Team PRMAS: Datathon@IndoML 2024

Priyanshu Raj Mall
Indian Institute of Science Education and Research (IISER)
Kolkata, India
mall.priyanshu07@gmail.com

Aman Sinha
Indian Institute of Science Education and Research (IISER)
Kolkata, India
ai.amansinha@gmail.com

Abstract

This report presents the solution by Team PRMAS for Datathon @ IndoML 2024. Our final leaderboard approach involves a hard-voting ensemble of three distinct models. The first model is ByT5, trained on the description field to predict labels through its generated text. The second approach is a multi-step pipeline: we efficiently LoRA fine-tune an LLM to generate complete product names, which are then used as inputs for a Flan-T5 model to predict labels. The third model is a multi-modal adaptation of ByT5 that processes three input types—description as text, retailer as a categorical feature, and price as a numerical feature—to output labels in text form. For each model, we corrected output hallucinations and utilized label hierarchy to enhance performance. This innovative ensemble achieved an item accuracy of 0.4764, securing our team the 2nd position on the leaderboard.

Keywords

Label Prediction, E-Commerce

Reference Format:
Priyanshu Raj Mall and Aman Sinha. 2024. Team PRMAS: Datathon@IndoML 2024.

1 Approach

We began Phase 2 of the Datathon by thoroughly exploring the dataset to gain initial insights. The competition task description notes that the short product descriptions were extracted from receipts or invoices. Upon manual inspection of the data, we observed that the `description` field contained typical shorthand characteristics found in receipts, such as truncated words, abbreviations, omitted vowels, and shorthand brand names. However, we also identified other anomalies, including character mutations (e.g., character replacement, insertion, and deletion), which were systematically present across the text in alphabetical order, often with letters replaced in fixed positions while the rest of the text remained unchanged. These errors were difficult to attribute to natural receipt/invoice text processing, suggesting possible artificial noise in the dataset.[1] Consequently, we concluded that constructing a reliable dictionary mapping abbreviations to expanded forms would not be feasible.

[1]We chose not to delve further into whether the descriptions were artificially obscured, as opposed to real-world data, leaving this determination to the organizers.

For IndoML 2024 Datathon

Given the noisy nature of the text, we anticipated that traditional tokenization methods might fail to produce semantically meaningful tokens. Therefore, we explored models that operate on a character level or are tokenizer-free. We identified CANINE [2] and ByT5 [4] as promising options. We initially trained two Multi-Task Classification models using CANINE (-c and -s) embeddings on the `description`, `retailer`, and `price` fields; however, these models yielded poor item accuracy on the intermediate test set. Based on insights from the ByT5 paper [4], which demonstrated the model's robustness to synthetic noise, we fine-tuned ByT5-small using `description`, `retailer`, and `price` as input and all four labels as output, achieving competitive item accuracy. To further improve, we freshly retrained ByT5-small using only `description` as input to avoid possible model deviations from digit-level patterns from the `price` as text. This approach yielded an item accuracy of 0.4387, securing a strong position on the leaderboard with a relatively compact model. Even if we had stopped here we would have been at the 2nd position on the leaderboard, but we pushed forth.

We then sought to enhance the semantic quality of the product descriptions to increase their predictive power. We experimented with different LLMs to complete these descriptions but found that even GPT-4 struggled with accurate expansion in the absence of contextual information. To overcome this, we crafted a prompt for training data that included `price`, `supergroup`, `group`, `module`, and `brand` as inputs, asking the model to generate the full product name. We found that `Llama 3.1 8B-Instruct` [3] was the smallest open-source model capable of deriving meaningful outputs from noisy descriptions. For the test data, where label information would be unavailable, we LoRA fine-tuned `Llama 3.1 8B-Instruct` using only `description` and `price` as inputs and the previously inferred full product names as responses. We used this fine-tuned Llama model to infer product names for the test data. Finally, we fine-tuned a flan-T5-base [1] model with the "inferred description" as input and the target labels as output, achieving an item accuracy of 0.4129.

Upon realizing that including retailer names and prices as text hindered model performance, we explored alternative ways to incorporate these features. Specifically, we wanted to use the retailer as a categorical variable and `price` as a numerical feature. To integrate these into the model, we appended the retailer category and price to the end of the ByT5 encoder's last hidden state, effectively transforming ByT5 into a multi-modal model capable of processing categorical and numerical variables alongside text. We trained this multi-modal model end-to-end with the transformers library, but it only achieved an item accuracy of 0.3832, despite showing a better validation loss than the ByT5-small model trained on `description` alone. We speculate that a slight shift in price distribution between the training and test sets may have affected performance on the test set.

