Efficient Vocabulary Reduction for Small Language Models

Yuta Nozaki[*], **Dai Nakashima**[*], **Ryo Sato**[*],
Naoki Asaba[*], **Shintaro Kawamura**[*]
[*]Ricoh Company, Ltd.
`{yuta.nozaki1,dai.nakashima,ryo.sato4,`
`naoki.asaba,shintaro.kawamura}@jp.ricoh.com`

Abstract

The increasing size of large language models (LLMs) poses significant challenges due to their high computational costs and energy consumption, making their deployment in industrial settings difficult. Small language models (SLMs) have been introduced to mitigate these challenges by reducing model size while preserving performance. However, the embedding layer, which occupies a significant portion of the model, remains a bottleneck in model compression efforts. In this paper, we valdated vocabulary reduction as a solution to compress the embedding layer and reduce model size without significant loss of performance. We conduct a series of experiments to investigate how vocabulary reduction affects GPU memory footprint, inference speed, and task performance. Our results show that while performance generally declines with vocabulary reduction, fine-tuning can recover much of the lost performance. Moreover, in some tasks, such as truthfulness and summarization, the vocabulary-reduced models outperform the baseline. Finally, we demonstrate that vocabulary reduction can be effectively applied in domain adaptation, particularly in the medical domain, and in multilingual adaptation, improving task efficiency and cross-lingual robustness.

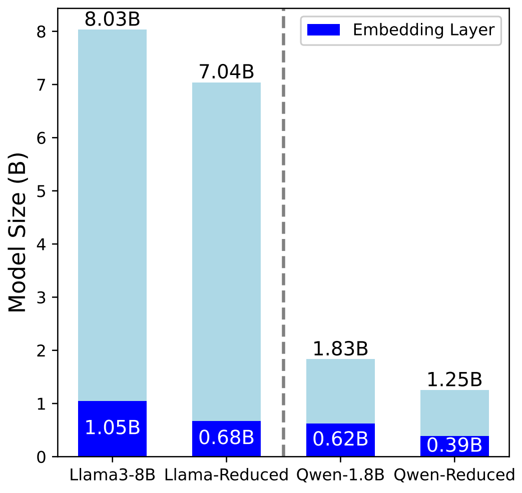

Figure 1: Model size reduction using the embedding layer compression method employed in this study. In the case of Llama3-8B, reducing the vocabulary to 8k results in a decrease of approximately 1B total parameters.

1 Introduction

The practical deployment of large language models (LLMs) has introduced significant challenges due to their computational costs and energy consumption (Stojkovic et al., 2024). While increasing model size generally leads to better performance (Kaplan et al., 2020), the high cost of inference makes it difficult to apply LLMs in real-world, industrial environments. These costs not only affect operational efficiency but also raise environmental concerns (Luccioni et al., 2024), leading to a growing demand for more efficient solutions.

To address these challenges, small language models (SLMs) (Lu et al., 2024) have been developed to reduce computational load while maintaining performance. Many SLMs achieve compression through distillation techniques, reducing the number of transformer blocks (Abdin et al., 2024; Dubey et al., 2024). However, the embedding layer, which constitutes a large portion of the model, often remains unchanged, creating a bottleneck in model compression and limiting the benefits of transformer block reduction.

As a preliminary experiment, we demonstrate that the smaller the model size, the higher the proportion of the embedding layer becomes. For instance, in LLMs such as Llama-3-70B (Dubey et al., 2024), the embedding layer represents only 3% of the model size, while in smaller models like Llama-3-8B, it accounts for 13%. In even smaller models, such as Qwen1.5-1.8B (Yang et al., 2024), this proportion reaches 34%. These results indicate that in smaller language models, the embedding layer makes up a constant portion of the overall model (Table **??**).

Model	Total Params	Embed Params	Embed Ratio
Llama-3-70B	70B	2.1B	3%
Llama-3-8B	8B	1.1B	13%
Qwen1.5-1.8B	1.8B	622M	34%

Table 1: Total and Embedding Parameters and Ratios for Different Models

In this paper, we explore a novel approach to mitigate this bottleneck through vocabulary reduction (Figure 1). In SLMs that employ Byte Pair Encoding (BPE) (Gage, 1994; Sennrich et al., 2016) for tokenizer construction, directly pruning vocabulary from source tokenizers is not feasible. Instead, we reconstruct the tokenizer with a smaller vocabulary and replace the corresponding embedding vectors. This method reduces the size of the embedding layer while also allowing the introduction of a more targeted vocabulary for specific tasks or domains.

This approach, while promising, presents trade-offs, such as potential declines in inference speed and task performance. Thus, our key research question is: how can we construct a smaller, efficient vocabulary without sacrificing performance? To address this, we focus on three key aspects:

- How does vocabulary reduction affect GPU memory footprint, inference speed, and task performance?
- What methods can mitigate the performance degradation caused by vocabulary reduction?
- What methods can mitigate the performance degradation caused by vocabulary reduction?

To address these questions, we conducted a series of experiments. First, we evaluated the impact of vocabulary reduction on GPU memory footprint (model size), and inference speed by reducing the vocabulary of Llama3-8B at various levels. We observed that memory footprint decreased steadily, but inference speed initially slowed before improving with further reductions.

