VRCP: Vocabulary Replacement Continued Pretraining for Efficient Multilingual 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

Building large language models (LLMs) for non-English languages involves leveraging extensively trained English models through continued pre-training on the target language corpora. This approach harnesses the rich semantic knowledge embedded in English models, allowing superior performance compared to training from scratch. However, tokenizers not optimized for the target language may make inefficiencies in training. We propose Vocabulary Replacement Continued Pretraining (VRCP), a method that optimizes the tokenizer for the target language by replacing unique (solely available) vocabulary from the source tokenizer while maintaining the overall vocabulary size. This approach preserves the semantic knowledge of the source model while enhancing token efficiency and performance for the target language. We evaluated VRCP using the Llama-2 model on Japanese and Chinese corpora. The results show that VRCP matches the performance of vocabulary expansion methods on benchmarks and achieves superior performance in summarization tasks. Additionally, VRCP provides an optimized tokenizer that balances token efficiency, task performance, and GPU memory footprint, making it particularly suitable for resource-constrained environments.

1 Introduction

Recent advancements in large language models (LLMs) based on transformer architectures (Vaswani et al., 2017) have brought significant progress to the field of NLP. Models such as GPT-4 (OpenAI et al., 2023) and Llama-2 (Touvron et al., 2023) have predominantly been trained on extensive English corpora, leaving a gap in the availability of models optimized for non-English languages. This disparity is due to the relative scarcity of high-quality, large-scale corpora for many non-English languages compared to English. Consequently, this limits the potential improvements based on the scaling laws (Kaplan et al., 2020) of language models for these languages.

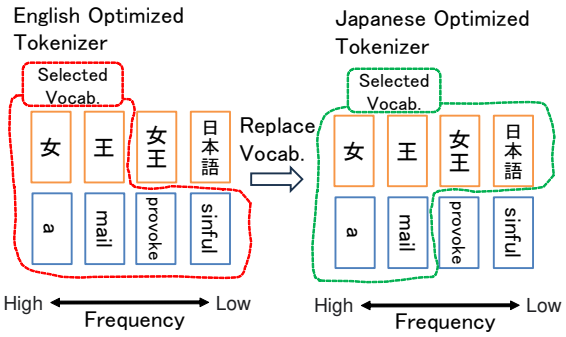

Figure 1: Illustration of VRCP: The English Optimized Tokenizer (left) replaces low-frequency English vocabulary with high-frequency Japanese vocabulary to create the Japanese Optimized Tokenizer (right). This retains common vocabulary while optimizing for Japanese.

One promising approach to developing models specialized for non-English languages involves continued pretraining of extensively pretrained models on the target language corpora (Yong et al., 2023; Wang et al., 2020; Pfeiffer et al., 2021). This approach leverages the semantic knowledge and intelligence of the English-based models, enabling high performance even with relatively small amounts of target language corpora. This approach, akin to DAPT (domain-adaptive pre-training) (Gururangan et al., 2020), allows for the incorporation of new linguistic characteristics while retaining the established knowledge base.

However, this approach presents significant challenges. The vocabulary of tokenizers used in English-based model is optimized for efficient tokenizing of English texts. When applied to texts in target languages, these tokenizers often segment the text into excessively small units. This leads to a considerable reduction in the length of sequences that can be processed in the same batch, thereby increasing the training time for the target

Proceedings of the 2nd SUMEval Workshop: Scaling Up Multilingual & Multi-Cultural Evaluation, pages 48–59
January 20, 2025.

language data and significantly decreasing training efficiency. This inefficiency is particularly problematic in scenarios involving Retrieval-Augmented Generation (RAG), where language models need to process long texts as prompts for seq2seq tasks, e.g., summarization (Lewis et al., 2021). Given the inherent limitations in sequence length imposed by transformer architectures, using a tokenizer that inefficiently for the target language restricts the amount of text that can be included in the prompt. Even when a model supports very long sequence lengths, the computational complexity increases at an $O(n^2)$ rate with the sequence length (Vaswani et al., 2017), leading to poor computational efficiency.

Additionally, languages with non-alphabetic scripts, such as Japanese and Chinese, often encounter issues with frequent out-of-vocabulary words or characters being fragmented at the byte level (Rust et al., 2021).

To address these inefficiencies, a previous approach is to expand the tokenizer's vocabulary with tokens relevant to the target language, reducing the number of tokens needed to represent the same text (Wang et al., 2020; Yao et al., 2021; Liu et al., 2020; Wang et al., 2019; Minixhofer et al., 2022). However, this method inadvertently increases the parameter count of the embedding layer of the model and, consequently, a lager memory footprint.