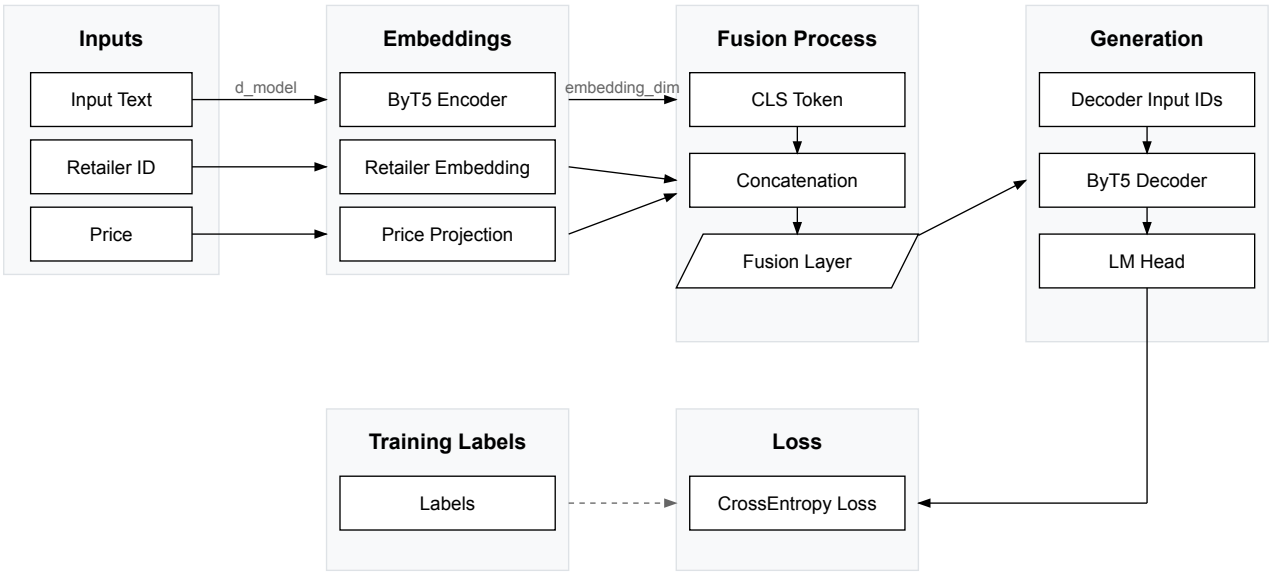

Figure 1: Multimodal ByT5 Architecture (modification)

To improve accuracy, we refined the model outputs across all three models, ensuring compliance with the hierarchical structure observed in the training data and correcting any hallucinations. This refinement led to a 1% to 3% boost in item accuracy for each model.

With three models yielding competitive results, we implemented a hard-voting ensemble strategy, incorporating hierarchy compliance across models. This approach resulted in our final leaderboard item accuracy of 0.4764.

2 Final Submission Solution Model

We group the provided train data by their 4 labels and randomly withdraw 10% of the data rows from each group to construct our validation set. We sample group-wise so that train and validation set both will have all the labels.

2.1 Models

2.1.1 ***Model-1: ByT5-small fine-tuned.*** We employed the pre-trained `ByT5-small` model (300M parameters) [4] as the foundation for our label prediction task, fine-tuning it on product descriptions to generate hierarchical labels as structured text. In training, we used only the `description` field as input text and generated a structured output text consisting of concatenated label types and their values (e.g., "supergroup: ..., group: ..., module: ..., brand: ..."). We fine-tuned `ByT5-small` using Hugging Face's `Trainer` API, employing a series of training parameters optimized for large datasets, including a batch size of 32, a learning rate of 5e-4, gradient accumulation, and early stopping to avoid overfitting. This setup allowed us to efficiently predict hierarchical labels based on product descriptions alone, achieving a strong performance metric on the leaderboard.

2.1.2 ***Model-2: Llama+FlanT5 fine-tuned.*** The Llama+FlanT5 pipeline was designed to enhance product descriptions, thereby improving item accuracy and hierarchical label prediction. For this, we utilized the `Llama 3.1 8B-Instruct` [3] model to generate enriched product names by prompting it with available descriptions, price, and label information. Recognizing that noisy descriptions alone were insufficient for accurate expansion—even for advanced models like GPT-4—we included specific product labels in the prompt to guide the Llama model toward accurate predictions. During inference on the test set, where label data was absent, we fine-tuned Llama to generate inferred descriptions based only on `description` and `price` fields.