Next, we evaluated task performance across four model variants: the source model, a vocabulary-reduced model, a vocabulary-reduced model fine-tuned on additional data, and a vocabulary-reduced model with task arithmetic applied. Overall, performance initially declined with vocabulary reduction, but fine-tuning helped recover accuracy in most tasks. In some cases, such as truthfulness evaluation, the models with reduced vocabularies and task arithmetic even matched or exceeded the performance of the baseline models.

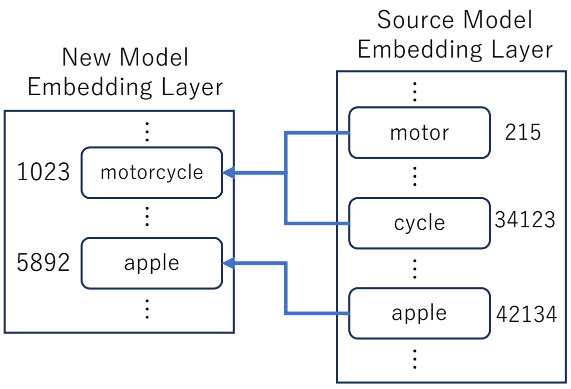

Figure 2: Process of generating embeddings for new tokens by combining subword tokens from the source model. For tokens not present in the source model (e.g., ’motorcycle’), the embeddings of existing subword tokens (’motor’ and ’cycle’) are averaged to generate a new embedding. This is made possible by the Byte-level BPE tokenization used in many SLMs, where the smallest unit of division is a byte, preventing unknown words.

Finally, we applied vocabulary reduction in both domain adaptation and multilingual adaptation settings. In the medical domain, we created tokenizers tailored to medical terminology and fine-tuned the Llama3 model on PubMed abstracts. Vocabulary reduction improved inference speed and memory footprint, but reducing the vocabulary size too much resulted in slower inference and lower accuracy, emphasizing the need to balance vocabulary size for optimal domain-specific performance. In multilingual adaptation, integrating vocabularies across languages demonstrated improved cross-lingual efficiency and robustness, further validating the versatility of vocabulary reduction.

2 Reduce Method

We describe the specific method used to reduce the vocabulary size. LLMs typically use tokenizers based on BPE. In BPE, vocabulary selection starts by splitting text into byte-level tokens, then repeatedly merging the most frequent token pairs until the predefined vocabulary size is reached. This method prioritizes high-frequency tokens, ensuring their inclusion in the tokenizer's vocabulary.

However, when using a BPE-based tokenizer, both a dictionary and merge rules are required. Thus, reducing the vocabulary size involves adjusting the merge rules. Manually altering these optimized rules risks disrupting the tokenization process and leading to suboptimal results. To avoid these complications, we construct a new tokenizer with a reduced and arbitrary vocabulary size. Normally, the embedding layer is closely tied to the tokenizer's vocabulary, and reducing the vocabulary would disrupt this mapping. However, based on Zipf's Law (Zipf, 1942), high-frequency tokens tend to remain consistent across different tokenizers and corpora, as a small number of tokens dominate text samples. This means that there will be a significant overlap between the source and new vocabularies. For tokens common to both the source and new tokenizers, the source embeddings are simply remapped. For tokens not present in the source model, new embeddings are generated by combining the embeddings of existing subword tokens and averaging them (Figure 2). This approach helps assign meaningful vectors to new tokens in the reduced vocabulary while aiming to maintain the model's performance and reduce the size of the embedding layer.

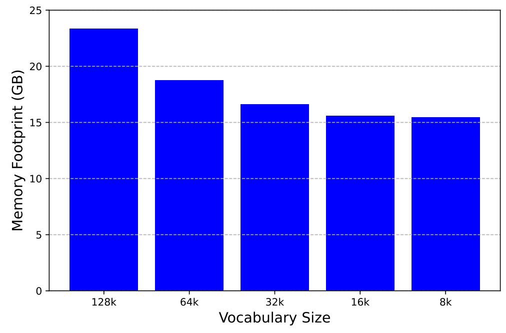

Figure 3: Graph of peak GPU memory footprint during inference on 5,000 summarization tasks. It shows that memory footprint decreases as the vocabulary size is reduced.

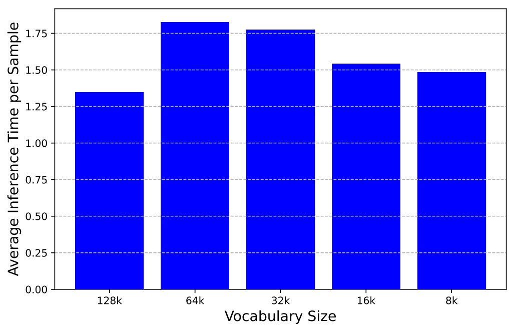

Figure 4: Graph of average inference time per sample during inference on 5,000 summarization tasks. Inference time increases when the vocabulary is reduced to 32k, but then begins to decrease, with only a 10% increase compared to the source at 8k.

3 Experiments

3.1 GPU Memory Footprint and Inference Time

To evaluate the impact of vocabulary reduction on memory footprint and inference time, we measured the model size in terms of actual GPU memory footprint during inference. We constructed tokenizers with reduced vocabulary sizes—64,128 (64k), 32,064 (32k), 16,032 (16k), and 8,016 (8k)—using a 1:1 mixture of Wikipedia (Guo et al., 2020) and C4 datasets (Raffel et al., 2020), with 128,256 (source) as a reference. Inference on 5,000 summary examples using one NVIDIA H100 GPU showed that reducing the vocabulary to 64k reduced the GPU memory footprint by 20%, and further reduction to 8k decreased it by 34% (Figure 3). This demonstrates significant memory savings when reducing vocabulary from the source size.

