Efficient and Effective Vocabulary Expansion Towards Multilingual Large Language Models

Seungduk Kim* **Seungtaek Choi*** **Myeongho Jeong**
Yanolja, South Korea
{seungduk.kim, seungtaek.choi, myeongho.jeong}@yanolja.com

Abstract

This report introduces `EEVE-Korean-v1.0`, a Korean adaptation of large language models that exhibit remarkable capabilities across English and Korean text understanding. Building on recent highly capable but English-centric LLMs, such as SOLAR-10.7B and Phi-2, where non-English texts are inefficiently processed with English-centric tokenizers, we present an efficient and effective vocabulary expansion (EEVE) method, which encompasses parameter freezing and subword initialization. In contrast to previous efforts that believe new embeddings require trillions of training tokens, we show that our method can significantly boost non-English proficiency within just 2 billion tokens. Surpassing most instruction-tuned LLMs on the Open Ko-LLM Leaderboard, as of January 2024, our model `EEVE-Korean-10.8B-v1.0` ranks as the leading Korean pre-trained model in the open-source community, according to Hugging Face's leaderboard. We open-source our models on Huggingface to empower the open research community in various languages.

1 Introduction

Recent advancements in the field of large language models (LLMs), such as GPT-4 (OpenAI, 2023), Gemini (Team et al., 2023a), and Claude (Anthropic, 2023), have demonstrated remarkable capabilities in processing and understanding multiple languages. On the other hand, though notable models in open source community, such as LLaMA (Touvron et al., 2023a,b), MPT (Team et al., 2023b), Falcon (Almazrouei et al., 2023), Mistral (Jiang et al., 2023), Mixtral (Jiang et al., 2024), SOLAR (Kim et al., 2023), and Phi-1.5 (Li et al., 2023b) have set benchmarks in English tasks, these developments have predominantly favored English, leading to a performance gap in non-English languages.

Such disparity can be found not only in their language proficiency but also in computational efficiency, where non-English languages like Korean require significantly more tokens than English even for equivalent semantic content (Figure 1). And, of course, this negatively affects the user experiences, such as longer response times, shorter context lengths, and higher API costs (Petrov et al., 2023). Expanding the tokenizer vocabulary, which introduces some frequently used yet long words as additional tokens, is thus indispensable for non-English users, but vocabulary expansion is a very challenging task because new embeddings require trillions of training tokens (Zhao et al., 2024).

To this end, this technical report presents a novel approach for **e**fficient and **e**ffective **v**ocabulary **e**xpansion, namely EEVE, which can better train the embeddings of newly added tokens. For ease of adaptation, we utilize subword-based embedding initialization and design seven training stages with parameter freezing, which elaborately adjust the order and amount of parameters to be trained. We meticulously transfer the advanced capabilities of foundational models from English to Korean by initially focusing on the training of only input embeddings and progressively expanding to encompass the full parameters in the final stage.

Using `EEVE`, we officially release a family of Korean LLMs, `EEVE-Korean-10.8B-v1.0`[1] and `EEVE-Korean-2.8B-v1.0`[2], which are built on recent English-centric LLMs, specifically SOLAR-10.7B (Kim et al., 2023) and Phi-2 (Li et al., 2023b), with further Korean-centric pre-training. We evaluate our models on `lm-evaluation-harness`[3] (Gao et al., 2023) for both English and Korean language tasks, such

* Equal Contribution.

[1] https://huggingface.co/yanolja/EEVE-Korean-10.8B-v1.0

[2] https://huggingface.co/yanolja/EEVE-Korean-2.8B-v1.0

[3] https://github.com/EleutherAI/lm-evaluation-harness

as boolean question answering (BoolQ; Clark et al. 2019), commonsense causal reasoning (COPA; Roemmele et al. 2011, context-sensitive word understanding (WiC; Pilehvar and Camacho-Collados 2019), commonsense reasoning (HellaSwag; Zellers et al. 2019), and sentiment negation recognition (SentiNeg). From the evaluation, we observe that our models outperform the recent open Korean pre-trained LLMs like OPEN-SOLAR-KO-10.7B (L. Junbum, 2024), Polyglot-Ko (Ko et al., 2023), and KoGPT (Kim et al., 2021), while preserving the strong English capability of the base English-centric LLMs in terms of benchmark performance, being ranked as the leading Korean pre-trained model in Open Ko-LLM Leaderboard (Park et al., 2023).