The pipeline then used these enriched descriptions as inputs to the `Flan-T5-base` model [1], which was fine-tuned to predict hierarchical labels (`supergroup`, `group`, `module`, and `brand`) based on these enhanced descriptions. This configuration allowed Flan-T5 to deliver more accurate and contextually aligned label predictions, effectively using the Llama-inferred descriptions as refined input.

For efficient fine-tuning of `Llama 3.1 8B-Instruct`, we applied LoRA (Low-Rank Adaptation), which enables parameter-efficient adaptation by focusing on the low-rank updates of attention heads. Using LoRA, we modified key projections in the model's transformer layers (such as `q_proj`, `k_proj`, `v_proj`, and others) while keeping the original model weights largely intact. This method allowed us to enhance the model's inference capability with minimal additional resources.

The fine-tuning process leveraged a custom chat-style prompt format to instruct the Llama model on generating complete product names. The prompt structure included a system message identifying the assistant as a knowledgeable product name generator and a user message containing the `description` and `price` fields. Training

utilized the Hugging Face `SFTTrainer` (Supervised Fine-Tuning Trainer), configured with an optimized setup that included 8-bit model weights, gradient accumulation steps for batch efficiency, and an AdamW optimizer with a learning rate of `3e-4` and linear scheduling. We applied gradient checkpointing and saved the final model in a 16-bit format for efficient deployment.

2.1.3 ***Model-3: MultiModal ByT5 fine-tuned.*** The MultiModal ByT5-small model modifies the `ByT5-small` architecture by incorporating embeddings for retailer IDs and price data, providing a richer multi-modal input that combines categorical and numerical information with text embeddings. Specifically, each unique retailer is assigned an integer ID and embedded into an 8-dimensional vector using a dedicated embedding layer. Price values are first normalized with a MinMaxScaler and then projected through a linear layer to an 8-dimensional vector, aligning with the retailer embedding dimensions. These retailer and price embeddings are concatenated with the 512-dimensional text embedding output from the ByT5 encoder's [CLS] token, forming a combined vector of dimension 528.

This 528-dimensional concatenated vector is then processed by a fusion layer—a fully connected linear layer—that projects it back to the original 512-dimensional space of the ByT5 encoder. The output of this fusion layer replaces the [CLS] token in the encoder's hidden states, embedding both categorical and numerical information directly into the textual representation. The modified architecture is in the Figure-1.

Throughout training, these retailer ID and price embeddings are incorporated at every step using Hugging Face's `Trainer` API, with fine-tuned hyperparameters optimized for multi-modal processing. The model learns to condition its predictions on this integrated input, enabling it to generalize label predictions across diverse product descriptions by leveraging the combined information from text, retailer, and price in a unified representation.

2.2 Techniques

2.2.1 ***Aiding LLM to generate Accurate Product Names.*** One of our key innovations was generating near-accurate product names by fine-tuning a large language model (LLM) using hierarchical label data, including specific brand names. While initial attempts to expand abbreviated or noisy product descriptions using only the description field were unsuccessful—even with advanced closed-source models—this approach proved remarkably effective when the model was provided with all hierarchical labels as input. By incorporating `supergroup`, `group`, `module`, and `brand` labels, we trained the LLM to predict detailed and contextually appropriate product names. This targeted input enabled the model to infer accurate product names despite the limited context and noisy nature of the descriptions, achieving a level of precision that was otherwise unattainable with traditional input data.

2.2.2 ***Hallucination Correction.*** During analysis of our model's output, we discovered that some predicted labels did not align with the predefined classes within each hierarchy. This deviation, a common issue with language models known as hallucination, arises when the model generates outputs outside of the valid label set. To mitigate this, we first identified predictions that fell outside the acceptable label classes. Next, we used the `difflib` library to match these incorrect predictions with the most similar valid labels within the respective classes. This correction method was initially applied to the `brand` category, where we found that hallucinated labels often bore close resemblance to valid options. By correcting these hallucinations, we maintained label accuracy and coherence, ensuring that outputs adhered closely to the valid class set for each category.