Maintaining a constant vocabulary size while optimizing for target languages is crucial for preventing an increase in model parameters and controlling memory consumption. This is particularly important for smaller models, where the embedding layer represents a significant portion of the overall model size. For instance, in models like Qwen-2 (0.5B and 1.5B), embedding tying is used to prevent an increase in model size (Yang et al., 2024). Moreover, works on machine translation models have shown that embedding tying can reduce model size while maintaining performance (Press and Wolf, 2017). Additionally, increasing the size of the embedding layer exacerbates communication overhead during distributed training, as highlighted in studies by (Acun et al.). Maintaining a constant vocabulary size allows us to mitigate these issues and improve training efficiency.

We propose a novel method, Vocabulary Replacement Continued Pretraining (VRCP), to enhance token efficiency for the target language in continued pretraining while maintaining the vocabulary size. Our method involves constructing a new tokenizer tailored to the target language and substituting unique (solely available) vocabulary from the source tokenizer with vocabulary from the target language. This enables continued pretraining that leverages the semantic knowledge of the existing model while improving token efficiency. As illustrated in Figure 1, VRCP is a simple method whereby low-frequency words from the target language corpus within the source tokenizer are replaced with high-frequency words from the same target language corpus.

We evaluated VRCP through experiments on Japanese and Chinese texts. The results demonstrated that VRPC matches the token efficiency and task performance of expanding vocabulary methods. By not increasing the model size, VRCP also prevents any additional GPU memory footprint. Notably, for summarization tasks, VRCP showed superior performance compared to previous methods. This indicates its suitability for use cases such as RAG.

Our main contributions are as follows:

- We propose a method, VRCP, to enhance token efficiency for the target language in continued pretraining while maintaining the vocabulary size.
- Through experiments with Japanese and Chinese, we demonstrated that VRCP can achieve token efficiency and task performance comparable to vocabulary expanding methods while maintain vocabulary size.
- We showed that VRCP improves summarization task performance, making it ideal for use cases such as RAG.

2 Formulization

2.1 Tokenizer Definition

A tokenizer performs two primary tasks: segmenting a text into tokens and mapping these tokens to unique IDs using its vocabulary V.

First, the segmentation function S processes a text $text$ and produces a sequence of tokens $\{t_1, t_2, \ldots, t_n\}$, where each token t_i belongs to V:

$$S(text, V) = \{t_1, t_2, \ldots, t_n\} \quad (1)$$

Then, the mapping function M assigns each token t_i an ID id_i within the range $\{0, 1, \ldots, |V| - 1\}$:

$$M(t_i, V) = id_i \quad (2)$$

2.2 Embedding Definition

Each token ID id_i is associated with an embedding vector from the embedding matrix E. This matrix E is of size $|V| \times d$, where $|V|$ is the vocabulary size and d is the dimensionality of the embedding vectors. For a token t_i, its embedding vector $\vec{e}_{id_i}$ is obtained by mapping t_i to its ID id_i using M, and then retrieving the corresponding vector from E:

$$\vec{e}_{id_i} = E[M(t_i, V)] \tag{3}$$

2.3 Vocabulary Expansion Method

To expand the vocabulary of a source tokenizer, a new vocabulary V' is constructed from the target language corpus. This new vocabulary V' is combined with the original vocabulary V to form an expanded vocabulary:

$$V_{\text{new}} = V \cup V' \tag{4}$$

For each new token v'_i in V', its embedding $\vec{e}_{id_{v'_i}}$ is computed by taking the arithmetic mean of the embeddings of its segmented subtokens. Specifically, if v'_i is segmented into $\{t_1, t_2, \ldots, t_n\}$, we first map these tokens to their IDs $\{id_1, id_2, \ldots, id_n\}$ using M, and then retrieve their embeddings $\vec{e}_{id_1}, \vec{e}_{id_2}, \ldots, \vec{e}_{id_n}$ from E. The new embedding vector $\vec{e}_{id_{v'_i}}$ is calculated as the arithmetic mean of these vectors:

$$\vec{e}_{id_{v'_i}} = \frac{1}{n} \sum_{j=1}^{n} \vec{e}_{id_j} \tag{5}$$

These new embeddings are then added to the original embedding matrix E, resulting in an updated matrix:

$$E' = \begin{cases} \vec{e}_{id} & \text{if } id \in V \\ \vec{e}_{id_{v'_i}} & \text{if } v'_i \in V' \end{cases} \tag{6}$$

3 Proposed Method: VRCP

Our proposed method, Vocabulary Replacement Continued Pretraining (VRCP), consists of four main components: Vocabulary Construction, Vocabulary Replacement, Embedding Replacement, and Continued Pretraining.

3.1 Vocabulary Construction

The first step of VRCP is to construct a new vocabulary specialized for the target language. We define the vocabulary size to be equal to that of the source tokenizer and develop the tokenizer using a corpus that combines both the target language and English. Including English corpus ensures that common vocabulary between the target language and English is retained, which helps in effectively utilizing the semantic knowledge of the source model.

