE but selects a single expert per input based on the highest gate weight, enforcing computational sparsity. The MoE layer replaces the standard classification head, leveraging specialized expert knowledge for diverse input patterns.

Dropout: A dropout layer with a probability of 0.1 is applied to the CLS token before the MoE layer to mitigate overfitting.

Algorithm 1 Forward Pass for MoE Classifier (Hard and Soft)

```
1:  Input: $input\_ids$, $attention\_mask$
2:  Output: $output\_logits$, $gate\_logits$
3:  Extract $hidden\_state$ from base transformer
4:  Get CLS token: $cls \leftarrow hidden\_state[:, 0, :]$
5:  Apply dropout to $cls$
6:  Compute: $gate\_logits \leftarrow \text{Linear}(cls)$
7:  if model is HardMoE then
8:      $expert\_choice \leftarrow \arg\max(gate\_logits, \text{dim} = 1)$
9:      Initialize $output\_logits$ as zeros
10:     for each expert $i$ do
11:         $mask \leftarrow (expert\_choice == i)$
12:         if $mask$ not empty then
13:             $out \leftarrow expert_i(cls[mask])$
14:             $output\_logits[mask] \leftarrow out$
15:         end if
16:     end for
17: else SoftMoE
18:     $gate\_weights \leftarrow \text{Softmax}(gate\_logits)$
19:     $expert\_outputs \leftarrow []$
20:     for each expert $i$ do
21:         $out \leftarrow expert_i(cls)$
22:         Append $out$ to $expert\_outputs$
23:     end for
24:     Stack $expert\_outputs$
25:     $output\_logits \leftarrow \sum(\text{gate\_weights} \times \text{expert\_outputs})$
26: end if
27: return $output\_logits$, $gate\_logits$
```

4.3. Training Method

Models are trained on Amazon Web Services (*AWS*) Cloud server, Amazon Elastic Compute Cloud (*EC2*) instance. In the EC2 instance, we initiated an instance for Accelerated Computing. The specifications are **g6e.xlarge** instance, which provides **3rd generation AMD EPYC processors (*AMD EPYC 7R13*)**, with a **NVIDIA L40S Tensor Core GPU with 48 GB GPU memory**, and 4x vCPU with 32 GiB memory and a network bandwidth of 20GBps, and our OS type is **Ubuntu Server 24.04 LTS (*HVM*), EBS General Purpose (*SSD*) Volume Type.**

Models are trained on a CUDA-enabled GPU, and for all the models the hyperparameter settings are as follows: the batch-size is 32, the maximum sequence length is 512, AdamW optimizer with a learning rate of 1e-5 and weight decay of 0.01, Cross-entropy loss, `ReduceLROnPlateau` reduces the learning rate by a factor of 0.1 if validation loss plateaus for 1 epoch, up to 10 epochs with early stopping, with a maximum mean of ROC-AUC, Brier, c@1, F1, F0.5u for Subtask 1, and maximum Recall for Subtask 2.

5. Results

For subtask 1, we submitted our best-performing model to TIRA [13] for further execution, and for subtask 2, we submitted the corresponding .zip file which contained a predictions.jsonl file with 'id' and 'label' in CodaLab. Table 4 shows the performance of models on subtask 1 in val-set and smoke-test set. Table 5 shows the performance of the models in subtask 2 in the dev set. The Auc-Roc curve of few models.

Dataset	Model	Roc-Auc	Brier	C@1	F1	F0.5u	Mean
Val	ALBERT-v2-base-HardMoE	0.995	0.995	0.995	0.996	0.997	0.995
	DeBERTa-v3-Large-HardMoE	0.996	0.996	0.996	0.997	0.998	0.996
Smoke-test	ALBERT-v2-base-HardMoE	0.938	0.941	0.941	0.933	0.972	0.945
	DeBERTa-v3-Large-HardMoE	0.969	0.971	0.971	0.968	0.987	0.973

Table 4
Subtask 1 Performance metrics comparison across validation and smoke-test datasets

