Large Vocabulary Size Improves Large Language Models

Anonymous ACL submission

Abstract

This paper empirically investigates the relationship between subword vocabulary size and the performance of large language models (LLMs) to provide insights on how to define the vocabulary size. Experimental results show that larger vocabulary sizes lead to better performance in LLMs. Moreover, we consider a continual training scenario where a pre-trained language model is trained on a different target language. We introduce a simple method to use a new vocabulary instead of the pre-defined one. We show that using the new vocabulary outperforms the model with the vocabulary used in pre-training.

1 Introduction

Since the GPT series demonstrated that Large Language Models (LLMs) excel in complex reasoning tasks (Radford et al., 2018a,b; Brown et al., 2020), they have rapidly become indispensable tools for various natural language processing tasks. To construct better LLMs, previous studies have addressed theoretical analyses of internal layers (Xiong et al., 2020; Takase et al., 2024) and conducted extensive experiments to provide empirical findings (Kaplan et al., 2020; Hoffmann et al., 2022; Wortsman et al., 2024). For example, Hoffmann et al. (2022) reported the compute-optimal training configuration, which determines suitable parameter and training data sizes for a given computational resource.

In contrast, although previous studies have explored the properties of internal layers in LLMs, parameters related to the vocabulary, the embedding and output layers, are under-explored. Specifically, there are no well-established findings on how to determine the subword vocabulary size, which defines the parameter size of the embedding and output layers. As a standard strategy, a vocabulary size in the 30k-60k range is used for monolingual LLMs (Radford et al., 2018b; Brown et al., 2020; Black et al., 2022; Zhang et al., 2022; Touvron et al., 2023), while around 250k is used for multilingual LLMs (Chowdhery et al., 2022; Le Scao et al., 2022). For monolingual LLMs, a larger vocabulary size has been discussed in terms of efficiency during the inference phase (Almazrouei et al., 2023). However, the question remains: does a larger vocabulary size offer any advantages for the quality of monolingual LLMs? To address this question, we empirically investigate the relationship between vocabulary size and performance on downstream tasks.

We conduct experiments on two languages: English, which is widely used, and Japanese, which is character-rich. We show that a larger vocabulary size improves the performance of LLMs in both languages. In addition to training from scratch, we consider the continual training scenario. When adapting a pre-trained LLM to another language, it may be beneficial to reconstruct an appropriate vocabulary instead of reusing the original vocabulary. For this purpose, we propose a strategy to swap parameters related to the vocabulary. We demonstrate that using the reconstructed vocabulary can improve performance.

2 Vocabulary Construction

To construct subword vocabularies, there are two widely used algorithms: Byte-Pair Encoding (BPE) (Sennrich et al., 2016) and unigram language model (Kudo, 2018). In this study, we use the unigram language model implemented in SentencePiece (Kudo and Richardson, 2018). For each language, we use the following vocabulary sizes: 5k, 10k, 50k, 100k and 500k.

We conduct experiments on two languages: English and Japanese. For the English training data, we extract English corpora from SlimPajama (Soboleva et al., 2023), excluding the book corpus, which was reported to have copyright infringement issues. For the Japanese training data, we extract the Japanese portion of CommonCrawl corpus with the language identification and document deduplication applied using CCNet (Wenzek et al., 2020). For the vocabulary construction, we sample a small portion (50GB) from each language training data.

3 Experiments on Vocabulary Size

3.1 Settings

To investigate the relationship between vocabulary size and performance, we train Transformer-based language models on the training data described in Section 2. Table 1 shows the number of tokens in the training data calculated from each vocabulary set. As shown in this table, the number of tokens varies drastically based on the vocabulary size. Therefore, we must take care not to give any unfair advantages to any setting.