To assess the effect on inference time, we measured the average inference time per example at the same vocabulary levels. Reducing the vocabulary to 64k increased inference time by 35%, the longest observed duration. However, further reductions gradually improved inference times, with the 8k vocabulary only resulting in a 10% increase compared to the source model, showing an improvement from the initial 35% increase (Figure 4).

3.2 Task Performance

Next, we examine the impact of simple vocabulary reduction on downstream task performance using the Llama3-8B model. The experiments were conducted in three stages: (1) performance evaluation after vocabulary reduction (*-init), where we tested the models immediately after reducing the vocabulary without any fine-tuning; (2) performance after fine-tuning (*-ft), where we fine-tuned the vocabulary-reduced models using 1% of Wikipedia data (Guo et al., 2020); and (3) task arithmetic (*-ft-ta), where we applied task arithmetic to the fine-tuned models from stage (2) to transfer the instruction-tuning effects.

Model	CoLA	MNLI	MNLI-m	GSM8K	ARC	Hella Swag	MMLU	Truthful QA	XL Sum
Llama-3 (128k)	.471	.542	.538	.498	.590	.820	.651	.439	.905
Llama-3-Instruct	.432	.672	.665	.748	.623	.787	.657	.516	.887
8k-init	.259	.499	.503	.289	.398	.586	.616	.490	.872
8k-ft	.275	.491	.510	.347	.542	.773	.594	.436	.908
8k-ft-ta	.340	.628	.625	.569	.564	.735	.595	.521	.904
16k-init	.378	.545	.546	.382	.446	.656	.628	.449	.882
16k-ft	.428	.426	.427	.368	.549	.780	.597	.423	.907
16k-ft-ta	.390	.620	.620	.619	.577	.750	.600	.510	.904
32k-init	.419	.511	.511	.434	.516	.727	.635	.447	.902
32k-ft	.380	.537	.542	.379	.559	.789	.609	.432	.907
32k-ft-ta	.396	.632	.632	.639	.584	.758	.609	.512	.904
64k-init	.387	.535	.531	.463	.519	.750	.619	.436	.903
64k-ft	.422	.540	.547	.384	.565	.790	.600	.429	.903
64k-ft-ta	.386	.635	.630	.635	.584	.759	.604	.518	.901

Table 2: Task performance comparison of Llama3 and vocabulary-reduced variants (8k, 32k, 64k) at different stages: init (reduced), train (fine-tuned), and train-chat (fine-tuned with task arithmetic). Results are shown across benchmarks including **CoLA**, **MNLI**, **GSM8K**, **ARC**, **HellaSwag**, **MMLU**, **TruthfulQA**, and **XL-Sum**.

For evaluation, we selected downstream tasks commonly used in NLP benchmarks. Among them, we prioritized tasks that are especially important in industrial applications, focusing on general knowledge, common sense reasoning, recognition of logical relationships, truthfulness, and summarization.

3.2.1 Vocabulary Reduction Only (*-init)

First, the performance of each model was evaluated immediately after vocabulary reduction, without any further fine-tuning). As shown in Table 2, vocabulary reduction had various effects depending on the task. Performance degradation was relatively minimal across most tasks, even when the vocabulary was reduced to 32k. Tasks requiring broader contextual understanding, such as MNLI (Williams et al., 2018), **HellaSwag** (Zellers et al., 2019), and **XL-Sum** (Hasan et al., 2021), showed only slight decreases in performance with smaller vocabularies, suggesting reliance on contextual understanding over specific token granularity. However, tasks such as **CoLA** (Warstadt et al., 2019) and **GSM8K** (Cobbe et al., 2021), which require precise linguistic or logical structures, exhibited more significant performance drops. For instance, performance on **GSM8K** deteriorated sharply as the vocabulary was reduced to 16k tokens, indicating that this task is particularly sensitive to vocabulary size.

In knowledge-based tasks like **ARC** (Clark et al., 2018) and **MMLU** (Hendrycks et al., 2021), the impact of vocabulary reduction varied. **ARC** saw a consistent drop in performance as the vocabulary shrank, while **MMLU** maintained relatively stable performance until the vocabulary size dropped to 16k.

Interestingly, performance improved in **TruthfulQA** (Lin et al., 2022) after vocabulary reduction. This task tests the ability to avoid hallucinating false information, and limiting the vocabulary may have restricted the production of ambiguous or misleading tokens. This aligns with the "inverse scaling" concept (Lin et al., 2022), where larger models sometimes perform worse in tasks requiring truthfulness.

3.2.2 Fine-Tuning (*-ft)

Next, the vocabulary-reduced models were fine-tuned using 1% of the English Wikipedia dataset to re-optimize them. Fine-tuning helped recover much of the performance lost due to vocabulary reduction, particularly in tasks requiring contextual understanding (Table 2). For instance, in **MNLI**, **HellaSwag**, and **XL-Sum**, substantial performance recovery was observed, even with vocabularies reduced to 16k. This suggests that models can adapt to reduced vocabularies for tasks that depend more on understanding context rather than token precision. However, certain tasks like **CoLA** and **GSM8K** continued to struggle even after fine-tuning. These tasks rely heavily on the precise relationships between tokens, and fine-tuning alone could not fully compensate for the loss in token resolution caused by vocabulary reduction.