Method	Model	Accuracy	Macro F1	Macro-Recall
SoftMoE	DistilBERT-base-uncased	0.552	0.589	0.687
	ALBERT-base-v2	0.528	0.46	0.536
	DeBERTa-v3-base	0.535	0.576	0.743
	ModernBERT-base	0.555	0.626	0.731
	RoBERTa-base	0.544	0.629	0.772
	XLNet-base-uncased	0.563	0.612	0.775
HardMoE	DistilBERT-base-uncased	0.56	0.601	0.701
	ALBERT-base-v2	0.561	0.64	0.777
	DeBERTa-v3-base	0.514	0.581	0.74
	ModernBERT-base	0.543	0.62	0.728
	RoBERTa-base	0.547	0.587	0.798
	XLNet-base-uncased	0.545	0.595	0.781
	RoBERTa-large	0.575	0.616	0.782
	DeBERTa-v3-large	**0.612**	**0.683**	**0.818**

Table 5
Subtask 2 Performance of SoftMoE and HardMoE methods across different models on Dev Set

6. Conclusion

In this paper, we presented our work to the PAN: Voight-Kampff Generative AI Detection 2025. We used the Mixture-of-Experts architecture with several transformer backbones and checked which model gives better performance, surpassing the baselines. An ablation study on expert routing highlights the critical role of the gating mechanism in enhancing performance. However, further analysis of misclassified cases could uncover specific weaknesses for future improvement. Our findings highlight the scalability, interpretability, and superior performance of MoE-enhanced transformers, establishing a robust framework for advancing generative AI detection and making a significant contribution to the tasks.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[2] K. Chowdhary, K. Chowdhary, Natural language processing, Fundamentals of artificial intelligence (2020) 603–649.

[3] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[5] P. He, X. Liu, J. Gao, W. Chen, Deberta: Decoding-enhanced bert with disentangled attention. arxiv 2020, arXiv preprint arXiv:2006.03654 (2006).

[6] S. Masoudnia, R. Ebrahimpour, Mixture of experts: a literature survey, Artificial Intelligence Review 42 (2014) 275–293.

[7] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, arXiv preprint arXiv:1701.06538 (2017).

[8] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108 (2019).

[9] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallstrom, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, et al., Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference (2024), arXiv preprint arXiv.2412.13663 (????).

[10] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, Advances in neural information processing systems 32 (2019).

[11] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, arXiv preprint arXiv:1909.11942 (2019).

[12] J. Bevendorff, D. Dementieva, M. Fröbe, B. Gipp, A. Greiner-Petter, J. Karlgren, M. Mayerl, P. Nakov, A. Panchenko, M. Potthast, A. Shelmanov, E. Stamatatos, B. Stein, Y. Wang, M. Wiegmann, E. Zangerle, Overview of PAN 2025: Generative AI Authorship Verification, Multi-Author Writing Style Analysis, Multilingual Text Detoxification, and Generative Plagiarism Detection, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2025), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2025.

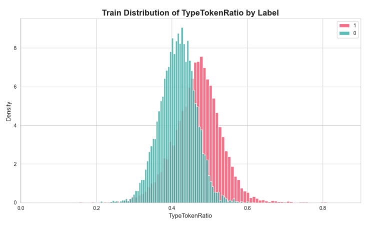
[13] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein,

M. Potthast, Continuous Integration for Reproducible Shared Tasks with TIRA.io, in: J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2023, pp. 236–241. URL: https://link.springer.com/chapter/10.1007/978-3-031-28241-6_20. doi:10.1007/978-3-031-28241-6_20.

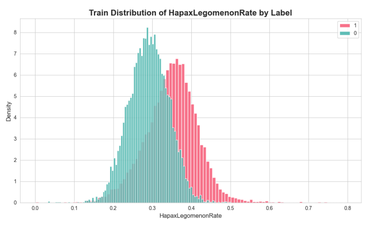
A. Data Analysis

A.1. Sub Task 1 dataset

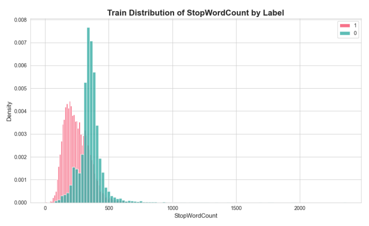
We have visualised how the data is by the following linguistic features by label count: 1) Stop Word Count, 2) Hapax Legomenon Rate, 3) Type Token Ratio.