3.2 Vocabulary Replacement

Next, we replace the unique vocabulary of the source vocabulary V with those from the constructed vocabulary V'. This process retains the token ID mappings for the common vocabulary between V and V', enabling the model to leverage its knowledge of source model effectively.

The process is carried out using the following steps and equations:

1. **Identifying Common Vocabulary**:

   Identify the common vocabulary between the source vocabulary V and the constructed vocabulary V' by taking their intersection:

   $$V_{\text{com}} = V \cap V' \tag{7}$$

2. **Identifying Unique Vocabulary**:

   Determine the unique vocabulary in V' that are not present in V by taking the difference:

   $$V'_{\text{uni}} = V' \setminus V \tag{8}$$

3. **Constructing the New Vocabulary**:

   Form the new vocabulary V_{new} by combining the common vocabulary V_{com} and the unique vocabulary from V', V'_{uni}:

   $$V_{\text{new}} = V_{\text{com}} \cup V'_{\text{uni}} \tag{9}$$

To preserve the integrity of token IDs, for any vocabulary $v_{\text{com}} \in V_{\text{com}}$, the token ID in the new vocabulary V_{new} remains the same:

$$M(v_{\text{com}}, V_{\text{new}}) = M(v_{\text{com}}, V) \tag{10}$$

This equality holds because v_{com} is a part of the common vocabulary V_{com} and thus its token ID does not change with the new vocabulary V_{new}. By preserving these mappings, the model retains its knowledge associated with the shared tokens, enabling effective utilization of existing knowledge while adapting to new language nuances.

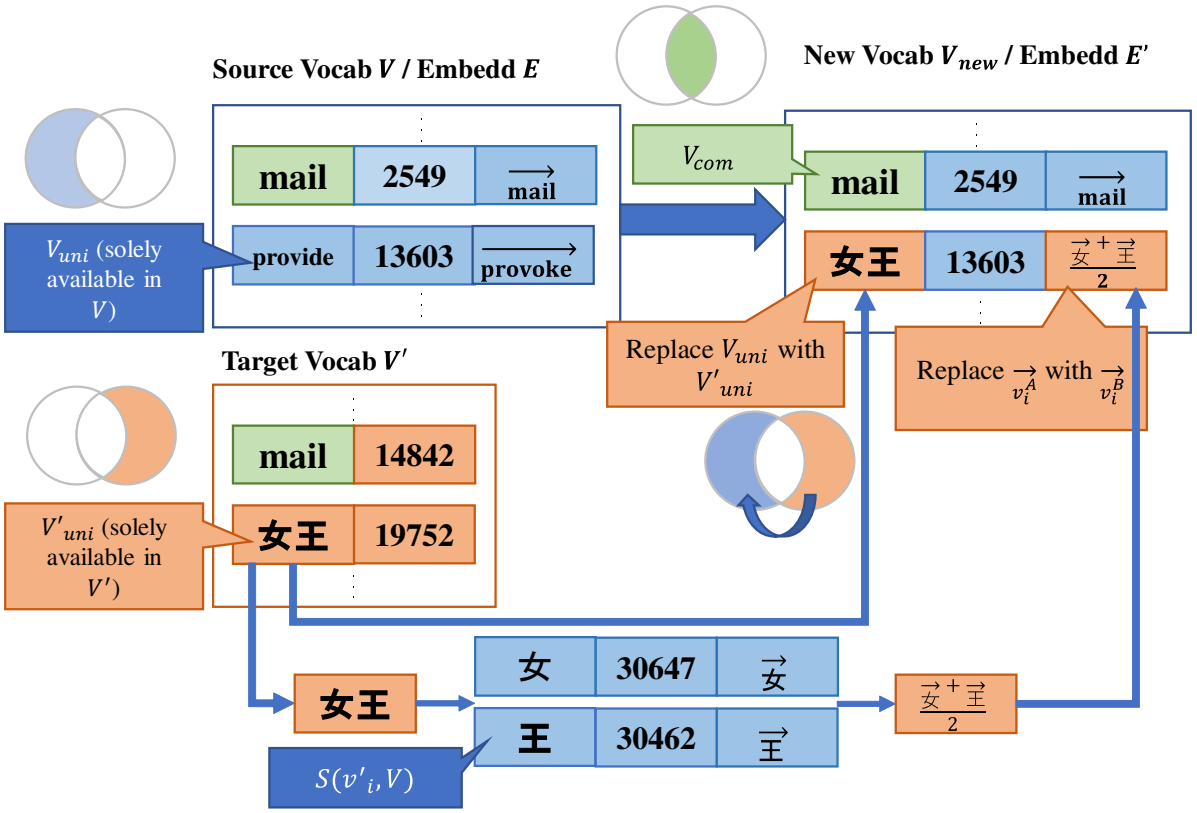

Figure 2: VRCP Process: Source vocabulary V and target vocabulary V' are used to create a new vocabulary V_{new}. Common tokens V_{com} are retained, while unique tokens V_{uni} from V are replaced with unique tokens V'_{uni} from V'.