2.2.3 ***Leveraging Label Hierarchy.*** To further enhance label accuracy, we utilized the hierarchical structure of the training data, which defined relationships between `supergroup`, `group`, `module`, and `brand` categories. This hierarchy enabled us to constrain predictions by context, ensuring that only valid labels for a given hierarchical path were considered. For example, rather than correcting `brand` predictions based on a global list of possible brands, we limited each correction to brands valid within a specific combination of `supergroup`, `group`, and `module` labels. This approach reduced mismatches and increased accuracy, as corrections were contextually appropriate and aligned with the logical label hierarchy. Seeing the success of this technique for `brand`, we extended it to `group` and `module` labels, refining model predictions across categories and boosting consistency throughout the hierarchy.

2.2.4 ***Simple Yet Effective Hard-Voting Ensemble Method.*** To capitalize on the strengths of our trained models, we employed a straightforward hard-voting ensemble technique. This approach involved combining predictions from three distinct models, allowing each model's predictions to contribute to the final output based on majority agreement. Hard voting proved to be an effective method to consolidate individual model strengths, filtering out idiosyncrasies and mitigating the weaknesses of any single model. This ensemble approach, despite its simplicity, significantly improved the overall performance by reinforcing consistent predictions, ultimately increasing model robustness and enhancing our leaderboard accuracy.

3 Novelty

In analyzing the data, we recognized the pervasive noise patterns within product descriptions and quickly identified the need for tokenizer-free models capable of handling such irregular text. After a thorough review of literature, we selected ByT5 as an ideal model for this challenge. While the world is using LLMs as the ultimate hammer for every problem, we recognized that using an LLM would be impractical for a focused task like E-commerce attribute prediction. Instead, we aimed to achieve competitive item accuracy and F1 scores with models under 300M parameters, making ByT5 a strategic choice.

For cases where we expected the LLM to infer product names from noisy descriptions, we found that all tested LLMs—open-source and closed-source—failed to expand abbreviations accurately. However, by structuring the prompt with hierarchical labels alongside the product description, we guided the LLM to produce accurate product names. Using LoRA fine-tuning on an 8B-parameter LLM, we enabled the model to learn label structures while retaining its internal product catalog knowledge. Given our objective to avoid unnecessary computational overhead, we refrained from using an

Model	Description	Inferred Description	Retailer	Price	Hallucination Correction	Item Accuracy	Supergroup F1	Group F1	Module F1	Brand F1
ByT5-small	✓					0.4386	0.5881	0.4869	0.4612	0.2523
ByT5-small	✓				✓	0.4510	0.5881	0.4976	0.4737	0.3816
Flan-T5-base		✓				0.4129	0.5623	0.4341	0.4071	0.2329
Flan-T5-base		✓			✓	0.4219	0.5799	0.4645	0.4371	0.3447
MultiModal ByT5	✓		✓	✓		0.3832	0.5696	0.4749	0.4486	0.3782
MultiModal ByT5	✓		✓	✓	✓	0.3862	0.5696	0.4749	0.4482	0.2977
Hard Voting	✓	✓	✓	✓	✓	0.4764	0.6176	0.5172	0.4882	0.4198

Table 1: Performance of our models before and after hallucination correction and compliance with hierarchy. Finally, we employed the Hard Voting ensemble strategy using the results from the three models mentioned above.

Model	No. of Model Parameters	Self CPU Time Total (ms)	Self CUDA Time Total (ms)	GFLOPs during Training	Inference Time (s)	GFLOPs during Testing
ByT5-small fine-tuned	299,637,760	516.213	139.091	205.937177802	4.83382835239172	13.246457007
Llama 3.1 8B-Instruct ft	8,072,204,288	629.991	221.309	5327.74754803	6.693120772019029	1740.513683872
Flan-T5-base fine-tuned	247,577,856	605.048	140.625	293.535165616	6.7896833550185	8.608508107
MultiModal ByT5-small	301,830,088	608.943	149.218	92.084084778	5.75746718235314	13.246457007

Table 2: Overall Efficiency metrics table

LLM for direct label prediction, instead leveraging a more efficient T5 model (Flan-T5-small) for hierarchy-aware, autoregressive classification.

While many solutions overlook numerical and categorical features, favoring text-based NLP models, we chose to leverage all available data in its native form. Realizing that pretrained embeddings struggled with noisy descriptions, we modified ByT5 to incorporate price and retailer data as structured inputs, extending ByT5's natural handling of noisy text to a multi-modal architecture. This modification enabled ByT5 to process text, numerical, and categorical features simultaneously. Although our experiments with this multi-modal ByT5 did not outperform the standard ByT5 in this Datathon, we believe that with further refinement, the multi-modal T5 setup has potential for improved performance on similar tasks. Our diverse approach provides an alternative perspective and inspiration for future work on noisy and structured e-commerce data.