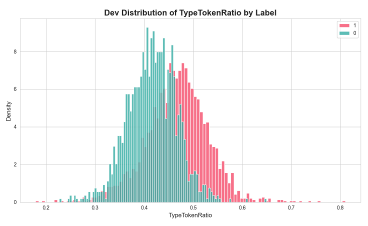
(a) Subtask 1, train dataset Type Token Ratio histograms for both Human and Machine Labels.



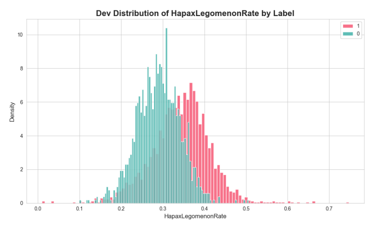
(b) Subtask 1, train dataset Hapax Legomenon Rate histograms for both Human and Machine Labels.



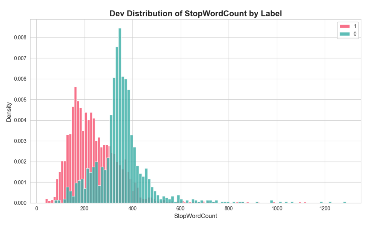
(c) Subtask 1, train dataset Stop Word Count histograms for both Human and Machine Labels.



(d) Subtask 1, dev dataset Type Token Ratio histograms for both Human and Machine Labels.



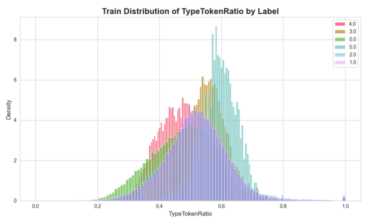
(e) Subtask 1, dev dataset Hapax Legomenon Rate histograms for both Human and Machine Labels.



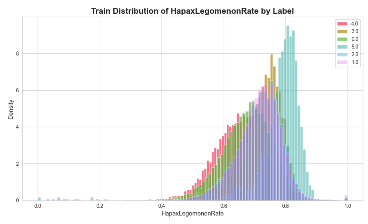
(f) Subtask 1, dev dataset Stop Word Count histograms for both Human and Machine Labels.

A.2. Sub Task 2 dataset

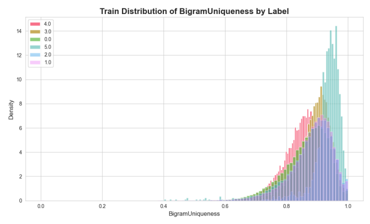
We have visualised how the data is by the following linguistic features by label count: 1) Bigram Uniqueness, 2) Hapax Legomenon Rate, 3) Type Token Ratio.



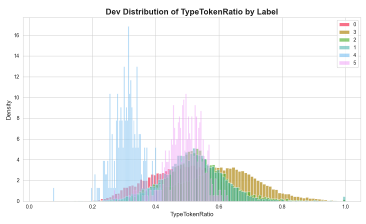
(a) Subtask 2, train dataset Type Token Ratio histograms for all classes.



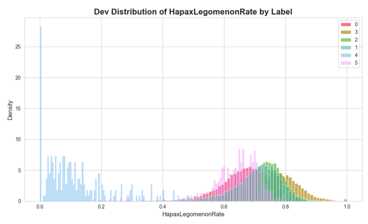
(b) Subtask 2, train dataset Hapax Legomenon Rate histograms for all classes.



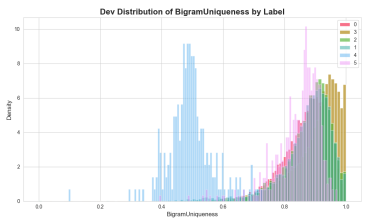
(c) Subtask 2, train dataset Bi-Gram Uniqueness histograms for all classes.



(d) Subtask 2, dev dataset Type Token Ratio histograms for all classes.

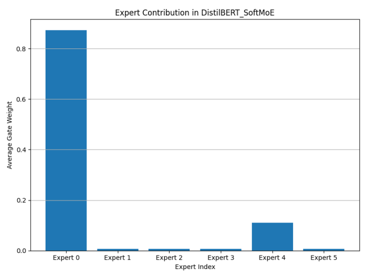


(e) Subtask 2, dev dataset Hapax Legomenon Rate histograms for all classes.