3.3 Embedding Replacement

Replacing the vocabulary alone does not ensure that the embedding vectors for the unique vocabulary V' (V'_{uni}) align with the embeddings E. This is because the unique vocabulary $v'_i \in V'_{\text{uni}}$ have embedding vectors $\vec{e}_{id_{v'_i}}$ that may not fit well with the source embedding space. To address this, we need to replace these embeddings to maintain semantic consistency. Specifically, the following relation holds:

1. **Token Segmentation and ID Mapping**:

   For each token v'_i in the constructed vocabulary V', use the source tokenizer S and the vocabulary V to segment v'_i and map it to a sequence of token IDs $\{id_1, id_2, \ldots, id_n\}$:

   $$\{id_1, id_2, \ldots, id_n\} = M(S(v'_i, V), V) \tag{11}$$

   where each id_j corresponds to the token t_j in the vocabulary V.

2. **Retrieving Source Embedding Vectors**:

   Retrieve the corresponding embedding vectors $\vec{e}_{id_j}$ from the source embedding matrix E for each token ID id_j:

   $$\vec{e}_{id_j} = E[id_j] \tag{12}$$

3. **Calculating the Arithmetic Mean Embedding Vector**:

   Compute the arithmetic mean of the embedding vectors $\vec{e}_{id_j}$ for all sub-tokens t_j and create the embedding vector $\vec{e}_{id_{v'_i}}$ for the constructed vocabulary v'_i:

   $$\vec{e}_{id_{v'_i}} = \frac{1}{n} \sum_{j=1}^{n} \vec{e}_{id_j} \tag{13}$$

4. **Updating the Embedding Matrix**:

   After calculating the embedding vectors $\vec{e}_{id_{v'_i}}$ for the constructed vocabulary V', update the source embedding matrix E to form the new embedding matrix E':

   $$E' = \begin{cases} \vec{e}_{id} & \text{if } id \in V \\ \vec{e}_{id_{v'_i}} & \text{if } v'_i \in V'_{\text{uni}} \end{cases} \tag{14}$$

   This update ensures that the new embedding matrix E' remains semantically consistent with the existing embedding matrix E.

3.4 Continued Pretraining

In VRCP, the continued pretraining process primarily utilizes the target language corpus. However, to enhance the stability and effectiveness of training, we include a small portion of English corpus, aligning with the Domain-Adaptive Pretraining (DAPT) (Gururangan et al., 2020) strategy.

This inclusion follows approach of DAPT, where maintaining a minor amount of English data helps to mitigate the risk of catastrophic forgetting and prevents abrupt changes in the data distribution from destabilizing the model adaptation process.

4 Experiments

We evaluated the effectiveness of our proposed method using experiments with Japanese and Chinese corpora.

We evaluated the methods based on three key aspects:

- **Token Efficiently (Target/English)**
  - We evaluated each method by tokenizing the test data and measuring the average character length per token (Length Per Token, LPT) of the resulting token sequence.
- **Task performance (Target/English)**
  - We evaluated the models trained with each tokenizer method on benchmark tasks to assess their performance.
- **GPU Memory Footprint**
  - We measured the average GPU memory footprint per device in training.

4.1 Setup

Preparation of Corpora: We prepared mixed corpora of the target language and English for both Japanese and Chinese, ensuring a balanced ratio of 6:5 between the target language and English. The details of the corpora used are as follows:

- **Japanese**: Wikipedia (Japanese), CC100 Conneau et al. (2020) (Japanese), Wikipedia (English)
- **Chinese**: NLP Chinese Corpus (Xu, 2019) (Baike, News, Wikipedia), Wikipedia (English)

Construction of Tokenizer: We built a tokenizer with 32,000 words (V') using SentencePiece (Kudo and Richardson, 2018) and BPE (character_coverage=0.9995, Byte Fallback=True) from the mixed corpus of the target language and English, prepared as described in the previous section. Subsequently, we applied VRCP to replace the unique vocabulary (V_{uni}) from the Llama-2 tokenizer with the unique vocabulary (V'_{uni}) derived from the vocabulary constructed using this mixed corpus.