For example, with a fixed number of training tokens, the 500k vocabulary model trains for around 1.5 epochs in English and 2 epochs in Japanese, while the 5k vocabulary model trains for only 1 epoch. The larger vocabulary size has an advantage of seeing more data in this configuration. In contrast, with a fixed number of training epochs, the 5k vocabulary model consumes much more computational resources than the larger vocabulary models. Especially in Japanese, where the 5k vocabulary model contains about twice as much tokens as the 500k vocabulary model in 1 epoch. Because the performance of LLMs is correlated with the computational costs during training (Kaplan et al., 2020), this configuration might favor smaller vocabulary sizes. Thus, we prepare two training configurations: 1T tokens and 1 epoch[1].

For hyper-parameters of the language model, we use the GPT-3 Large setting described in Brown et al. (2020). We set the number of layers 24 and the hidden dimension size 1536. In this setting, the number of parameters for internal layers is 680M. We use Megatron-LM (Shoeybi et al., 2020)[2] as our codebase to train large language models. To stabilize the training, we use the scaled embed technique (Takase et al., 2024).

#Vocab	English	Japanese
5k	830B	950B
10k	750B	750B
50k	670B	590B
100k	650B	550B
500k	640B	490B

Table 1: The number of tokens in training data tokenized by each vocabulary.

We evaluate each model on the commonsense reasoning tasks. For English, we use PIQA (Bisk et al., 2020), OpenBookQA (OBQA) (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021) and ARC easy and challenge (Clark et al., 2018). For Japanese, we use JSQuAD and JCommonsenseQA (JCQA) from JGLUE (Kurihara et al., 2022), the Japanese portion of XWinograd (Tikhonov and Ryabinin, 2021), and JAQKET[3]. Following the previous study (Touvron et al., 2023), we use the normalized likelihood in evaluation (Brown et al., 2020; Gao et al., 2023).

3.2 Results

Tables 2 and 3 present the performance of the models trained with 1T tokens and 1 epoch. For each configuration, we show the average score of each task, and the improvement of the average score from the 5k vocabulary model.

As shown by the average scores, for both English and Japanese, larger vocabulary sizes lead to better performance. The improvement is particularly notable in Japanese, largely due to the gains in JAQKET. Unlike the other tasks where the model selects answers from provided candidates, JAQKET is a factoid QA task where the model generates answers without any candidates. This suggests that a larger vocabulary size particularly benefits generation tasks.

In addition, the larger vocabulary size achieves better performance in either situation where we fix the number of training tokens or training epochs. With a fixed number of epochs, the larger vocabulary size settings, e.g., 100k and 500k, use a much smaller number of training tokens (Table 1). This means that the larger vocabulary size also improves the training efficiency because we can obtain a better model with a smaller computational

[1] In addition to the training data size, we have to discuss the number of parameters for a fair comparison because the model with the small vocabulary size contains less parameters for the embedding and output layers. However, as described in Appendix D, the model with the small vocabulary size does not improve the performance when we increase the number of parameters related to the vocabulary. Thus, we focus only on varying the training data size in our main experiments.

[2] https://github.com/NVIDIA/Megatron-LM

[3] https://sites.google.com/view/project-aio/competition1

#Vocab	PIQA	OBQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Avg.
1T tokens							
5k	69.9	33.2	51.0	55.2	49.6	27.7	47.8 (±0.0)
10k	71.2	33.4	51.5	55.2	50.6	27.1	48.2 (+0.4)
50k	**71.7**	32.8	53.9	54.5	50.8	27.7	48.6 (+0.8)
100k	70.9	33.4	53.9	54.8	54.3	27.7	49.2 (+1.4)
500k	71.4	**34.0**	**55.3**	**57.5**	**55.1**	**28.3**	**50.3** (+2.5)
1 Epoch							
5k	70.1	32.4	50.9	**55.2**	50.2	**28.5**	47.9 (±0.0)
10k	71.1	33.6	50.6	55.7	49.0	27.1	47.9 (±0.0)
50k	70.6	33.6	52.1	53.8	52.3	27.3	48.3 (+0.4)
100k	**71.7**	33.8	53.4	54.7	52.7	27.6	49.0 (+1.1)
500k	70.4	**34.2**	**54.3**	55.1	**54.0**	28.2	**49.4** (+1.5)

Table 2: The performance on English commonsense reasoning tasks in training 1T tokens and 1 epoch.