2 Efficient and Effective Vocabulary Expansion

To address the challenge of efficiently extending English-centric Language Models (LLMs) to include non-English languages, we introduce a novel methodology for vocabulary expansion. This method combines parameter freezing with subword-based embedding initialization to effectively incorporate and adapt to new linguistic tokens from languages beyond its initial training scope, thereby enhancing its applicability across various linguistic contexts. Our approach outlines a structured seven-stage training process, as illustrated in Figure 1, meticulously designed to effectively integrate new tokens into the model's vocabulary. During pre-training, our objective is causal language modeling.

Our core assumption is that foundational models, having been extensively trained in English texts, possess a substantial level of understanding and reasoning capabilities. Transferring these capabilities from English to another language, such as Korean, could be more efficient than developing performance from standalone Korean pre-training.

2.1 Preliminary 1: Tokenizer Training

We trained a new tokenizer on our Korean corpus. Since our goal is to maximize the leverage of the base model's performance, we maintained the base model's vocabulary and added 8,960 tokens from the corpus that appeared at least 6,000 times, prioritizing those with the highest frequency. Ultimately, the tokenizer's vocabulary expanded to 40,960 tokens for `EEVE-Korean-10.8B-v1.0`. This process

English (***8 tokens***)
"*Hello, the weather is nice today.*"
`['_Hello', ',', '_the', '_weather', '_is', '_nice', '_today', '.']`
Korean (***26 tokens***)
"*안녕하세요, 오늘은 날씨가 좋네요.*"
`['_', '안', '<0xEB>', '<0x85>', '<0x95>', '하', '세', '요', ',', '_', '오', '<0xEB>', '<0x8A>', '<0x98>', '은', '_', '날', '<0xEC>', '<0x94>', '<0xA8>', '가', '_', '좋', '네', '요', '.']`
Expanded Tokenizer (***9 tokens***)
`['_안', '녕', '하세요', ',', '_오늘은', '_날씨가', '_좋', '네요', '.']`

Table 1: A comparison of token consumption between English and Korean. We used the tokenizers of SOLAR (Kim et al., 2023) and our `EEVE-Korean-10.8B-v1.0`.

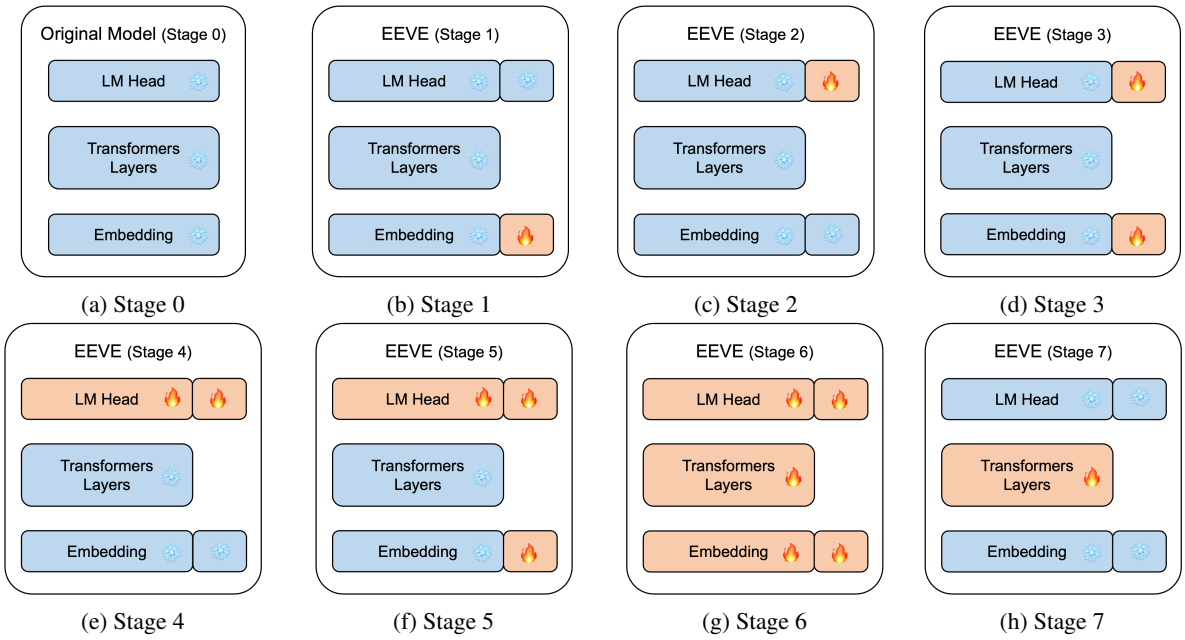
entailed several rounds of tokenizer training and a manual curation of tokens, based on an analysis of token frequency, ensuring a comprehensive and relevant vocabulary for our model. As shown in Table 1, the overall token consumption for Korean texts is significantly improved, almost three-fold, contributing to the reduction of computational costs during the entire training process.