3.2.3 Task Arithmetic (*-ft-ta)

Finally, we explored the effectiveness of task arithmetic for the vocabulary-reduced models. Task arithmetic, as proposed by (Ilharco et al., 2023), involves taking the weight difference between an Instruct-tuned model and its base model, and applying this difference to the vocabulary-reduced models to transfer the instruction-tuning effects. This technique allows models to inherit instruction-following capabilities even after vocabulary reduction, without requiring the embedding layer to be retrained from scratch. Task arithmetic could be effectively applied between the Llama3-8B and Llama3-8B-Instruct models since, as is well-known, these models share the same embedding layer size. However, in our case, due to the difference in the size of the embedding layers after vocabulary reduction, it was not possible to take the difference for the embedding layer between the reduced model and the Instruct model. To address this, we conducted a preliminary experiment and found that the cosine similarity between the embedding layers of Llama3-8B and Llama3-8B-Instruct was nearly 1, indicating that the semantic representation of the embedding vectors remained almost identical. The only observed change was in the magnitude of the vectors. Therefore, we concluded that the embeddings of individual tokens did not undergo significant semantic changes between the base and Instruct models. Based on this finding, we applied task arithmetic by calculating the difference in all layers except for the embedding layer. By applying task arithmetic only to the transformer blocks and excluding the embedding layer, the resulting task arithmetic models (train-chat) generally achieved performance levels close to the Llama-3-Instruct model(Table 2). Notably, in the **TruthfulQA** task, the train-chat models performed almost on par with the Llama3-Instruct model, even with reduced vocabularies. This suggests that vocabulary reduction, combined with task arithmetic, can effectively limit the model's tendency to produce incorrect or false information, especially in tasks requiring precise handling of knowledge and truthfulness. While **TruthfulQA** showed significant improvement, other tasks, such as **MNLI** and **CoLA**, demonstrated more variable results, with the train-chat models not fully reaching the performance of Llama-3-Instruct in some cases.

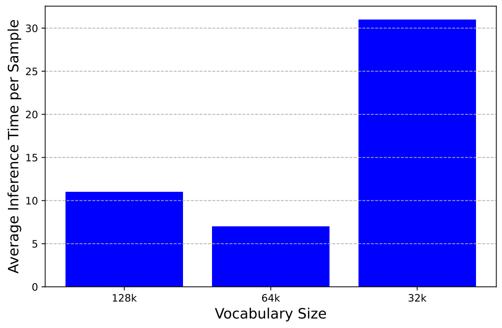

Figure 5: Graph of total inference time for 375 predictions on the **PubMedQA** task. Comparison of inference time per sample across models with different vocabulary sizes (64k, 32k, and the source model). The 64k model shows the best inference speed, while the 32k model, despite its improved task accuracy, experiences a noticeable slowdown.

3.3 Industry Application

We investigated practical applications of vocabulary reduction through experiments in medical domain adaptation and multilingual adaptation.

3.3.1 Medical Domain Adaptation

Model	PubMedQA
Llama-3 (source, 128k)	.730
source-128k-ft	.780
PubMed-64k-init	.738
PubMed-64k-ft	.780
PubMed-32k-ft	.746

Table 3: Vocabulary sizes and accuracies on **PubMedQA**. 'Llama-3 (128k)' is the source model; '-ft' denotes models fine-tuned on PubMed abstracts; 'PubMed-Xk' refers to models with vocabulary reduced to X thousand tokens.

One of the most practical applications of vocabulary reduction is in domain adaptation (Gururangan et al., 2020). In specialized fields such as the medical domain, where many domain-specific terms are prevalent, building a tokenizer tailored for that domain becomes crucial for effectively reducing inference costs and memory footprint while maintaining or even improving task performance. Additionally, domain-specific models do not always require high accuracy on general tasks, making them suitable for SLMs focused on efficiency.

In this experiment, we applied our vocabulary reduction technique to the medical domain by constructing tokenizers specifically tailored to medical terminology. Using the PubMed abstracts corpus (pub, 2024), we created two tokenizers with vocabulary sizes of 64k and 32k. These reduced tokenizers were then used to fine-tune the Llama3 model on the same PubMed abstracts corpus. To establish a baseline for comparison, we evaluated the source model and a version fine-tuned on the PubMed abstracts corpus without modifying its vocabulary.

The performance evaluation was conducted using the **PubMedQA** task (Jin et al., 2019). We measured two key metrics: the average inference time per example in the **PubMedQA** dataset and the task accuracy. The results showed that the vocabulary-reduced models demonstrated mixed outcomes in terms of inference time and task accuracy. The model with the 64k tokenizer achieved nearly the same task accuracy as the source Llama3 model fine-tuned directly on the medical domain, confirming that vocabulary reduction at this level had little negative impact on performance. However, the 32k tokenizer, while still outperforming the source Llama3 model in task accuracy, did not surpass the fine-tuned model that used the original, unmodified tokenizer (Table 3).

In terms of inference time, the vocabulary-reduced models performed differently. The 64k model demonstrated significant improvements in inference speed compared to the source model, confirming the efficiency benefits of moderate vocabulary reduction. However, the 32k model, despite showing better task accuracy than the source model, exhibited a significant slowdown in inference speed (Figure 5).