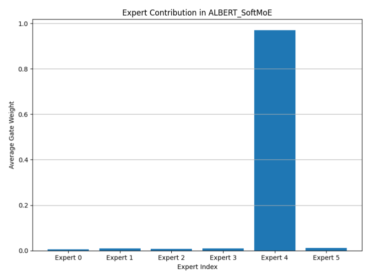


(f) Subtask 2, dev dataset BiGram Uniqueness histograms for all classes.

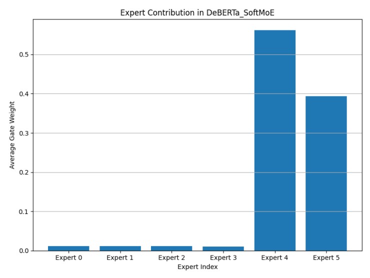
B. SoftMoE Expert Routing SubTask 2



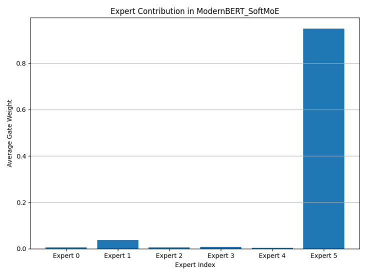
(a) DistilBERT Soft MoE Expert Routing



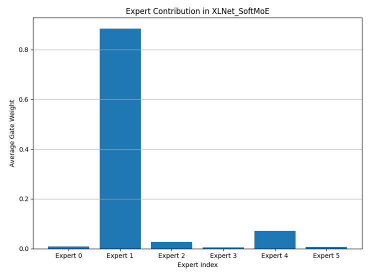
(b) ALBERT Soft MoE Expert Routing



(c) DeBERTa Soft MoE Expert Routing



(d) ModernBERT Soft MoE Expert Routing

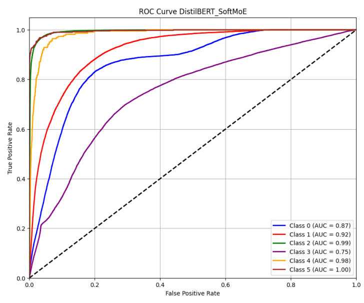


(e) RoBERTa Soft MoE Expert Routing

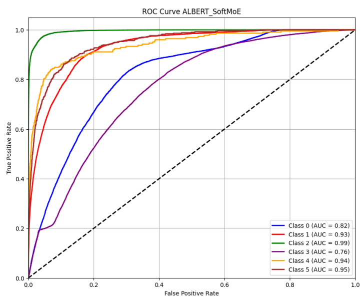
Figure 5: All Expert Routing plots of SoftMoE models