2.2 Preliminary 2: Subword-based Embeddings Initialization

The integration process starts before actual training, introducing new input and output embeddings, called `embed_tokens` and `lm_head`, to the model's parameters. This preliminary step is crucial for preparing for the sophisticated learning process that follows.

For the input embeddings of the newly added tokens, we adopt the approach of using the average embeddings of the subword tokens that make up these new tokens as in (Hewitt, 2021; Welch et al., 2020). This method utilizes the semantic richness of the model's existing subword embeddings to offer a meaningful starting point for the new tokens' representations.

Conversely, the output embeddings for the newly added tokens are initialized with the embeddings of the first subword token that comprises the new token. This strategy aims to align the new tokens' output representations closely with the semantic characteristics of their constituent subwords, enabling a smoother integration into the model's predictive framework. The significance of such initialization



(a) Stage 0 (b) Stage 1 (c) Stage 2 (d) Stage 3

(e) Stage 4 (f) Stage 5 (g) Stage 6 (h) Stage 7

Figure 1: Training stages with parameter freezing. The fire and snowflake emojis indicate the trainable and frozen parameters respectively.

will be further discussed.

2.3 Multi-stage Training

Here we describe the nuanced approach of our seven-stage training methodology for efficient vocabulary expansion, emphasizing the meticulous process of integrating new tokens derived from languages beyond the initial English-centric training scope.

new_tokens 빼고 다 freeze

Stage 1 (new input embeddings): Initially, our focus is narrow yet critical: to learn the input embeddings of the newly added tokens while freezing all other model parameters. This stage is foundational, allowing the model to adjust its recognition and processing of these tokens from the beginning. The pre-initialized embeddings serve as a starting point, guiding the model to better utilize these new tokens in its existing framework. Our principal hypothesis here is that if the input and output token sequences in causal language modeling can be differentiated, by utilizing both the old and new tokenizers at the same time, the model can more efficiently and effectively learn new vocabulary embeddings, as it could leverage its established knowledge in the embedding spaces from old tokens. However, employing distinct tokenizers for input and output sequences at once poses implementation challenges, such as the difficulty of applying teacher forcing due to mismatched input/output sequences. Here, the subword-based embedding initialization (Sec 2.2) provides a proxy for using the old tokenizer for output sequences, such that the model is tasked to generate the subword token (old) given the whole word token (new). In other words, the model could learn to align their representations for generating the new token with that for generating its first subword token, by optimizing only the input embeddings without any modification of input/output token sequences as described in Figure 2. However, at this stage, the model is not yet able to distinguish between tokens sharing the same hidden state.

토크나이저가 다르니까 Sequence 맞추기 어려움
-> Subword 토크나이징으로 프록시처럼 맞춤

Input은 새로운 토크나이저 / Output은 기존 토크나이저로 학습시키면
OLD / NEW를 둘다 쓸 수 있으니까 기존 모델의 Reasoning을 활용가능하다고 생각함

Stage 2 (new output embeddings): Our goal is to enhance the model's proficiency in accurately generating new tokens across various contexts by solely adjusting the output embeddings (`lm_head`). The decision to freeze all other parameters stems from the model's current unstable state. Allowing both input and output embeddings to be trained simultaneously would complicate achieving convergence, thus hindering the model's progress toward optimal performance. By freezing most of the parameters, we achieve more stable convergence. Moreover, this approach significantly reduces the training time, as it eliminates the necessity for backpropagation through the other layers.