#Vocab	JSQuAD	JCQA	XWinograd	JAQKET	Avg.
1T tokens					
5k	58.1	68.1	58.9	12.5	49.4 (±0.0)
10k	61.2	67.2	59.0	23.3	52.7 (+3.3)
50k	61.8	71.6	59.0	29.2	55.4 (+6.0)
100k	62.1	**71.9**	**59.6**	34.9	57.1 (+7.7)
500k	**64.5**	71.6	59.3	**38.9**	**58.6** (+9.2)
1 Epoch					
5k	57.7	68.1	58.8	14.4	49.8 (±0.0)
10k	57.7	63.4	**60.0**	22.0	50.8 (+1.0)
50k	60.9	69.1	58.5	28.7	54.3 (+4.5)
100k	61.3	**70.1**	58.7	31.0	55.3 (+5.5)
500k	**63.2**	69.8	57.7	**34.1**	**56.2** (+6.4)

Table 3: The performance on Japanese commonsense reasoning tasks in training 1T tokens and 1 epoch.

cost. In fact, the GPU hours[4] in the 100k setting are 0.7 times shorter than in the 5k when we fix the number of training epochs in Japanese[5].

4 Experiments on Continual Training

4.1 Increasing Vocabulary Size

Section 3 shows that the larger vocabulary size is useful in constructing LLMs from scratch. In contrast, nowadays, we often start from a high-quality pre-trained model such as the Llama series (Touvron et al., 2023) and continue training on the target language data (Müller and Laurent, 2022; Yong et al., 2023; Yamada and Ri, 2024).

Here, we check if we can readily increase the vocabulary size from the pre-trained model. Similar techniques have been explored as vocabulary expansion (Fujii et al., 2024; Kim et al., 2024) or sophisticated embedding initialization using cross-lingual word embeddings (Minixhofer et al., 2022), but our focus here is to check if we could increase the vocabulary size in a rather simplistic way. We consider a situation where we construct an entirely new vocabulary independently of the original vocabulary.

Let V_{orig} and V_{new} be the vocabulary set of the pre-trained model and a newly constructed vocabulary set respectively, and let d be the dimension size of each layer. To exploit knowledge learned in the pre-trained embedding matrix, we construct a new embedding matrix $E_{new} \in \mathbb{R}^{|V_{new}|\times d}$ from the original embedding matrix $E_{orig} \in \mathbb{R}^{|V_{orig}|\times d}$ with the way inspired by the randomized algorithm (Halko et al., 2011):

$$E_{new} = \frac{WE_{orig}}{\sqrt{|V_{orig}|}}, \qquad (1)$$

where $W \in \mathbb{R}^{|V_{new}|\times|V_{orig}|}$ is the random matrix whose elements are sampled from the standard normal distribution independently. To maintain the standard deviation of E_{orig} in E_{new}, we scale the matrix multiplication by $\frac{1}{\sqrt{|V_{orig}|}}$[6].

[4] We used A100 80GB for all experiments.

[5] Since the larger vocabulary size slows the computation of the output distribution, we should use an efficient way such as the adaptive softmax (Grave et al., 2017) in practice.

[6] We assume that E_{orig} contains independent random variables with mean 0 and variance $\text{var}(E_{orig})$. Then, the variance of the matrix multiplication WE_{orig} has mean 0 and variance $\text{var}(E_{orig}) \times |V_{orig}|$.

Setting	#Vocab	JSQuAD	JCQA	XWinograd	JAQKET	Avg.
From scratch	100k	71.8	76.0	63.6	54.2	66.4
Llama2 (w/o train)	32k	71.2	60.8	62.4	15.3	52.4 ($\pm$0.0)
Llama2 vocab	32k	80.7	79.4	**72.6**	47.7	70.1 (+17.7)
Swap	100k	79.2	**80.2**	67.5	56.3	70.8 (+18.4)
Swap&Insert	100k	**81.9**	**80.2**	69.2	**61.2**	**73.1** (+20.7)
Fujii et al. (2024)	100k	81.6	77.6	69.1	61.1	72.4 (+20.0)

Table 4: The performance on Japanese commonsense reasoning tasks in the continual training from Llama2.