4 Results

The results presented in Table-1 highlight the performance metrics of various models employed during the experiment, particularly in terms of item accuracy. Notably, the Hard Voting ensemble strategy achieved an highest item accuracy of 0.4764, showcasing the effectiveness of combining outputs from multiple models to enhance overall performance.

4.1 Efficiency

The efficiency metrics evaluated using the provided code[2] are presented in the Table-2 and Table-3.

[2]https://colab.research.google.com/drive/1353eGUwGFsQrXRGJdyWUPyGN6E3g3EJh?usp=sharing

5 Reproducibility

We conducted our training and inference using an RTX 4090 GPU equipped with 24GB of VRAM and a minimum of 16GB of system RAM. All our code is organized in a directory called `codes`, where executing `run.sh` will seamlessly install the necessary dependencies and execute all pre-processing, training, inference, and post-processing scripts in a single run. The prediction results on the test set will be stored in the `results` directory. Within the `codes` directory, each model or approach discussed in this report has its own subdirectory containing the corresponding code files in `.py` format.

6 Discussion

Our approach underscores the importance of model selection and architecture adaptation when dealing with noisy, real-world text data. By identifying the limitations of traditional LLMs for structured attribute prediction, we demonstrated the effectiveness of tokenizer-free models like ByT5, complemented by lightweight LLM fine-tuning strategies. Leveraging label hierarchies allowed us to control for hallucinations and improve label coherence, while a multi-modal extension of ByT5 highlighted the potential for integrating diverse data types in a single model. Though our multi-modal variant did not surpass the performance of vanilla ByT5, it opens avenues for future research in using multi-source data for E-commerce applications. Our strategy, from model selection to hierarchy-aware corrections, demonstrates a thoughtful balance between performance, computational efficiency, and adaptability to complex, noisy datasets.

References

[1] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024.

Scaling instruction-finetuned language models. *Journal of Machine Learning Research* 25, 70 (2024), 1–53.

[2] Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Transactions of the Association for Computational Linguistics* 10 (2022), 73–91. https://doi.org/10.1162/tacl_a_00448

[3] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[4] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models. *Transactions of the Association for Computational Linguistics* 10 (2022), 291–306. https://doi.org/10.1162/tacl_a_00461