Stage 3 (new input and output embeddings): At this stage, the input embeddings (`embed_tokens`) still remain optimized based on the initial embeddings of the output embeddings. This stage allows

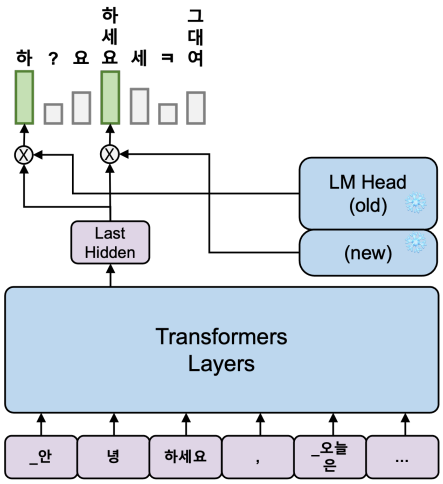


Figure 2: An illustrative example of showing how our subword-based embedding initialization enables harmonize the old and new tokens at Stage 1. In Stage 1, the output embeddings of newly added tokens are initialized with the output embeddings of their first subword tokens that make up these new tokens, such that the last hidden representation for predicting "하세요" yields the same logits for the newly added token "하세요" with its first subword token "하". Even if we give the new token "하세요" as a gold token, the gradients are eventually computed based on its subword token "하", so the model takes the input embeddings of "하세요" to predict its subword "하".

the updates of both input and output embeddings of the newly added tokens simultaneously. By aligning between input and output embeddings, the model learns to use the new tokens in both understanding and prediction.

Stage 4 (all output embeddings): As all the original parameters of the base model were frozen until this stage, we assumed the logits between old and new tokenizers were differently scaled, or less optimized to be used as a whole vocabulary. To this end, we begin to allow the update of the old parameters, specifically the output embeddings of old tokens here, making the model better generate the new tokens. In our preliminary experiments, we found this stage is critical for improving the model's generative capabilities.

Stage 5 (new input and all output embeddings): At this stage, the training extends to fine-tuning all output embeddings across the model's vocabulary while continuing to refine the input embeddings for the newly added tokens. The goal is to ensure that the model can accurately predict any token within its expanded vocabulary. This phase emphasizes the integration of new tokens within the broader context of the model's linguistic understanding, ensuring that they are both well-represented as inputs and accurately generated as outputs. This dual focus aids in harmonizing the model's overall performance, ensuring that the expanded vocabulary is seamlessly woven into its language generation processes.

Stage 6 (all layers): Contrary to being the final phase, this stage represents an advanced step in the vocabulary expansion process, where all model parameters are subject to optimization, including both newly introduced and pre-existing ones. The focus here is on integrating the enhancements made to the embedding layers within the model's overall parameters. Techniques such as QLoRA are utilized not just for efficiency but to ensure the preservation of the model's strong capabilities as much as possible, while allowing effective integration of the expanded vocabulary.

Stage 7 (internal layers): Following the extensive integration and optimization efforts, this stage serves as a "cool down" phase, focusing on updating the model's internal layers, which includes all the layers except the input and output embedding layers. The objective is to ensure that the enhancements made during the vocabulary expansion are deeply embedded within the model's core processing capabilities. This phase prepares the model for robust performance, ensuring that it not only recognizes and generates the new tokens but does so with a nuanced understanding of their use in varied linguistic contexts.

3 Implementation Details

3.1 Datasets

For *pre-training*, we curated publicly available Korean corpora from diverse sources, such as Korean web content, English vocabulary, and parallel corpus in Korean AI Hub[4], etc. To construct a high-quality pre-training corpus, we applied a set of pre-processing rules: 1) perplexity-based filtering, 2) n-gram repetition (Li et al., 2023a)-based filtering, and 3) stopword-based filtering. For the efficient training of newly added Korean tokens, we intentionally filtered out documents that do not contain many of these tokens. Subsequently, we acquired a pre-training corpus totaling 3.2M documents (or, 6.7GB).

As can be seen in Table 2, for the entire corpus, the SOLAR tokenizer needed to use 3.1B to-

[4] https://aihub.or.kr/

Model	Total (tokens)	Average (tokens)
SOLAR-10.7B	3.1B	964
EEVE-Korean-10.8B-v1.0	1.6B	500
Phi-2	5.6B	1748
EEVE-Korean-2.8B-v1.0	1.6B	484

Table 2: Comparison of tokenizers for our 6.7GB pre-training corpus of a total 3.2M documents.