In the naive way, we swap E_{new} with E_{orig}. However, Equation 1 randomizes embeddings even if V_{new} contains the corresponding subwords which may possess useful knowledge transferable to the new model. Therefore, we insert the pre-trained embedding in E_{orig} into E_{new} if the corresponding subword is included in both V_{orig} and V_{new}[7]. For the output layer, we construct a new weight matrix with the same manner.

4.2 Results

We train the Llama2 7B parameter model (Touvron et al., 2023) with 100B tokens on our Japanese training data. We use the Japanese vocabulary whose size is 100k. Table 4 shows results on Japanese commonsense reasoning tasks. In this table, 'Swap' uses new parameters related to the vocabulary without inserting the corresponding pre-trained parameters. We train a language model from scratch to compare the effectiveness of the continual training. Moreover, we compare the embedding initialization method by Fujii et al. (2024) because their study is the same situation: continual training of Llama2 on Japanese data.

Table 4 shows that 'Swap' and 'Swap&Insert' outperform the model using the original Llama2 vocabulary even though these settings randomize parameters related to the vocabulary. This result indicates that it is better to prepare an appropriate vocabulary even in the continual training situation. Moreover, the insertion strategy achieves further improvement. The 'Swap&Insert' outperforms the method of Fujii et al. (2024), which initializes an embedding of the new subword with the average of the pre-trained embeddings[8], and thus, the 'Swap&Insert' is simple but effective.

5 Related Work

Before the paradigm of subword units and LLMs, researchers sometimes needed to handle the large vocabulary size such as more than 100k to decrease the number of unknown words. For example, the vocabulary sizes of One Billion Word Benchmark and WikiText-103 are about 800k and 300k respectively (Chelba et al., 2013; Merity et al., 2017). Some previous studies reported that character-level information was useful for neural language models with the large vocabulary size (Jozefowicz et al., 2016; Takase et al., 2019). In this paradigm, Chen et al. (2019) explored the impact of the vocabulary size.

Since the use of subword units is proposed (Sennrich et al., 2016; Kudo, 2018), the vocabulary sizes 30k-60k are widely used as the magic numbers (Libovický et al., 2022). As examples, the BERT and GPT papers use 30k and 40k for their vocabulary sizes respectively without any justification (Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2018a). Kiyono et al. (2020) investigated the relation between the performance and the vocabulary size but the maximum vocabulary size of their investigation is too small, i.e., 32k.

For large language models, the vocabulary sizes 30k-60k are also frequently used (Radford et al., 2018b; Touvron et al., 2023). In using the large vocabulary size, the authors claim to support multilinguality (Le Scao et al., 2022; Xue et al., 2021) or improve the efficiency (Lieber et al., 2021; AI@Meta, 2024). In contrast, we investigate the relation between the vocabulary size and the performance of monolingual LLMs on each task.

6 Conclusion

In this paper, we empirically investigate the performance of monolingual LLMs when we vary the vocabulary size. We conduct experiments on two languages: English and Japanese. Experimental results show that the larger vocabulary size is, the better performance the language model achieves in both languages. Moreover, we introduce a method to use the entirely new vocabulary in the continual training situation. We show that using the appropriate vocabulary also improves the performance in the continual training.

[7]See Appendix B for more details.

[8]For the existing subwords, their method uses the pre-trained embeddings. Thus, their method is regarded as using the 'Insert' strategy.

Limitations

In this study, we conducted experiments on two languages: English and Japanese. We believe that our findings can be applied to other languages because we do not depend on linguistic features in the subword vocabulary construction. However, we also agree that it is better to conduct exhaustive experiments on various languages to confirm the generality of our findings.

