R-BPE: Improving BPE-Tokenizers with Token Reuse

Nancy Hamdan[1] **and Osama Rakan Al Mraikhat**[1] **and Fadi A. Zaraket**[1,2]
[1]Arab Center for Research and Policy Studies, Doha
[2]American University of Beirut
{nhamdan,oalmraikhat,fzaraket}@dohainstitute.edu.qa

Abstract

This paper presents R-BPE, a lightweight framework for adapting existing Byte-Pair Encoding (BPE) tokenizers to better support a specified target language. It reuses tokens from user-excluded languages and creates ID-based maps to resolve the new tokens of the chosen language. We evaluate R-BPE on Arabic as a target language. R-BPE reduced subword fertility by an average of 24.4% across the LLaMA 3.1 8B, Command R 35B, and Qwen 3 8B models. Applied to LLaMA 3.1 8B in continued pretraining mode, R-BPE yields a 7.33% reduction in training time. On the ArabicMMLU benchmark, the resulting model improved by 5.09 points on five in-domain topics and matched the original model's overall performance. It also preserved performance on EnglishMMLU. R-BPE effectively leverages existing models' tokenizers, embedding layers, and performance to better support target languages without incurring model size changes. We release an R-BPE implementation that is compatible with HuggingFace interfaces and thereby readily applicable to a wide range of existing models at https://acr.ps/1L9GPmL.

1 Introduction

Large pretrained language models (PLMs) provide robust language representations after a single costly training run (Devlin et al., 2019; Conneau and Lample, 2019), but their language-specific performance is often limited by their tokenizers. Multilingual PLMs prioritize high-resource languages in their vocabularies, leaving others with poor coverage (Rust et al., 2021). This leads to over-segmentation of common words, inflating sequence length and compute cost, and degrading downstream performance (Rust et al., 2021; Petrov et al., 2023).

Prior work addresses this using two main strategies: adding new tokens (Wang et al., 2020) or rebuilding the vocabulary for continued pretraining (Minixhofer et al., 2022; Dobler and de Melo, 2023). While both can be effective, they increase memory usage or require expensive retraining.

A lighter alternative has been proposed for SentencePiece (Kudo and Richardson, 2018) tokenizers: vocabulary replacement. By swapping low-utility tokens with target-language-specific ones, researchers have improved performance without increasing vocabulary size (Kajiura et al., 2023; Kiulian et al., 2024). This is possible because SentencePiece unigram tokenizers store tokens explicitly as strings and scores, enabling controlled substitutions. However, this method is incompatible with byte-pair encoding (BPE) (Sennrich et al., 2016) tokenizers, where tokens are defined implicitly via a merge table. Naive replacements can disrupt merges and lead to non-deterministic tokenization.

We present R-BPE, a lightweight framework for BPE-based PLMs that expands a target language vocabulary. It reuses tokens from user-excluded languages to support additional tokens from the target language. Given a representative corpus for the target language, we construct an expanded vocabulary. We identify tokens from user-excluded languages in the base vocabulary of the tokenizer and replace them with the newly learned ones, keeping the overall vocabulary size fixed. We use an ID-based map to encode and decode the reused tokens. Other untouched tokens retain their IDs and embeddings to preserve original model behavior and performance.

R-BPE wraps any HuggingFace-compatible BPE tokenizer and requires no model changes, nor additional parameters. We evaluate R-BPE on Arabic and show that it shortens sequences, improves task performance, and preserves English capabilities. We make the following contributions:

1. We propose a vocabulary-replacement method for BPE-based PLMs. It applies for any target language where token reuse is possible.
2. We demonstrate efficiency and accuracy improvements on Arabic while preserving En-