뚜렷하다

kens to represent them, but our new tokenizer can do so with almost half, using only 1.6B tokens. This difference becomes even more pronounced in the case of Phi-2 and `EEVE-Korean-2.8B` models, where they require 5.6B tokens and 1.6B tokens respectively. Considering that transformers have a quadratic-increasing computation complexity with respect to token length, this can be interpreted in two significant ways. First, it allows for processing sequences more than 4 times longer on the same GPU. Or second, it means our model can be trained nearly 4 times more computationally efficiently on the same dataset. This difference becomes even more pronounced in the case of the Phi-2 and `EEVE-Korean-2.8B` tokenizers.

트랜스포머의 연산량은 토큰수의 2차식에 비례함

For *fine-tuning* of `EEVE-Korean` models, we employed the Direct Preference Optimization (DPO; Rafailov et al. 2023) based on LLaMA-Factory implementation. To further enhance the models' capabilities of following Korean instructions, we translated the publicly available instruction datasets, specifically Orca[5] (Mukherjee et al., 2023; Lian et al., 2023) and UltraFeedback[6] (Cui et al., 2023) into Korean. In the process of translating these datasets into Korean, ensuring the integrity of programming code formats and correcting translation errors, such as instances where both the source and target languages were inadvertently translated into Korean, was crucial for maintaining the quality and effectiveness of our fine-tuned models. We named the fine-tuned models as `EEVE-Korean-Instruct`.

DPO 사용

Orca랑 UltraFeedback 번역해서 사용

3.2 Training

As foundational architectures, we opt for SOLAR-10.7B (Kim et al., 2023) and Phi-2 (Li et al., 2023b), because both have shown outstanding performances among similar sizes of LLMs. This choice of foundational architectures aligns with our strategic training objectives, leveraging their proven strengths to ensure our new models achieve similar levels of language understanding and reasoning capabilities in Korean.

For the training of the model variants, we utilized two distinct codebases: Axolotl[7] for the initial pre-training phase and LLaMA-Factory[8] (hiyouga, 2023) for subsequent fine-tuning. These codebases provided a strong and reliable base for our training process.

Specifically, we train our models with a setup of 8 x NVIDIA H100 GPUs with 80GB memory each, utilizing 64 CPU cores. For `EEVE-Korean-10.8B-v1.0`, under bf16 precision, the training process is configured with a sequence of length 4096, gradient accumulation steps set to 4, and a micro-batch size of 8, whereas `EEVE-Korean-2.8B-v1.0` adopts a sequence length of 2048, gradient accumulation of 16, and a micro-batch size of 16. We employ the AdamW (Loshchilov and Hutter, 2018) optimizer, paired with a cosine learning rate scheduler that includes a warmup phase of 10 steps. The learning rate for the 10.8B variant is set to 4e-5, while we used 2e-4 for the small model. We continued training at each stage until the loss converged, observing the loss converged before reaching 400 global steps, which signifies the efficiency of our training strategy. Though our training strategy involves 7 different stages, it is noteworthy that, for our 2.8B variant, the overall pre-training can be done in less than two days as optimizing only the output embeddings doesn't incur much computation.

4 Evaluations

We evaluate our models on both Korean and English LLM benchmarks, to highlight the advantages of our vocabulary expansion method, which could efficiently leverage the strong multilingual capabilities of base foundational models. Desirably, we expect a model to show improved performance in Korean tasks and comparable performance in English tasks.

4.1 Benchmarks

For Korean tasks, we adopt the KoBEST benchmark (Jang et al., 2022), whose tasks are designed

[5] https://huggingface.co/datasets/Open-Orca/SlimOrca-Dedup

[6] https://huggingface.co/datasets/argilla/ultrafeedback-binarized-preferences-cleaned