In this study, we used 500k as the maximum vocabulary size. Because it is impractical to construct a much larger vocabulary than 500k, we could not investigate the improvement by the tremendously large vocabulary size such as one million and the upper bound of the performance. The computational time of the vocabulary construction depends on the corpus size and the desired vocabulary size. We roughly estimate that the vocabulary whose size is larger than one million requires at least over a month in its construction in our environment.

Furthermore, the parameter sizes of internal layers are 680M in training from scratch, and 7B in the continual training. We consider that the discussions on subword vocabulary size are orthogonal to the parameter size of internal layers, but we would conduct additional experiments with more than 10B parameters if we had a large amount of computational resources.

References

AI@Meta. 2024. Llama 3 model card.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7432–7439.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 1877–1901.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Yang Wang. 2019. How large a vocabulary does text classification need? a variational approach to vocabulary selection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 3487–3497.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.

Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. 2024. Continual pre-training for cross-lingual llm adaptation: Enhancing japanese language capabilities. *Preprint*, arXiv:2404.17790.

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. 2023. A framework for few-shot language model evaluation.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for GPUs. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1302–1310.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. 2022. Training compute-optimal large language models. *ArXiv*, abs/2203.15556.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

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.

Seungduk Kim, Seungtaek Choi, and Myeongho Jeong. 2024. Efficient and effective vocabulary expansion towards multilingual large language models. *Preprint*, arXiv:2402.14714.

Shun Kiyono, Takumi Ito, Ryuto Konno, Makoto Morishita, and Jun Suzuki. 2020. Tohoku-AIP-NTT at WMT 2020 news translation task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 145–155, Online. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 66–75.

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 (EMNLP)*, pages 66–71.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. 2022. What language model to train if you have one million GPU hours? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 765–782.

Jindřich Libovický, Helmut Schmid, and Alexander Fraser. 2022. Why don't people use character-level machine translation? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2470–2485.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation. Technical report, AI21 Labs.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2381–2391.

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.

Martin Müller and Florian Laurent. 2022. Cedille: A large autoregressive French language model. *ArXiv*, abs/2202.03371.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018a. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018b. Language models are unsupervised multitask learners.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

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 (ACL)*, pages 1715–1725.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-lm: Training multi-billion parameter language models using model parallelism.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2024. Spike no more: Stabilizing the pre–training of large language models.

Sho Takase and Sosuke Kobayashi. 2020. All word embeddings from one embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 3775–3785.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2019. Character n-gram embeddings to improve rnn language models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 5074–5082.

Alexey Tikhonov and Max Ryabinin. 2021. It's All in the Heads: Using Attention Heads as a Baseline for Cross-Lingual Transfer in Commonsense Reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3534–3546.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 5998–6008.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC)*, pages 4003–4012.

Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie E Everett, Alexander A Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-Dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. 2024. Small-scale proxies for large-scale transformer training instabilities. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 10524–10533.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 483–498.

Ikuya Yamada and Ryokan Ri. 2024. Leia: Facilitating cross-lingual knowledge transfer in language models with entity-based data augmentation. *Preprint*, arXiv:2402.11485.

Zheng Xin Yong, Hailey Schoelkopf, Niklas Muennighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, Genta Winata, Stella Biderman, Edward Raff, Dragomir Radev, and Vassilina Nikoulina. 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 (ACL)*, pages 11682–11703.

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 (ACL)*, pages 4791–4800.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Hyper-parameter	Value
Number of layers	24
Hidden dimension size	1536
Number of attention heads	16
Sequence length	2048
Batch size	2048
Learning rate	3e-4
Learning rate scheduler	Cosine
Warmup ratio	0.01
Adam β_1	0.9
Adam β_2	0.95
Weight decay	0.01
Gradient clipping	1.0

Table 5: Hyper-parameters used in experiments described in Section 3.