Received 31 October 2024

ByT5-small fine-tuned										
Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls	Total MFLOPs
aten::mm	2.87%	14.806ms	11.96%	61.717ms	159.476us	9.216ms	6.63%	9.216ms	23.814us	387	204031.918
aten::bmm	0.60%	3.081ms	0.96%	4.947ms	41.228us	1.345ms	0.97%	1.345ms	11.212us	120	1736.442
aten::mul	1.84%	9.503ms	4.33%	22.333ms	35.392us	1.218ms	0.88%	1.218ms	1.930us	631	143.094
aten::add	0.50%	2.589ms	0.91%	4.717ms	30.431us	230.268us	0.17%	230.268us	1.486us	155	25.724
aten::to	0.33%	1.705ms	2.14%	11.049ms	14.481us	0.000us	0.00%	16.609us	0.022us	763	–
aten::_has_compatible_shallow_copy_type	0.01%	36.954us	0.01%	36.954us	0.107us	0.000us	0.00%	0.000us	0.000us	344	–
aten::empty	1.35%	6.993ms	1.35%	6.993ms	12.998us	0.000us	0.00%	0.000us	0.000us	538	–
aten::lift_fresh	0.01%	37.420us	0.01%	37.420us	0.211us	0.000us	0.00%	0.000us	0.000us	177	–
aten::detach_	0.06%	326.291us	0.08%	434.105us	2.467us	0.000us	0.00%	0.000us	0.000us	176	–
detach_	0.02%	107.814us	0.02%	107.814us	0.613us	0.000us	0.00%	0.000us	0.000us	176	–
Llama 3.1 8B-Instruct - LoRA fine-tuned											
Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls	Total MFLOPs
aten::mm	10.25%	64.562ms	15.57%	98.068ms	33.654us	68.849ms	31.11%	68.849ms	23.627us	2914	5327604.941
aten::add	0.39%	2.464ms	0.61%	3.845ms	21.722us	368.546us	0.17%	368.546us	2.082us	177	78.644
aten::mul	1.54%	9.675ms	2.30%	14.459ms	20.805us	725.051us	0.33%	725.051us	1.043us	695	63.964
cudaStreamSynchronize	0.44%	2.781ms	0.44%	2.781ms	69.528us	0.000us	0.00%	0.000us	0.000us	40	–
aten::amax	0.06%	374.381us	1.35%	8.518ms	2.839ms	2.241ms	1.01%	2.241ms	746.939us	3	–
aten::as_strided	0.81%	5.112ms	0.81%	5.112ms	0.880us	0.000us	0.00%	0.000us	0.000us	5809	–
cudaLaunchKernel	14.34%	90.354ms	14.34%	90.354ms	7.836us	0.000us	0.00%	0.000us	0.000us	11531	–
void at::native::reduce_kernel<512, 1, at::native::R...	0.00%	0.000us	0.00%	0.000us	0.000us	2.239ms	1.01%	2.239ms	1.119ms	2	–
aten::le	0.04%	232.438us	0.06%	395.461us	197.730us	3.904us	0.00%	3.904us	1.952us	2	–
aten::where	0.00%	20.901us	0.26%	1.632ms	815.860us	0.000us	0.00%	17.217us	8.609us	2	–
flan-T5-base fine-tuned											
Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls	Total MFLOPs
aten::mm	3.63%	21.990ms	10.69%	64.671ms	99.341us	12.864ms	9.15%	12.864ms	19.760us	651	279284.023
aten::bmm	0.88%	5.325ms	1.69%	10.214ms	47.286us	3.534ms	2.51%	3.534ms	16.360us	216	13982.810
aten::mul	2.37%	14.338ms	4.52%	27.336ms	27.364us	2.083ms	1.48%	2.083ms	2.085us	999	229.163
aten::add	0.71%	4.291ms	1.83%	11.070ms	45.555us	352.446us	0.25%	352.446us	1.450us	243	39.169
aten::to	0.33%	2.020ms	1.97%	11.900ms	9.573us	0.000us	0.00%	18.241us	0.015us	1243	–
aten::_has_compatible_shallow_copy_type	0.01%	54.102us	0.01%	54.102us	0.095us	0.000us	0.00%	0.000us	0.000us	568	–
aten::empty	0.76%	4.578ms	1.20%	7.280ms	8.485us	0.000us	0.00%	0.000us	0.000us	858	–
aten::lift_fresh	0.01%	56.121us	0.01%	56.121us	0.194us	0.000us	0.00%	0.000us	0.000us	289	–
aten::detach_	0.08%	487.845us	0.10%	632.240us	2.195us	0.000us	0.00%	0.000us	0.000us	288	–
detach_	0.02%	144.395us	0.02%	144.395us	0.501us	0.000us	0.00%	0.000us	0.000us	288	–
MultiModal ByT5-small fine-tuned											
Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls	Total MFLOPs
aten::mm	2.65%	16.142ms	9.50%	57.825ms	148.269us	11.093ms	7.43%	11.093ms	28.443us	390	289126.330
aten::bmm	0.53%	3.239ms	0.95%	5.787ms	48.226us	1.679ms	1.13%	1.679ms	13.994us	120	2717.909
aten::mul	1.62%	9.894ms	3.53%	21.482ms	34.045us	1.469ms	0.98%	1.469ms	2.329us	631	199.422
aten::add	0.45%	2.763ms	0.99%	6.056ms	39.070us	269.440us	0.18%	269.440us	1.738us	155	36.043
aten::addmm	0.04%	260.231us	1.69%	10.295ms	5.147ms	16.768us	0.01%	16.768us	8.384us	2	4.381
aten::to	0.27%	1.635ms	1.88%	11.463ms	14.621us	0.000us	0.00%	18.239us	0.023us	784	–
aten::_has_compatible_shallow_copy_type	0.01%	37.232us	0.01%	37.232us	0.105us	0.000us	0.00%	0.000us	0.000us	354	–
aten::empty	0.53%	3.209ms	0.53%	3.209ms	5.762us	0.000us	0.00%	0.000us	0.000us	557	–
aten::lift_fresh	0.01%	38.890us	0.01%	38.890us	0.211us	0.000us	0.00%	0.000us	0.000us	184	–
aten::detach_	0.05%	309.473us	0.07%	404.350us	2.210us	0.000us	0.00%	0.000us	0.000us	183	–

Table 3: Training efficiency metrics table