[7] https://github.com/OpenAccess-AI-Collective/axolotl

[8] https://github.com/hiyouga/LLaMA-Factory

Model	Types	English			Korean					Avg.
		BQ (0)	CP (0)	HS (0)	BQ (0)	CP (0)	HS (0)	SN (0)	WIC (0)	
meta-llama/Llama-2-7b-hf	PT	0.7774	0.8700	0.5714	0.5242	0.5700	0.4420	0.4610	0.4881	0.5880
meta-llama/Llama-2-13b-hf	PT	0.8055	0.9100	0.6006	0.5214	0.6010	0.4380	0.5038	0.4881	0.6086
mistralai/Mistral-7B-v0.1	PT	0.8379	0.9200	0.6129	0.6282	0.5880	0.4300	0.5365	0.4881	0.6302
meta-llama/Llama-2-7b-chat-hf	FT	0.7976	0.8700	0.5779	0.5157	0.5530	0.4160	0.4987	0.4881	0.5896
meta-llama/Llama-2-13b-chat-hf	FT	0.8165	0.8800	0.6072	0.5057	0.5760	0.4040	0.4685	0.4881	0.5933
upstage/SOLAR-10.7B-v1.0 (base)	PT	0.8257	0.8700	0.6393	0.5057	0.5750	0.4320	0.6146	0.4881	0.6188
upstage/SOLAR-10.7B-Instruct-v1.0	FT	**0.8853**	**0.9400**	**0.6866**	0.8184	0.6370	0.4560	0.5668	0.4921	0.6853
beomi/OPEN-SOLAR-KO-10.7B*	PT	0.8187	0.8800	0.5570	0.8355	**0.8010**	**0.5040**	0.6952	0.4897	0.6976
yanolja/EEVE-Korean-10.8B-v1.0*	PT	0.8492	0.9000	0.6203	0.8568	0.7530	0.4900	0.6675	**0.4992**	0.7045
yanolja/EEVE-Korean-Instruct-10.8B-v1.0*	FT	0.8810	0.9300	0.6502	**0.8860**	0.7610	0.4700	**0.9521**	0.4937	**0.7530**
microsoft/Phi-2 (base)	PT	**0.8336**	**0.9000**	**0.5583**	0.5021	0.4770	0.3280	0.5063	0.4881	0.5742
daekeun-ml/phi-2-ko-v0.1*	PT	0.6141	0.5800	0.3257	0.5164	**0.6100**	**0.3860**	0.4484	0.4881	0.4961
yanolja/EEVE-Korean-2.8B-v1.0*	PT	0.7404	0.8900	0.5247	0.5299	0.5820	0.3800	0.5164	0.4881	0.5814
yanolja/EEVE-Korean-Instruct-2.8B-v1.0*	FT	0.8248	0.8700	0.5392	**0.7066**	0.5640	0.3660	**0.5290**	**0.5230**	**0.6153**

Table 3: Main evaluation results based on `lm-evaluation-harness`. The dataset names are abbreviated for brevity: BQ for BoolQ, CP for COPA, HS for HellaSwag, and SN for SentiNeg. Korean tasks are from (Jang et al., 2022). Accuracy (`acc`) is used as the evaluation metric for all tasks. In the 'Types' column, the models are categorized into two groups: pre-trained (PT) and fine-tuned (FT). We denote the models trained in Korean datasets with *. For ease of reproduction, we adopt their official names at HuggingFace.

to evaluate the various aspects of language understanding and reasoning. Specifically, this benchmark provides a Korean-translated version of language understanding tasks: boolean question answering (BoolQ; Clark et al. 2019), commonsense causal reasoning (COPA; Roemmele et al. 2011, context-sensitive word understanding (WiC; Pilehvar and Camacho-Collados 2019), commonsense reasoning (HellaSwag; Zellers et al. 2019), and sentiment negation recognition (SentiNeg). For English tasks, we employ the following original tasks of KoBEST, BoolQ, COPA, and HellaSwag, which can better highlight the alignment between the English and Korean capabilities of LLMs. To ensure consistent comparisons, we employ an open-source LLM evaluation framework, `lm-evaluation-harness`[9] (Gao et al., 2023).

4.2 Results

We now present evaluation results for both our `EEVE-Korean` and `EEVE-Korean-Instruct` variants with other top-performing models in Table 3. `EEVE-Korean-10.8B-v1.0` outperforms other pre-trained models of similar sizes in the average performance. It is noteworthy that, `EEVE-Korean` is the only case where the performance in Korean is improved without compromising the performance in English. For example, though OPEN-SOLAR-KO-10.7B, which is built on the same base model as ours, performs slightly better than our `EEVE-Korean-Instruct-10.8B-v1.0`, it fails to preserve the English capabilities, showing lower performance in English tasks than its base model, SOLAR-10.7B-v1.0. We observe similar trends even for our smaller model, `EEVE-Korean-2.8B-v1.0` in comparison with the phi-2-ko-v0.1 model, sharing Phi-2 as its base model. This demonstrates the effectiveness of our training strategy, especially considering that we used even fewer training tokens than our competitors.