Initialization	Vocab type	JSQuAD	JCQA	XWinograd	JAQKET	Avg.
Fujii et al. (2024)	Expansion	78.8	63.5	63.6	50.1	64.0
Swap&Insert	Expansion	80.7	60.6	67.4	55.9	66.2
Fujii et al. (2024)	Appropriate	81.6	77.6	69.1	61.1	72.4
Swap&Insert	Appropriate	**81.9**	**80.2**	**69.2**	**61.2**	**73.1**

Table 6: The performance on Japanese commonsense reasoning tasks in the continual training from Llama2 when we construct the 100k vocabulary with the vocabulary expansion approach and construct the appropriate 100k vocabulary to the Japanese training data.

Type	Number
UTF-8 byte pieces	256
Alphabet & number (e.g., a, the, 1)	5349
Symbol (e.g., +, =, ##)	209
Others such as Japanese characters	1083
Total	6897

Table 7: The type and number of shared subword units between the original Llama2 vocabulary and appropriate vocabulary, whose size is 100k, to the Japanese data in the continual training.

A Hyper-parameters

Table 5 shows hyper-parameters used in our main experiments described in Section 3.

B Formula of 'Insert' in Section 4.1

We formulate the procedure of 'Insert' in Section 4.1. Let e_i^{orig} and e_i^{new} be the i-th row vectors of E_{orig} and E_{new}, and let w_i^{orig} and w_i^{new} be the corresponding subwords to e_i^{orig} and e_i^{new}. The 'Insert' function, $\mathrm{Insert}(\cdot)$, replaces e_i^{new} with e_i^{orig} when the corresponding subword is included in the original vocabulary V_{orig} as follows:

$$\mathrm{Insert}(e_i^{new}) = \begin{cases} e_j^{orig} & \text{if } w_i^{new} \in V_{orig} \wedge w_j^{orig} = w_i^{new} \\ e_i^{new} & \text{otherwise} \end{cases} \quad (2)$$

Therefore, the matrix contains both the randomized embeddings and the original pre-trained embeddings after the 'Insert' procedure. As shown in Section 4.2, this procedure leads to further improvement.

C Comparison on Vocabulary Expansion in Continual Training

In Section 4, we conduct the continual training experiment on the scenario where we construct an appropriate vocabulary to the target language. In this scenario, most subword units in the original vocabulary might be removed. In contrast, the vocabulary expansion approach maintains the whole original vocabulary because it only adds new subword units to the original vocabulary (Fujii et al., 2024). We investigate which approach is empirically better in this section.

#Vocab	Vocab #Params.	Total #Params.	JSQuAD	JCQA	XWinograd	JAQKET	Avg.
5k	8M	690M	58.1	68.1	58.9	12.5	49.4 ($\pm$0.0)
5k w/ Expansion	200M	880M	61.0	60.3	59.5	15.8	49.2 ($-$0.2)
100k	150M	840M	**62.1**	**71.9**	**59.6**	**34.9**	**57.1** (+7.7)

Table 8: The performance on Japanese commonsense reasoning tasks when we use 1T tokens for training. For a fair comparison between 5k and 100k, we increase the parameter sizes of the embedding and output layers (Vocab #Params. in this Table) for 5k with the matrix factorization technique (Lan et al., 2020).

We construct 100k vocabulary with the vocabulary expansion approach, and compare it with the appropriate vocabulary used in Section 4. We apply two strategies to initialize the embedding matrix: Fujii et al. (2024) and our 'Swap&Insert'. Table 6 shows results of the continual training from Llama2. This table indicates that using appropriate vocabulary outperforms the vocabulary expansion approach. The appropriate vocabulary contains more subword units of the target language. We consider that this property improves the performance.

Table 7 shows the shared subword units between the original Llama2 vocabulary and the appropriate vocabulary. This table indicates that the number of shared subword units is only about 7000, which is about one-fifth of the original vocabulary. Moreover, this table suggests that the original vocabulary contains few Japanese subword units because the number of the shared Japanese characters is about 1000. Therefore, it is better to construct an entirely new vocabulary that is appropriate to the target language.