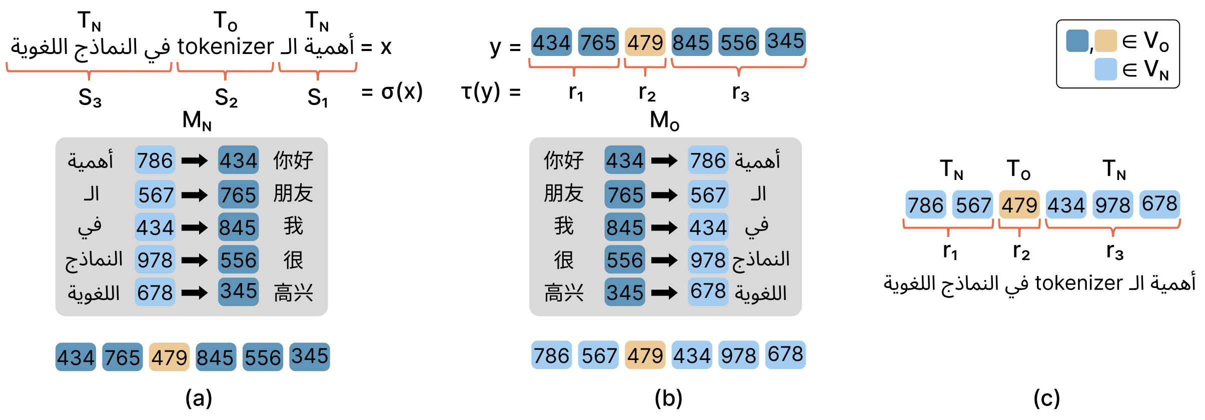

Figure 1: R-BPE with l_t = Arabic and $L_P = \{\text{Arabic}, \text{Latin}\}$. (a) Input is segmented by script; Arabic spans are encoded with T_N and mapped via M_N, others with T_O; (b) Token IDs are segmented by presence in $\text{dom}(M_O)$ and mapped back via M_O when applicable; (c) Final decoding uses T_N or T_O per segment to recover text.

glish performance.

3. We release an open-source implementation compatible with any HuggingFace BPE tokenizer.

2 Related Work

Vocabulary Extension Prior work adapts PLMs to target languages by expanding the vocabulary with tokens from a target-language tokenizer and continuing pretraining. This has been done for Chinese in LLaMA-2 (Cui et al., 2024), Korean in SOLAR-10.7B and Phi-2 (Kim et al., 2024), and Arabic in a bilingual LLaMA-2 model (Bari et al., 2024).

Vocabulary Replacement Other studies fully replace the tokenizer with a new tokenizer trained specifically for the target language, necessitating complete embedding reinitialization. Approaches include replacing English-centric tokenizers entirely and aligning new embeddings using multilingual static embeddings (Minixhofer et al., 2022), employing shared tokens as anchors between tokenizers (Dobler and de Melo, 2023), or training a bilingual tokenizer with a fixed vocabulary size, retaining shared tokens' embeddings and initializing new tokens using subword embedding averages or reinitialization (Nozaki et al., 2025).

BPE Algorithm Improvements PickyBPE (Chizhov et al., 2024), Scaffold-BPE (Lian et al., 2025), and Overlap BPE (Patil et al., 2022) propose improvements to the BPE algorithm. They train a tokenizer from scratch and modify the vocabulary generation process to reduce redundant tokens or improve cross-lingual overlap. They do not leverage or adapt existing BPE tokenizers. R-BPE focuses on modifying an existing BPE tokenizer to better support the target languages.

Vocabulary Reusing Vocabulary reusing recently emerged as a balanced alternative between extending and fully replacing vocabularies. Methods include substituting rare or non-subword tokens with domain-specific tokens, retaining original SentencePiece scores (Kajiura et al., 2023), and replacing non-English tokens in Mistral (Jiang et al., 2023) and Gemma 2 (Gemma Team et al., 2024), keeping original IDs for shared tokens and assigning new IDs for added tokens (Kiulian et al., 2024).

Vocabulary reuse efficiently adapts models without increasing parameters or GPU memory requirements. Current implementations only support SentencePiece unigram tokenizers, which store explicit $\langle piece, score\rangle$ pairs. This allows token substitution without disrupting tokenization logic. Conversely, BPE tokenizers implicitly define tokens via ranked and recursive merges, where prefix tokens might appear in several merge entries. This results in non-deterministic, and thus impractical, behavior with arbitrary token substitutions.

This work extends vocabulary reuse to BPE tokenizers, facilitating lightweight, domain-specific vocabulary augmentation in open-weight BPE-based LLMs without added memory overhead.

3 Problem Formulation

Let T_O be the **original** BPE tokenizer released with a PLM, that has vocabulary $V_O = \{v_0^O, \ldots, v_{K-1}^O\}$. We want to find a subset $S_R \subseteq \{0, \ldots, K-1\}$ of reusable token IDs, such that each $i \in S_R$ corresponds to a token $v_i^O \in V_O$. These are token IDs

Tokenizer	$\lvert V_O\rvert$	$\lvert V_N\rvert$	$\lvert S_C\rvert$	$\lvert L_R\rvert$	**En**$_{T_O}$	**En**$_{T_{\text{R-BPE}}}$	**Ar**$_{T_O}$	**Ar**$_{T_{\text{R-BPE}}}$	$\downarrow$ **(%)En**	$\downarrow$ **(%)Ar**
LLaMA	128,000	16,632	2,790	35	1.32	1.32	2.37	1.87	0.00	21.11
Command R	255,000	60,214	6,494	35	1.34	1.34	2.21	1.67	0.00	24.43
Qwen	151,643	41,752	4,509	40	1.34	1.34	2.38	1.72	0.00	27.73