To evaluate the effectiveness of VRCP, we compared it against three other tokenizer approaches:

- $V \cup V'$ ($|V'| = 16k$): In this approach, we expanded the Llama-2 tokenizer (V) by adding 16,000 vocabulary constructed exclusively from a corpus in the target language.
- $V \cup V'$ ($|V'| = 32k$): In this approach, we expanded the Llama-2 tokenizer (V) by adding 32,000 vocabulary constructed exclusively from a corpus in the target language, similar to the previous approach.
- **Unchanged Llama-2 tokenizer** (V): In this approach, we used the Llama-2 tokenizer as-is, without any replacements or additions.

Continued Pretraining: We conducted further pretraining of Llama-2-7B using a mixed corpus consisting of both the target language and English. For this process, we used the same types of corpora as those used for the construction of Tokenizer, but with a different sampling strategy. Specifically, the ratio of English data in the training dataset was adjusted to under 5%. We describe the detailed training settings in the appendix.

4.1.1 Result and Discussion on Token Efficiency

VRCP significantly improved token efficiency for both Japanese and Chinese compared to the Llama-2 tokenizer, achieving approximately 2.2 times better efficiency for Japanese and 1.8 times better for Chinese. This is a major achievement of our study, particularly because we impose a constraint of not expanding the vocabulary size (see Tables 1 and 2 for detailed efficiency metrics).

When comparing VRCP to the vocabulary expansion method with $|V'| = 16,000$ and $|V'| = 32,000$, we find that VRCP includes more target language vocabulary. Although VRCP shows

Table 1: Tokenization Efficiency for Japanese

Methods	Vocab Size	JA Vocab Size	Common Vocab Size	EN LPT	JA LPT
$V_{\text{com}} \cup V'_{\text{uni}}$ (VRCP)	32,000	15,293	12,137	3.740	1.838
$V \cup V'$ ($\|V'\| = 16k$)	46,312	14,553	32,000	3.738	1.883
$V \cup V'$ ($\|V'\| = 32k$)	61,701	29,273	32,000	**3.758**	**2.081**
Llama 2	32,000	837	32,000	3.523	0.851

Table 2: Tokenization Efficiency for Chinese

Methods	Vocab Size	ZH Vocab Size	Common Vocab Size	EN LPT	ZH LPT
$V_{\text{com}} \cup V'_{\text{uni}}$ (VRCP)	32,000	13,476	12,756	**3.767**	1.257
$V \cup V'$ ($\|V'\| = 16k$)	46,073	14,065	32,000	**3.767**	1.445
$V \cup V'$ ($\|V'\| = 32k$)	61,191	28,284	32,000	3.736	**1.580**
Llama 2	32,000	700	32,000	3.523	0.705

slightly lower token efficiency compared to the vocabulary expansion methods, the Length Per Token (LPT) is almost equivalent or better, especially for English. Even with $|V'| = 32,000$, the improvement in LPT is not significantly greater compared to the efficiency improvement observed when replacing the unique vocabulary in the Llama-2 tokenizer with VRCP's vocabulary (refer to Table 1 and Table 2).

Overall, we emphasize that substantial improvement of VRCP on token efficiency for the target languages, achieved without expanding the vocabulary size, remains highly competitive against vocabulary expansion methods. This suggests that excluding non-target languages from the tokenizer enhances tokenization efficiency for both the target language and English, providing a balanced and efficient approach for multilingual tokenization (see Tables 1 and 2 for a summary of results).

4.2 Evaluation of Task Performance

We evaluated the performance on benchmark tasks using models pretrained with each tokenizer method:

- **English**: ARC, HellaSwag, MMLU, XLSum-EN
- **Japanese**: JCommonsenseQA, JSQuAD, NIILC, XLSum-JA
- **Chinese**: C-Eval, CMMLU, CMRC, XLSum-ZH

4.2.1 Discussion on Task Performance for Japanese and Chinese

Japanese: Overall, the Unchanged method showed the highest average performance across tasks (see Table 3). This may be explained by the fact that the Llama-2 model has been extensively trained with its tokenizer on a vast amount of data, optimizing the model for this tokenizer. This phenomenon has also been reported in previous studies. However, when excluding the Unchanged method, VRCP demonstrated the best performance among the methods tested.

In particular, in summarization tasks, VRCP significantly improved performance compared to the Unchanged method (see Table 3). This improvement was especially notable in the version of VRCP without embedding replacement, which performed better than the vocabulary expansion methods. In fact, the more we expanded the total vocabulary size, the lower the performance tended to be. This may be because modifying the embedding vectors for vocabulary expansion can negatively impact text generation tasks. Even if the initial values of the embeddings do not align perfectly with the meaning of the vocabulary, it has been shown that maintaining these initial embeddings can be advantageous for text generation tasks.