For the embedding initialization methods, Table 6 shows that our 'Swap&Insert' achieves better averaged score than the method of Fujii et al. (2024) in the same as the results in Section 4. Thus, our approach is also more suitable in the vocabulary expansion situation.

D Comparison on Parameter Size

The smaller vocabulary size lessens the parameter sizes related to the vocabulary in comparison with the larger vocabulary size. Thus, the smaller vocabulary size might have the disadvantage in the number of parameters. To confirm this point, we increase the parameters related to the vocabulary for the 5k setting. Concretely, we expand the dimension of the embedding and output layers, and then modify the dimension size by the linear transformation such as the matrix factorization technique (Lan et al., 2020)[9]. Let $|V|$ be the vocabulary size, d_e be the dimension size of the embedding and output layers, and d be the hidden dimension size. We prepare the expanded embedding layer $E \in \mathbb{R}^{|V| \times d_e}$ and the trainable weight matrix $W \in \mathbb{R}^{d_e \times d}$. We convert the dimension size of E with the matrix multiplication EW. For the output layer, we convert the dimension with the same manner. We adjust $d_e = 30720$ for a fair comparison with the 100k setting in terms of the number of parameters. For other hyper-parameters, we use the values shown in Table 5.

Table 8 shows the performance on Japanese commonsense reasoning tasks. This table indicates that the 5k with the expansion does not improve the average score although it increases the number of parameters. This result suggests that the increase of the parameter size related to the vocabulary has no positive influence on the performance. In contrast, the 100k achieves much better average score in the similar parameter size. Therefore, the improvement by the increase of the vocabulary size is orthogonal to the increase of the parameter size.

E Experiments on Each Training Data Size

In addition to the 1T tokens in Section 3, we investigate the performance in other training data sizes: 10B, 50B, 100B, 200B, and 500B tokens. Tables 9 and 10 show the results of English and Japanese models when we use each training data size. These tables show that larger vocabulary sizes lead to better performance for both English and Japanese in all training data sizes in the same as the results in Section 3. These tables indicate that our findings are independent from the amount of training data.

[9]In contrast, we can reduce the number of parameters for the larger vocabulary size with the matrix factorization technique or more sophisticated way (Takase and Kobayashi, 2020), but we regard the 5k as the baseline in this experiment.

#Vocab	PIQA	OBQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Avg.
			10B tokens				
5k	58.4	25.4	29.3	51.9	34.3	22.3	36.9 (±0.0)
10k	59.1	27.8	29.6	**53.2**	35.0	21.6	37.7 (+0.8)
50k	62.1	26.2	29.5	49.9	38.7	21.9	38.0 (+1.1)
100k	**62.2**	**27.8**	29.7	49.6	**39.0**	22.7	38.5 (+1.6)
500k	62.1	27.6	**30.1**	51.3	38.7	**22.9**	**38.8** (+1.9)
			50B tokens				
5k	66.7	28.0	39.0	**52.3**	41.9	23.8	41.9 (±0.0)
10k	66.3	30.4	39.5	51.0	42.6	25.3	42.5 (+0.6)
50k	68.1	29.4	40.9	50.9	46.9	25.5	43.6 (+1.7)
100k	68.1	31.6	42.0	51.3	46.9	25.5	44.2 (+2.3)
500k	**68.8**	**32.2**	**43.1**	52.0	**47.9**	**25.7**	**44.9** (+3.0)
			100B tokens				
5k	67.2	30.8	42.7	52.2	44.1	26.7	44.0 (±0.0)
10k	68.9	**31.6**	42.7	51.6	45.1	25.7	44.3 (+0.3)
50k	68.9	30.8	45.1	52.6	49.1	26.2	45.5 (+1.5)
100k	70.2	**31.6**	46.1	52.9	49.1	25.8	45.9 (+1.9)
500k	**70.4**	**31.6**	**47.0**	**53.0**	**50.0**	**28.2**	**46.7** (+2.7)
			200B tokens				
5k	68.8	32.8	45.3	53.4	46.0	25.3	45.2 (±0.0)
10k	69.0	31.6	46.2	53.3	45.7	26.5	45.4 (+0.2)
50k	70.5	31.0	47.9	53.8	50.0	26.1	46.6 (+1.4)
100k	70.5	**33.6**	49.2	**54.6**	50.6	26.2	47.4 (+2.2)
500k	**70.7**	33.4	**50.2**	54.3	**51.8**	**29.6**	**48.3** (+3.1)
			500B tokens				
5k	69.7	32.6	49.6	52.9	47.7	26.4	46.5 (±0.0)
10k	70.8	**34.2**	49.7	54.5	49.0	26.2	47.4 (+0.9)
50k	70.2	32.2	52.0	54.4	51.6	27.2	47.9 (+1.4)
100k	70.1	33.4	52.7	55.3	52.8	27.8	48.7 (+2.2)
500k	**71.1**	31.8	**53.6**	**56.5**	**53.9**	**28.8**	**49.3** (+2.8)