Overall, these findings confirm that vocabulary reduction can optimize task performance and inference efficiency, but only to a certain extent. While the 64k model maintained a good balance between performance and efficiency, the 32k model highlighted the trade-offs involved: although it improved accuracy relative to the source model, it did not match the fine-tuned model with the original tokenizer, and its inference speed was significantly slower. This suggests that reducing the vocabulary too much can negatively impact both task accuracy and processing efficiency, and there may be a threshold where the benefits of vocabulary reduction begin to diminish.

These results indicate that while vocabulary reduction is a viable solution for domain adaptation tasks, particularly in specialized fields like healthcare, careful consideration is required when choosing the level of reduction. A moderate reduction can lead to faster inference and competitive accuracy, but excessive reduction may compromise both performance and efficiency.

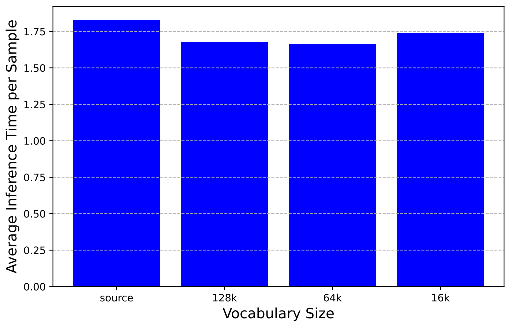

Figure 6: Graph of total inference time for 200 predictions on the **XLSum-JA** task. Vocabulary-reduced models, particularly the 64k model, achieved faster inference compared to the source model, likely due to the tokenizer's optimization for Japanese input.

3.3.2 Multilingual Adaptation

The multilingual adaptation of single-language models is a form of domain adaptation. Previous approaches (Wang et al., 2020; Pfeiffer et al., 2021) involve appending new vocabulary to the tokenizer and continuing pretraining, which increases model parameters and memory consumption. To address this, we propose applying vocabulary reduction to optimize multilingual models.

In this experiment, we constructed tokenizers integrating Japanese and English vocabularies using data from Japanese Wikipedia (Guo et al., 2020) and CC-100 (Conneau et al., 2020) (for Japanese) and English Wikipedia (Guo et al., 2020) (for English). The tokenizer was trained on data sampled at a 6:5 ratio of Japanese to English. The Llama3-8B model, initially specialized for English, was fine-tuned using a mixed dataset comprising 97% Japanese data and 3% English data to enhance its comprehension. Tokenizers with vocabulary sizes of 128k, 64k, and 16k were created and used for fine-tuning. For comparison, the original model was also fine-tuned on the same data without modifying its vocabulary.

Performance was evaluated using Japanese and English tasks, measuring task accuracy and average inference time. Results (Figure 6 and Table **??**) show that vocabulary reduction improved Japanese task performance and minimized forgetting on English tasks. In reading comprehension

Model	JSQuAD	NIILC	XL Sum-JA	ARC	Hella Swag	MMLU	Truthful QA	XL Sum-EN
Llama-3 (source, 128k)	.877	.396	.751	.590	.822	.651	.905	.440
source-128k-ft	.526	.349	.703	.514	.754	.544	.899	.425
128k-ja-en	.849	.474	.752	.519	.748	.540	.901	.427
64k-ja-en	.853	488	.752	.497	.743	.534	.904	.406
16k-ja-en	.864	.509	.752	.492	.727	.513	.898	.436

Table 4: Performance of Llama3-8B models with different vocabulary sizes on Japanese and English tasks.

(**JSQuAD**), smaller vocabularies achieved higher accuracy, while knowledge-intensive tasks like **NIILC** saw significant gains. Summarization tasks (**XLSum-JA**) remained stable, showing robustness to reduction. English tasks such as **ARC**, **HellaSwag**, and **MMLU** showed minimal degradation despite reduced English vocabulary and Japanese-focused fine-tuning.

Inference time analysis (Figure 6) revealed that both the 128k and reduced vocabulary models outperformed the source model in speed, likely due to the tokenizer's optimization for Japanese input. These findings confirm that vocabulary reduction is a practical solution for improving multilingual model performance and efficiency without increasing memory costs.

4 Related Works

Several techniques have been proposed to compress Transformer models, including distillation and pruning, which typically target intermediate layers. However, a substantial portion of the model parameters lies in the embedding layer, leading to recent efforts focused on compressing this component.

For example, (Cohn et al., 2023) introduced dynamic embeddings to reduce BERT's model size with minimal performance loss, while (Yu et al., 2024) explored character-level generation to remove long tokens in Chinese poetry models. Similarly, (Xue et al., 2022) developed a byte-level model to optimize vocabulary usage efficiently.

Our approach differs by constructing a tokenizer with a smaller, high-frequency vocabulary tailored to specific domains and reallocating the embedding layer accordingly. This method highlights the trade-off between vocabulary size and inference speed, particularly beneficial for SLMs where balancing compression and performance is crucial.

5 Conclusion and Limitations

In this paper, we investigated vocabulary reduction as a method for compressing the embedding layer of SLMs to enhance memory footprint and inference speed without compromising task performance. Our experiments confirmed that reducing the vocabulary size results in significant memory savings, and moderate reductions (e.g., 64k) provide a good balance between efficiency and accuracy across various tasks. In particular, for domain adaptation in the medical field, models with reduced vocabularies showed competitive task accuracy and improved inference speed. Additionally, in multilingual adaptation, vocabulary reduction enabled efficient integration of multiple languages, enhancing cross-lingual robustness while maintaining strong performance on language-specific tasks.