Regarding English tasks, VRCP showed slightly lower performance compared to the vocabulary expansion methods. However, the decrease in performance was not severe enough to suggest a breakdown of the model. Notably, the decrease in performance for English summarization tasks was less pronounced than for other tasks. This indicates that the extensive training of Llama-2 has made the model robust to some changes in the tokenizer, especially for text generation tasks, e.g., summarization (see Table 3 for performance metrics).

Chinese: Similar trends were observed for Chinese tasks. VRCP performed best in summarization tasks, with particularly strong results in the version without embedding replacement. The pattern of performance decreasing as the vocabulary size expanded was also noted in Chinese, indicating

Table 3: Performance for Different Methods (Japanese)

Type	Method	EN		JA		GPU Memory Footprint (GB)
		Avg.	XLSum-EN	Avg.	XLSum-JA	
Vocab Replace **(VRCP)**	$V_{\text{com}} \cup V'_{\text{uni}}$	.627	.900	.672	.734	**52.78**
	$V_{\text{com}} \cup V'_{\text{uni}}$ (Without Embed Replace)	.617	.897	.637	**.737**	**52.78**
Vocab Expand **(Previous)**	$V \cup V'$ ($\lvert V' \rvert = 16k$)	**.643**	**.901**	.668	.736	54.57
	$V \cup V'$ ($\lvert V' \rvert = 32k$)	.640	**.901**	.658	.717	56.16
Unchanged **(Baseline)**	V	.626	.900	**.691**	.712	**52.78**
Vanilla	Llama 2-7B	.670	.905	.591	.690	N/A

Table 4: Performance for Different Methods (Chinese)

Type	Method	EN		ZH		GPU Memory Footprint (GB)
		Avg.	XLSum-EN	Avg.	XLSum-ZH	
Vocab Replace **(VRCP)**	$V_{\text{com}} \cup V'_{\text{uni}}$	.622	**.902**	**.507**	.625	**52.78**
	$V_{\text{com}} \cup V'_{\text{uni}}$ (Without Embed Replace)	.616	**.902**	.502	**.652**	**52.78**
Vocab Expand **(Previous)**	$V \cup V'$ ($\lvert V' \rvert = 16k$)	.639	**.902**	.493	.605	54.51
	$V \cup V'$ ($\lvert V' \rvert = 32k$)	**.642**	.901	.488	.596	57.35
Unchanged **(Baseline)**	V	.633	.900	.483	.553	**52.78**
Vanilla	Llama 2-7B	.670	.905	.499	.619	N/A

that expanding the vocabulary may reduce performance, similar to what was seen with Japanese tasks. Additionally, the slightly lower performance in English tasks when using VRCP was observed in both Japanese and Chinese settings, but again, the decrease was not severe enough to compromise the model's effectiveness (see Table 4 for detailed results).

Summary of Task Performance: The experiments indicate that the vocabulary size in the target language does not necessarily impact performance. Both VRCP and the vocabulary expansion methods showed similar average scores across tasks. This suggests that expanding the vocabulary is not always essential to achieve high performance. Instead, the approach of VRCP, which involves replacing unique vocabulary without expanding the total vocabulary size, remains competitive and effective, especially in text generation tasks. Additionally, VRCP's improvement in English summarization tasks supports the benefit of vocabulary replacement for task performance in the target language as well as in English (see Tables 3 and 4 for task performance comparisons).

4.3 Evaluation of GPU Memory Footprint

We evaluated the GPU memory footprint of each method in experiments with Japanese and Chinese. VRCP maintains the same vocabulary size as the Llama-2 model, ensuring a consistent GPU memory footprint. This indicates its effectiveness in resource-constrained environments without the need for extensive vocabulary expansion (refer to Table 3 and Table 4 for memory footprint details).

Vocabulary expansion methods increased memory footprint. Specifically, as the vocabulary size increased, the GPU footprint increased linearly. This result indicates that extensive vocabulary expansions may not be efficient or necessary for improving performance (see Figure 3 for a visual representation of the linear increase in memory footprint).

As shown in Figure 3, even though increasing the vocabulary size does not significantly improve the Length Per Token (LPT), the GPU memory foot-

print continues to increase linearly. This demonstrates that increasing vocabulary size leads to a predictable linear increase in memory footprint, without a corresponding substantial improvement in tokenization efficiency.

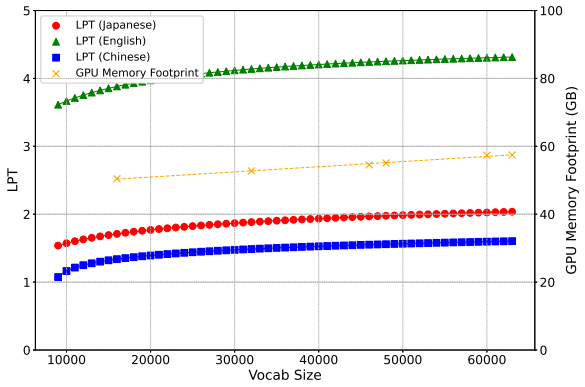

Figure 3: Relationship between vocabulary size, LPT, and GPU memory footprint.