Notably, but not surprisingly, preference tuning on English datasets even makes the models underperform in Korean tasks. For example, LLaMA-2-chat variants, which are the preference-tuned version of LLaMA-2 checkpoints, show improved performances in English tasks (Llama-2-7b 0.7774 → Llama-2-7b-chat 0.7976 in English BoolQ), while underperforming in Korean tasks (Llama-2-7b 0.5242 → Llama-2-7b-chat 0.5157 in Korean BoolQ), which highlights the importance of Korean-specific training for LLMs. On the other hand, we observe that preference tuning our models on Korean instruction datasets doesn't hurt the model performance in English tasks, rather even improving it. We posit that it is because the embedding spaces are already well-aligned between Korean and English tokens, thus fine-tuning on a specific language doesn't incur a significant change in model parameters.

[9] https://github.com/EleutherAI/lm-evaluation-harness

5 Conclusion & Future Work

The report introduces `EEVE-Korean-v1.0`, a Korean adaptation of large language models that utilizes an Efficient and Effective Vocabulary Expansion (EEVE) method to enhance Korean text processing capabilities significantly. The method, based on parameter freezing and subword initialization, enables the `EEVE-Korean-10.8B-v1.0` model to excel in Korean language tasks while maintaining strong English capabilities. Achieved with a corpus of just 2 billion tokens, this approach represents a notable advancement in language model training efficiency and effectiveness. By making these models available to the research community, the project aims to contribute to the development of more inclusive and efficient language processing technologies.

Expanding our vision, future efforts will explore the application of our vocabulary expansion methodology to additional languages, assessing its generalizability and effectiveness. We aim to not only extend the `EEVE-Korean` model's linguistic range but also to delve deeper into evaluating its reasoning and generative capabilities through diverse tasks, including complex mathematical reasoning tests like GSM8K (Cobbe et al., 2021), and human evaluations in interactive settings like chatbots (Zheng et al., 2023). Moreover, efforts to enhance pre-training data quality, and to analyze performance in code-switching scenarios (Zhang et al., 2023) will underpin our commitment to refining the model's robustness and versatility. These initiatives are designed to broaden the model's applicability and efficacy, pushing the boundaries of what is achievable with advanced language models.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Anthropic. 2023. Model card and evaluations for claude models. *Anthropic technical Report*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

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.

John Hewitt. 2021. Initializing new word embeddings for pretrained language models.

hiyouga. 2023. Llama factory. `https://github.com/hiyouga/LLaMA-Factory`.

Myeongjun Jang, Dohyung Kim, Deuk Sin Kwon, and Eric Davis. 2022. Kobest: Korean balanced evaluation of significant tasks. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3697–3708.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. 2023. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*.

Ildoo Kim, Gunsoo Han, Jiyeon Ham, and Woonhyuk Baek. 2021. Kogpt: Kakaobrain korean(hangul) generative pretrained transformer.

Hyunwoong Ko, Kichang Yang, Minho Ryu, Taekyoon Choi, Seungmu Yang, Sungho Park, et al. 2023. A technical report for polyglot-ko: Open-source large-scale korean language models. *arXiv preprint arXiv:2306.02254*.

L. Junbum. 2024. Solar-ko-10.7b.

Huayang Li, Tian Lan, Zihao Fu, Deng Cai, Lemao Liu, Nigel Collier, Taro Watanabe, and Yixuan Su. 2023a. Repetition in repetition out: Towards understanding neural text degeneration from the data perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023b. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, "Teknium", and Nathan Hoos. 2023. Slimorca dedup: A deduplicated subset of slimorca.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Chanjun Park, Hwalsuk Lee, Hyunbyung Park, Hyeonwoo Kim, Sanghoon Kim, Seonghwan Cho, Sunghun Kim, and Sukyung Lee. 2023. Open ko-llm leaderboard.

Aleksandar Petrov, Emanuele La Malfa, Philip HS Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *arXiv preprint arXiv:2305.15425*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023a. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

MosaicML NLP Team et al. 2023b. Introducing mpt-30b: Raising the bar for open-source foundation models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Charles Welch, Rada Mihalcea, and Jonathan K Kummerfeld. 2020. Improving low compute language modeling with in-domain embedding initialisation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8625–8634.

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*, pages 4791–4800.

Ruochen Zhang, Samuel Cahyawijaya, Jan Christian Blaise Cruz, and Alham Fikri Aji. 2023. Multilingual large language models are not (yet) code-switchers. *arXiv preprint arXiv:2305.14235*.

Jun Zhao, Zhihao Zhang, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Llama beyond english: An empirical study on language capability transfer. *arXiv preprint arXiv:2401.01055*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.