However, this study has some limitations. First, we did not investigate the underlying mechanisms behind the observed improvements in truthfulness evaluation and summarization tasks due to vocabulary reduction. Further research is needed to understand why reducing the vocabulary size led to better scores in these areas. Second, our industrial application was limited to the medical domain and Japanese adaptation. To confirm the general effectiveness of vocabulary reduction, further experiments are needed in other domains where specialized terminology is critical, such as legal tasks, as well as in other languages, including Chinese, French, and low-resource languages.

References

2024. Pubmed baseline dataset. `https://ftp.ncbi.nlm.nih.gov/pubmed/baseline/`. Accessed: 2024-8-07.

Marah Abdin et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Gabrielle Cohn, Rishika Agarwal, Deepanshu Gupta, and Siddharth Patwardhan. 2023. EELBERT: Tiny models through dynamic embeddings. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 451–459, Singapore. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2024. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Abhimanyu Dubey et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. 2020. Wiki-40B: Multilingual language model dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2440–2452, Marseille, France. European Language Resources Association.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Kentaro Kurihara, Daisuke Kawahara, and Tomohide Shibata. 2022. JGLUE: Japanese general language understanding evaluation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2957–2966, Marseille, France. European Language Resources Association.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5661–5681, Toronto, Canada. Association for Computational Linguistics.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. 2024. Small language models: Survey, measurements, and insights. *Preprint*, arXiv:2409.15790.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2024. Estimating the carbon footprint of bloom, a 176b parameter language model. *J. Mach. Learn. Res.*, 24(1).

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021. Unks everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Satoshi Sekine. 2003. Development of a question answering system focused on an encyclopedia. In *Proceedings of the 9th Annual Meeting of Japanese Association for Natural Language Processing (NLP2003)*, pages 637–640. In Japanese.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri, and Josep Torrellas. 2024. Towards greener llms: Bringing energy-efficiency to the forefront of llm inference. *Preprint*, arXiv:2403.20306.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

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*, 10:291–306.

An Yang et al. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Chengyue Yu, Lei Zang, Jiaotuan Wang, Chenyi Zhuang, and Jinjie Gu. 2024. CharPoet: A Chinese classical poetry generation system based on token-free LLM. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 315–325, Bangkok, Thailand. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

George Kingsley Zipf. 1942. The unity of nature, least-action, and natural social science. *Sociometry*, 5(1):48–62.

Appendix

A Fine-Tuning Settings and Hyperparameters

In our experiments, we used the following settings and hyperparameters for fine-tuning (see Table **??**).

Hyperparameter	**Value**
Global Batch Size (GBS)	32
Sequence Length	8192
Learning Rate (LR)	5e-5
Warmup Ratio	0.03
GPU	NVIDIA H100 (80GB)
Number of GPUs	8

Table 5: Training Hyperparameters

B Comparison with Quantization

In addition to our vocabulary reduction experiments, we evaluated the effectiveness of 8-bit quantization, a widely used method for reducing GPU memory consumption (Dettmers et al., 2024). Quantization reduces the precision of the model weights from 16-bit floating-point numbers to 8-bit integers, significantly decreasing memory usage while aiming to preserve model performance. To compare our vocabulary reduction method with quantization, we applied 8-bit quantization to the Llama-3 model and evaluated its performance on the same downstream tasks. Table **??** presents the results of the quantized model alongside the vocabulary-reduced models.

We examined whether our vocabulary reduction method is superior to the quantization method. Basically, in most tasks, the vocabulary reduction method performed worse than the quantized model. However, in **TruthfulQA** and **XL-Sum**, the vocabulary reduction method was superior.

C Qwen

C.1 Vocabulary Reduction on Qwen-1.5-1.8B

In addition to our experiments with Llama3, we also evaluated the impact of vocabulary reduction on another small language model, Qwen-1.5-1.8B (Yang et al., 2024), which has a significantly larger source vocabulary size of 151,936 tokens. This allowed us to assess whether our findings generalize to models with very large vocabularies.

We reduced the vocabulary of Qwen-1.5 to half (76k), one-quarter (38k), one-eighth (19k), and one-sixteenth (9.5k) of the source size using BPE tokenization on the English Wikipedia dataset. The performance of the vocabulary-reduced Qwen models was then evaluated on the same downstream tasks as before (Table **??**).

For Qwen-1.5, the impact of vocabulary reduction was more pronounced than in Llama3. In tasks like **MNLI**, accuracy decreased significantly with vocabulary reduction, suggesting that Qwen's performance on **MNLI** is more dependent on precise token distinctions. Conversely, tasks such as **HellaSwag**, which rely more on contextual reasoning, showed less sensitivity to the reduced vocabulary size.

Interestingly, Qwen also demonstrated improvements in the **TruthfulQA** task after vocabulary reduction. As with Llama-3, reducing the vocabulary may have limited the model's ability to generate ambiguous or misleading tokens, supporting the concept of "inverse scaling" in tasks that require truthfulness.

Fine-tuning the vocabulary-reduced Qwen models led to improvements in tasks like **HellaSwag** and **XL-Sum**, where contextual understanding is crucial. However, **MNLI** continued to show limited recovery even after fine-tuning, indicating that certain tasks in Qwen are particularly sensitive to token reduction.

C.1.1 Inference Time and GPU Memory Consumption

In addition to accuracy evaluation, we measured the impact of vocabulary reduction on inference time and GPU memory consumption for the Qwen-1.5 models. Similar to our observations with Llama3, reducing the vocabulary size led to a decrease in GPU memory footprint due to the smaller embedding layer (Figure 7).