5 Related Works

Several studies have explored enhancing model performance for low-resource languages by expanding vocabulary and embedding layers in continued pretraining. For example, Wang et al. (2020) and Pfeiffer et al. (2021) expanded the vocabulary and embedding layers for mBERT (Pires et al., 2019), which improved performance by incorporating low-resource language corpora. These methods typically involve expanding the vocabulary size, which increases the number of model parameters and the GPU memory footprint. This is because expanding the vocabulary requires adding corresponding embedding vectors. In decoder-only models like Llama-2 (Touvron et al., 2023), embeddings must be placed both after the input and before the output. Therefore, expanding the vocabulary by N_{new} tokens with an embedding dimension of D results in an increase of $2{\times}D{\times}N_{\text{new}}$model parameters. In contrast, our proposed method maintains the source vocabulary size by focusing on vocabulary replacement. This preserves the semantic knowledge of the source model while optimizing for the target language.

Limisiewicz et al. (2023) highlight the significant impact of tokenization on multilingual models, especially the importance of language-specific token coverage for word-level tasks. Their study provides guidelines for selecting tokenizers before expensive model pre-training.

Our research aligns with these insights by addressing the optimization of tokenizers for multilingual models without expanding the vocabulary size. This is particularly beneficial in resource-constrained environments. We provide an alternative strategy that leverages source model knowledge while adapting to the target language, complementing the guidelines suggested by Limisiewicz et al. (2023).

6 Conclusion

We introduced Vocabulary Replacement Continued Pretraining (VRCP), a method that optimizes tokenizers for non-English languages without increasing vocabulary size. VRCP replaces low-frequency English tokens with high-frequency target language tokens, leveraging the semantic knowledge of English models while enhancing token efficiency for the target language.

Our experiments with Japanese and Chinese corpora demonstrated that VRCP matches the performance of traditional vocabulary expansion methods and excels in tasks requiring text generation, such as summarization. This improved performance in generation tasks suggests that VRCP can be effectively applied to language models integrated into retrieval-augmented generation (RAG) frameworks, where generating coherent and contextually accurate summaries is critical. Additionally, VRCP avoids additional GPU memory costs by maintaining the original vocabulary size, making it suitable for resource-constrained environments.

Limitations

One limitation of our work is its language specificity. Our experiments were conducted only on Japanese and Chinese, meaning that the findings may not necessarily generalize to other languages. Since each language has unique characteristics, applying VRCP to other languages may require further adjustments and validation.

Additionally, while VRCP aims to prevent an increase in GPU memory consumption by maintaining a constant vocabulary size, modern distributed training techniques already provide efficient memory management solutions. Frameworks like TensorParallel, PipelineParallel (Narayanan et al., 2021), and ZeRO (Rajbhandari et al., 2020)) offer alternative or complementary strategies for managing resource constraints in large-scale model training.

References

Bilge Acun, Matthew Murphy, Xiaodong Wang, Jade Nie, Carole-Jean Wu, and Kim Hazelwood. Understanding training efficiency of deep learning recommendation models at scale. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 802–814. IEEE.

François Chollet. 2019. On the measure of intelligence. *Preprint*, arXiv:1911.01547.

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.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. A span-extraction dataset for chinese machine reading comprehension. 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 5883–5889, Hong Kong, China. Association for Computational Linguistics.

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*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, jiayi lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*, volume 36, pages 62991–63010. Curran Associates, Inc.

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.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024. Cmmlu: Measuring massive multitask language understanding in chinese. *Preprint*, arXiv:2306.09212.

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.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Benjamin Minixhofer, Fabian Paischer, and Navid Rekabsaz. 2022. WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3992–4006, Seattle, United States. Association for Computational Linguistics.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference*

for High Performance Computing, Networking, Storage and Analysis, SC '21, New York, NY, USA. Association for Computing Machinery.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. 2023. Gpt-4 technical report.

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.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, and Dong Yu. 2019. Improving pre-trained multilingual model with vocabulary expansion. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 316–327, Hong Kong, China. Association for Computational Linguistics.

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.

Bright Xu. 2019. Nlp chinese corpus: Large scale chinese corpus for nlp.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. 2021. Adapt-and-distill: Developing small, fast and effective pretrained language models for domains. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 460–470, Online. Association for Computational Linguistics.

Zheng Xin Yong, Hailey Schoelkopf, Niklas Muennighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, , et al. 2023. Bloom+1: Adding language support to bloom for zero-shot prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11682–11703, Toronto, Canada. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Choi Yejin. 2019. Hellaswag: Can a machine really finish your sentence? pages 4791–4800.