C. AUC-ROC Curves SubTask2

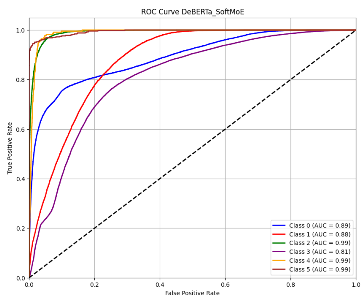
C.1. SoftMoE AUC-ROC



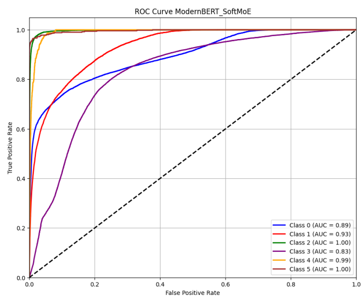
(a) DistilBERT Soft MoE AUC-ROC Curves



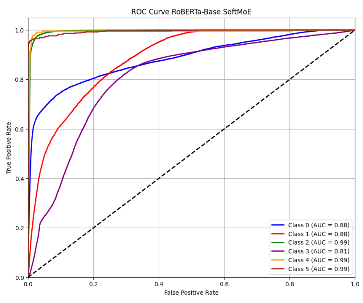
(b) ALBERT Soft MoE AUC-ROC Curves



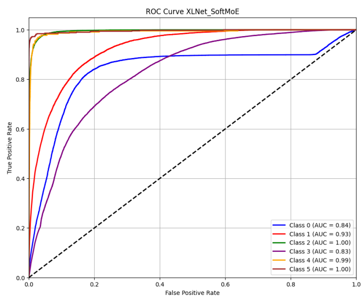
(c) DeBERTa Soft MoE AUC-ROC Curves



(d) ModernBERT Soft MoE AUC-ROC Curves



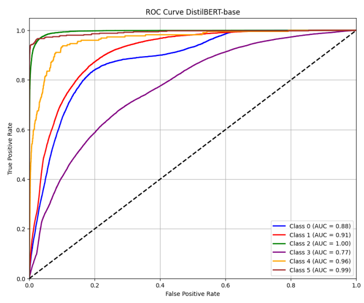
(e) RoBERTa Soft MoE AUC-ROC Curves



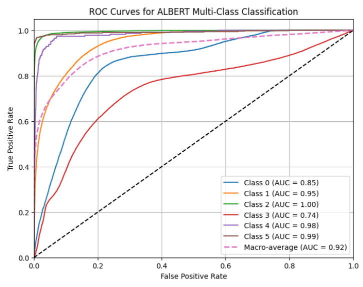
(f) XLNet Soft MoE AUC-ROC Curves

Figure 6: Six AUC-ROC plots of SoftMoE

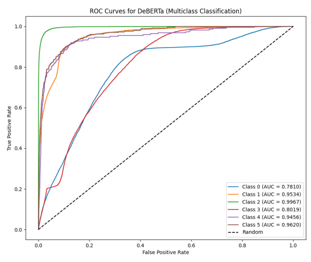
C.2. HardMoE AUC-ROC



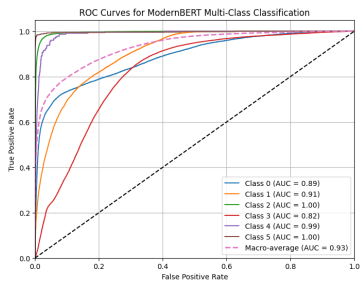
(a) DistilBERT Hard MoE AUC-ROC Curves



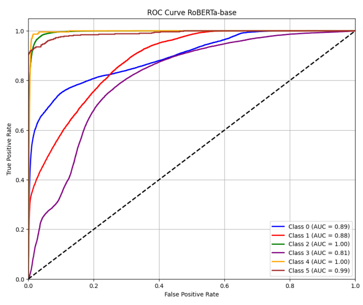
(b) ALBERT Hard MoE AUC-ROC Curves



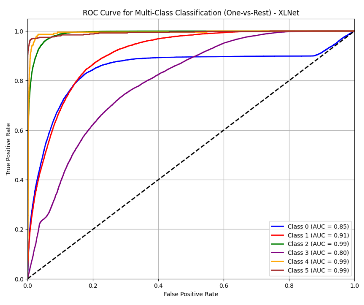
(c) DeBERTa Hard MoE AUC-ROC Curves



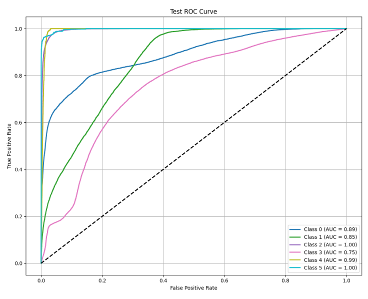
(d) ModernBERT Hard MoE AUC-ROC Curves



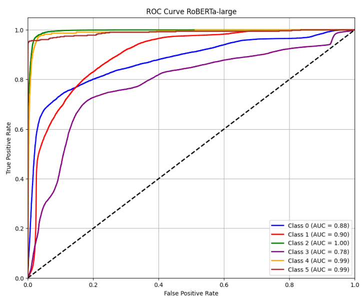
(e) RoBERTa-base Hard MoE AUC-ROC Curves



(f) XLNet Hard MoE AUC-ROC Curves



(g) DeBERTa-V3-Large Hard MoE AUC-ROC Curves



(h) RoBERTa-Large Hard MoE AUC-ROC Curves

Figure 7: Eight AUC-ROC plots of HardMoE