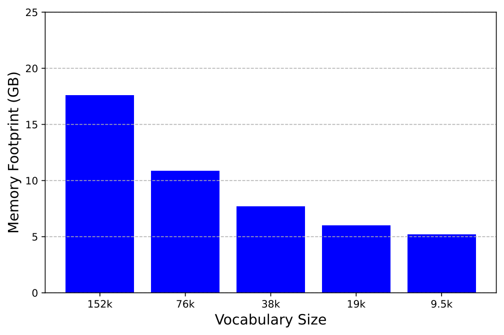

Figure 7: Graph of peak GPU memory footprint during inference on 5,000 summarization tasks.

Furthermore, we observed that the inference speed was slightly slower with the original 152k vocabulary size, but among the models with reduced vocabularies, the inference speed remained largely unchanged (Figure 8).

C.2 Overall

These findings indicate that vocabulary reduction can offer significant efficiency gains in terms of memory footprint for models with very large vocabularies like Qwen-1.5. However, the trade-offs between efficiency and task performance are more pronounced in Qwen compared to Llama3, particularly for tasks that rely heavily on precise token distinctions like **MNLI**. Therefore, careful consideration is required when applying vocabulary reduction to such models to ensure that efficiency gains do not come at the cost of unacceptable performance degradation in critical tasks. Overall, our experiments with Qwen-1.5 confirm that while vocabulary reduction is a generally applicable method for embedding layer compression, its impact varies depending on the model architecture and the specific tasks. Models with extremely large vocabularies may experience more significant performance drops in certain tasks, underscoring the need for task-specific evaluations when considering vocabulary reduction.

Model	CoLA	MNLI	MNLI-m	GSM8K	ARC	Hella Swag	MMLU	Truthful QA	XL Sum
Llama-3 (128k)	.471	.542	.538	.498	.590	.820	.651	.439	.905
Llama-3 (128k, 8bit)	.414	.541	.540	.491	.599	.821	.647	.432	.905
8k-init	.259	.499	.503	.289	.398	.586	.616	.490	.872
8k-ft	.275	.491	.510	.347	.542	.773	.594	.436	.908
16k-init	.378	.545	.546	.382	.446	.656	.628	.449	.882
16k-ft	.428	.426	.427	.368	.549	.780	.597	.423	.907
32k-init	.419	.511	.511	.434	.516	.727	.635	.447	.902
32k-ft	.380	.537	.542	.379	.559	.789	.609	.432	.907
64k-init	.387	.535	.531	.463	.519	.750	.619	.436	.903
64k-ft	.422	.540	.547	.384	.565	.790	.600	.429	.903

Table 6: Performance comparison between vocabulary reduction and 8-bit quantization across various tasks.

Model	CoLA	MNLI	MNLI-m	GSM8K	ARC	Hella Swag	MMLU	Truthful QA	XL Sum
Qwen-1.5 (152k)	.140	.463	.492	.344	.375	.615	.456	.394	.879
Qwen-1.5-chat	.138	.496	.519	.300	.390	.602	.445	.405	.
9.5k-init	.064	.362	.368	.239	.303	.441	.395	.429	.855
9.5k-ft	.073	374	.382	.175	.335	.523	.394	.417	.888
9.5k-ft-ta	.026	.384	.401	.208	.332	.503	.396	.420	.889
19k-init	.085	.364	.370	.266	.326	.402	.411	.433	.867
19k-ft	.026	.375	.385	.188	.329	.550	.412	.405	.892
19k-ft-ta	.026	.409	.422	.221	.341	.533	.415	.416	.892
38k-init	.038	.348	.362	.278	.334	.549	.413	.408	.855
38k-ft	.089	.362	.373	.227	.345	.567	.408	.403	.893
38k-ft-ta	.105	.361	.361	.243	.366	.551	.415	.409	.891
76k-init	.125	.485	.509	.220	.340	.541	.429	.406	.858
76k-ft	.064	.455	.475	.209	.350	.562	.430	.389	.893
76k-ft-ta	.044	.506	.509	.222	.371	.553	.423	.413	.885

Table 7: Performance of Qwen-1.5 models with different vocabulary sizes across various tasks.

D Evaluation of Vocabulary Reduction Using Characters per Token (CPT)

In the main text, we evaluated the impact of vocabulary reduction on inference speed to assess performance in practical deployment environments. However, there are other benchmarks that can provide additional insights into the effects of vocabulary reduction. One such metric is Characters per Token (CPT) (Limisiewicz et al., 2023), which measures token efficiency. To evaluate the impact of vocabulary reduction on token efficiency, we constructed tokenizers with vocabulary sizes ranging from 1,000 to 128,000 tokens, increasing in increments of 4,000.

As a specific experiment, we built a BPE tokenizer using a sample of the English Wikipedia dataset and calculated the CPT on test data. This experiment allowed us to systematically evaluate how vocabulary size affects token efficiency.

We used the metric Characters per Token (CPT) to measure token efficiency. The formula for CPT is defined as:

$$CPT = \frac{\text{Total Number of Characters}}{\text{Total Number of Tokens}}$$

A higher CPT indicates better token efficiency, as fewer tokens are used to represent more characters, while a lower CPT indicates worse efficiency because more tokens are required.