A Training Settings and Hyperparameters

In our experiments, we used the same settings and hyperparameters for continued pretraining across

all configurations, including VRCP, Vocabulary Expansion and Unchanged models (see Table 5).

Table 5: Training Hyperparameters

Hyperparameter	Value
Global Batch Size (GBS)	256
Sequence Length	4096
Learning Rate (LR)	7.5e-5
Warmup Ratio	0.05
Weight Decay	0.1
DeepSpeed ZeRO Stage (Rajbhandari et al., 2020)	2
AllGather Bucket Size	7e8
Reduce Bucket Size	7e8
GPU	NVIDIA H100 (80GB)
Number of GPUs	8

B Corpora Details

We used different corpora for Japanese and Chinese pretraining, as detailed in Table 6 and Table 7.

Table 6: Japanese Corpora Details

Corpus	Size (MB)
Wikipedia (Japanese)	3,178.95
CC100 (Japanese)	10,405.93
Wikipedia (English)	374.87

Table 7: Chinese Corpora Details

Corpus	Size (MB)
Wikipedia (Chinese)	1,115.09
Baike	1,186.11
News	5,837.07
Wikipedia (English)	374.87

C Evaluation Tasks

We evaluated the models on various benchmark tasks for Japanese, Chinese, and English. Each task focuses on different aspects of language understanding and generation, as summarized in Table 8, Table 9, and Table 10.

For the Japanese evaluation, we utilized the **JCommonsenseQA**, **JSQuAD**, **NIILC**, and **XLSum-JA** datasets.

Table 8: Japanese Evaluation Tasks

Task	Shots	Metric
JCommonsenseQA	4	Exact Match (EM)
JSQuAD	4	Character-level F1
NIILC	4	Character-level F1
XLSum-JA	1	BERTScore

Table 9: Chinese Evaluation Tasks

Task	Shots	Metric
C-Eval	4	Accuracy
CMMLU	4	Accuracy
CMRC	4	Accuracy
XLSum-ZH	1	BERTScore

Table 10: English Evaluation Tasks

Task	Shots	Metric
ARC	25	Normalized Accuracy
HellaSwag	10	Normalized Accuracy
MMLU	5	Accuracy
XLSum-EN	1	BERTScore

- **JCommonsenseQA** evaluates common sense reasoning abilities in Japanese. This dataset is part of the JGLUE benchmark and provides questions that require the model to use background knowledge to choose the correct answer from multiple choices (Kurihara et al., 2022).
- **JSQuAD** is a question-answering task also included in the JGLUE benchmark. It focuses on extracting answers from provided contexts based on Japanese text (Kurihara et al., 2022).
- **NIILC** (National Institute of Informatics Large-scale Encyclopedia Corpus) presents open-ended questions, where the model must generate answers using knowledge embedded within the model. This task assesses the model's encyclopedic knowledge and its ability to produce accurate responses (Sekine, 2003).
- **XLSum-JA** is a summarization task that requires the model to generate concise summaries from Japanese news articles (Hasan et al., 2021).

For evaluating Chinese language capabilities, we

used the **C-Eval**, **CMMLU**, **CMRC**, and **XLSum-ZH** datasets.

- **C-Eval** includes tasks for reading comprehension, text generation, and reasoning based on various domains of knowledge, such as literature, history, science, and technology (Huang et al., 2023).
- **CMMLU** (Massive Multitask Language Understanding in Chinese) encompasses a set of tasks across multiple domains, including comprehension, text generation, classification, translation, and dialogue (Li et al., 2024).
- **CMRC** (Chinese Machine Reading Comprehension) focuses on question-answering by extracting answers from given contexts (Cui et al., 2019).
- **XLSum-ZH** is the Chinese counterpart of the summarization task for news articles, where the model generates brief summaries from longer articles (Hasan et al., 2021).

The evaluation for English language tasks was conducted using the **ARC**, **HellaSwag**, **MMLU**, and **XLSum-EN** datasets, which test various aspects of knowledge and reasoning:

- **ARC** (AI2 Reasoning Challenge) is designed to assess middle school level science reasoning abilities. The dataset includes questions that require the model to apply scientific knowledge and reasoning skills to select the correct answer from multiple choices (Chollet, 2019).
- **HellaSwag** measures ability of the models to perform contextual and common sense reasoning (Zellers et al., 2019).
- **MMLU** (Massive Multitask Language Understanding) covers a wide range of knowledge domains and evaluates the model's capability to apply this knowledge in answering questions accurately (Hendrycks et al., 2021).
- **XLSum-EN** is the English version of the summarization task, where the model must create concise summaries from news articles (Hasan et al., 2021).