Table 9: The performance on English commonsense reasoning tasks when we use 100B, 200B, and 500B tokens for training.

The difference of the performance among vocabulary sizes is smaller in the 10B tokens than ones in other training data sizes. Thus, the small training data size decreases the advantage of the large vocabulary sizes. These results explain the relation between our findings and the previous study (Ali et al., 2024). Ali et al. (2024) concluded that the small vocabulary size such as 30k is sufficient for English monolingual LLMs. We consider that they led the contrary conclusion to our findings because their training data, which is about 50B tokens, is much smaller than ours.

#Vocab	JSQuAD	JCQA	XWinograd	JAQKET	Avg.
10B tokens					
5k	1.6	37.1	51.0	0.9	22.7 (±0.0)
10k	1.4	44.2	**53.6**	0.5	24.9 (+2.2)
50k	2.7	47.9	51.0	1.6	25.8 (+3.1)
100k	5.3	48.9	51.7	3.3	27.3 (+4.6)
500k	**10.1**	**50.8**	52.5	**4.1**	**29.4** (+6.7)
50B tokens					
5k	36.3	49.4	53.7	3.3	35.7 (±0.0)
10k	42.6	**59.1**	56.0	7.7	41.4 (+5.7)
50k	42.8	56.8	55.7	12.2	41.9 (+6.2)
100k	40.9	56.8	**56.9**	17.5	43.0 (+7.3)
500k	**48.9**	57.2	54.8	**17.9**	**44.7** (+9.0)
100B tokens					
5k	45.0	55.0	56.9	5.2	40.5 (±0.0)
10k	49.8	**60.9**	56.7	12.3	44.9 (+4.4)
50k	51.4	56.0	56.8	18.4	45.7 (+5.2)
100k	49.1	58.7	**58.9**	20.7	46.9 (+6.4)
500k	**56.3**	60.1	55.7	**26.3**	**49.6** (+9.1)
200B tokens					
5k	50.5	58.5	56.7	7.5	43.3 (±0.0)
10k	53.8	61.1	57.7	16.9	47.4 (+4.1)
50k	55.8	54.0	**58.7**	21.4	47.5 (+4.2)
100k	54.7	**64.2**	58.0	27.2	51.0 (+7.7)
500k	**60.6**	61.4	56.4	**32.1**	**52.6** (+9.3)
500B tokens					
5k	56.4	62.2	58.4	10.4	46.9 (±0.0)
10k	59.6	62.8	58.8	20.2	50.4 (+3.5)
50k	59.3	**64.7**	59.0	26.5	52.4 (+5.5)
100k	60.1	**64.7**	**59.3**	31.8	54.0 (+7.1)
500k	**62.6**	62.9	58.8	**36.4**	**55.2** (+8.3)

Table 10: The performance on Japanese commonsense reasoning tasks when we use 100B, 200B, and 500B tokens for training.