As shown in Figure 9, the results demonstrate that token efficiency improves rapidly as the vocabulary size increases up to approximately 32,000 tokens, after which the improvements begin to plateau. This suggests that beyond a certain vocabulary size, the benefits of additional tokens dimin-

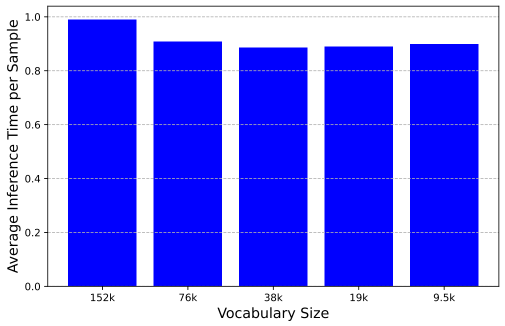

Figure 8: Graph of average inference time per sample during inference on 5,000 summarization tasks.

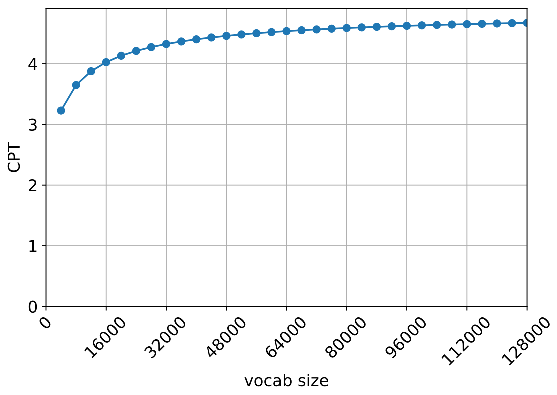

Figure 9: Characters per Token (CPT) as a function of vocabulary size. The graph shows that CPT increases rapidly up to a vocabulary size of around 32,000 tokens, after which the rate of improvement slows down significantly.

ish, and increasing the vocabulary further yields minimal gains in token efficiency.

This analysis complements our main findings by highlighting that while larger vocabularies can improve token efficiency up to a point, the trade-offs between vocabulary size and practical deployment considerations like inference speed and memory usage must be carefully balanced.

E Evaluated Tasks

We evaluated our models on several English language tasks that assess different aspects of linguistic understanding, reasoning, and knowledge application. All evaluations were conducted using the `lm-evaluation-harness` (Gao et al., 2024) toolkit. The number of few-shot examples and the evaluation metrics used for each task are specified below:

- **CoLA** (Corpus of Linguistic Acceptability): Tests the model's ability to judge the grammatical acceptability of English sentences, evaluating its understanding of linguistic rules and syntax (Warstadt et al., 2019). We used 4-shot prompting, and the performance was measured using the Matthews correlation coefficient (MCC).

- **MNLI** (Multi-Genre Natural Language Inference): Evaluates the model's ability to perform natural language inference across multiple genres. The task involves determining whether a given hypothesis is entailed by, contradicts, or is neutral with respect to a provided premise (Williams et al., 2018). We used 4-shot prompting, and accuracy was the evaluation metric.

- **GSM8K**: A dataset of grade school math word problems designed to test the model's mathematical reasoning and problem-solving skills (Cobbe et al., 2021). We used 5-shot prompting, and the performance was evaluated using the flexible extraction method.

- **TruthfulQA**: A benchmark designed to evaluate the model's ability to generate truthful and accurate answers, assessing its tendency to produce hallucinations or misinformation (Lin et al., 2022). We used zero-shot prompting, and the metric used was multiple-choice accuracy (`mc2`).

- **ARC** (AI2 Reasoning Challenge): Aimed at evaluating the model's science reasoning abilities at the middle school level. It requires the application of scientific knowledge and reasoning to select the correct answer from multiple choices (Clark et al., 2018). We used 25-shot prompting, and the normalized accuracy (`acc_norm`) was used as the evaluation metric.

- **HellaSwag**: Measures the model's ability to perform commonsense reasoning and understand context by selecting the most plausible continuation of a given situation (Zellers et al., 2019). We used 10-shot prompting, and the normalized accuracy (`acc_norm`) was used for evaluation.

- **MMLU** (Massive Multitask Language Understanding): Covers a wide range of knowledge domains and evaluates the model's ability to apply this knowledge accurately in answering questions (Hendrycks et al., 2021). We used 5-shot prompting, and accuracy was the evaluation metric.

- **XL-Sum**: A dataset for extreme summarization of news articles, where the model must create concise summaries that capture the essence of the articles. We used the English version (**XLSum-en**) for our evaluations (Hasan et al., 2021). We used 2-shot prompting, and the BERTScore was used as the evaluation metric.

- **PubMedQA**: A biomedical question-answering dataset designed to test the model's ability to comprehend and answer questions based on biomedical literature (Jin et al., 2019). We used 2-shot prompting, and accuracy was the evaluation metric.

- **JSQuAD**: A Japanese reading comprehension dataset derived from the SQuAD dataset, adapted for evaluating a model's ability to understand and answer questions based on Japanese texts (Kurihara et al., 2022). We used 2-shot prompting, and accuracy was the evaluation metric.

- **NIILC**: A dataset designed to test knowledge-based reasoning in Japanese. The model must apply its knowledge to answer questions spanning various topics (Sekine, 2003). We used 2-shot prompting, and accuracy was the evaluation metric.

- **XLSum-JA**: A Japanese version of the XL-Sum dataset for summarization tasks, where the model generates concise summaries of Japanese news articles (Hasan et al., 2021). We used 2-shot prompting, and the BERTScore was used as the evaluation metric.