Table 1: Tokenizer statistics and subword fertility scores for English (**En**) and Arabic (**Ar**). $|V_O|$: original vocabulary size; $|V_N|$: new tokenizer vocabulary size; $|S_C|$: size of the common token set; $|L_R|$: number of reusable languages; $\downarrow$ **(%)** = percentage reduction in fertility scores from T_O to T_{R-BPE}.

from the original vocabulary that we consider suitable for reuse in representing new vocabulary. Let T_N be the **new** BPE tokenizer we want to train on a corpus of our target language l_t and that would have vocabulary $V_N = \{v_0^N, \ldots, v_{P-1}^N\}$. We require that $|V_N| \leq |S_R|$, ensuring that every token in V_N can be mapped to a unique reusable token ID via M_N. Let $S_C \subseteq \{0, \ldots, K-1\}$ be the set of token IDs in V_O that match tokens in V_N:

$$S_C = \{i \mid \exists j,\ v_i^O = v_j^N\}.$$

We define a mapping $M_N : \{0, \ldots, P-1\} \to S_R$ by:

$$M_N(j) = \begin{cases} i, & \text{if } v_j^N = v_i^O \text{ for some } i \in S_C, \\ \text{pick } i \in S_R & \text{otherwise.} \end{cases}$$

Conversely, we define $M_O : S_R \to \{0, \ldots, P-1\}$ as the inverse mapping of M_N on its image, so that $M_O(M_N(j)) = j$ for all j in the domain of M_N. Our objective is to construct a mapping layer that enables encoding and decoding of text written in the target language l_t using T_N, while all other text is processed using T_O. Figure 1 demonstrates the overall R-BPE framework, which will be described in detail in the following sections.

4 Vocabulary Language Classification

We classify tokens in V_O by examining their Unicode character ranges to infer language associations. If a token contains multiple scripts, it receives a composite label representing all detected languages. This creates a language-based partition of V_O, where each language $\ell \in L_O$ corresponds to a set of token IDs $\mathcal{M}_C(\ell)$.

To identify reusable tokens S_R, we first define a *preserved set* $L_P \subseteq L_O$ containing the target language l_t and any other languages the user wishes to continue supporting. Tokens from these languages are excluded from reuse. For example, if l_t is Arabic but support for English is required, English must be included in L_P to preserve its tokens. The reusable languages form the set $L_R = L_O \setminus L_P$. We then sort languages in L_R by the size of their token sets $\mathcal{M}_C(\ell)$, accumulating token IDs into S_R until reaching the threshold $|S_R| \geq h$. Threshold h is the desired size, specified by the user, of the new vocabulary V_N.

5 Target Tokenizer Training

Data Preprocessing Before training T_N, we filter its training corpus to prevent token ID conflicts in M_N. Specifically, cases where a token might belong to both S_C and S_R. To avoid this, we discard any example containing text from languages in L_R. Only examples composed entirely of text from the preserved set L_P are retained.

Training We train T_N using the HuggingFace `tokenizers` library on the cleaned dataset. Its vocabulary size is set to $|V_N| = |S_R|$.

6 Mapping Layer

Encode Given an input string $x = c_1 \ldots c_L$, we segment x into maximal contiguous spans of identical script type using a function $\sigma(x) = (s_1, \ldots, s_m)$, where each segment s_i either belongs to the target language l_t or not. Whitespace are appended to the current segment regardless of script, preserving spacing and simplifying segmentation. Encoding proceeds as follows:

$$\text{Enc}(x) = \big\|_{s \in \sigma(x)} \begin{cases} M_N(T_N(s)) & \text{if } s \in l_t, \\ T_O(s) & \text{otherwise.} \end{cases}$$

Decode Given an input sequence of token IDs $y = (y_1, \ldots, y_n)$, we segment it based on whether each token ID belongs to the domain of M_O. Let $\tau(y) = (r_1, \ldots, r_k)$ be the resulting segmentation, where each segment r_i is either a mapped segment (all $y_j \in \text{dom}(M_O)$) or an unmapped segment (all $y_j \notin \text{dom}(M_O)$). Decoding is defined as:

$$\text{Dec}(y) = \big\|_{r \in \tau(y)} \begin{cases} T_N(M_O(r)) & \text{if } r \subseteq \text{dom}(M_O), \\ T_O(r) & \text{otherwise.} \end{cases}$$

A complication arises with segments containing byte-level tokens representing incomplete UTF-8 sequences. Let $S_U \subseteq \{0, \ldots, K-1\}$ denote

L_P	$\|\mathbf{L_R}\|$	$\|\mathbf{S_R}\|$	$\|\mathbf{S_C}\|$	$\mathbf{Fr}_{Ar}$	$\mathbf{Fr}_{En}$	$\mathbf{Fr}_{Ko}$	$\mathbf{Fr}_{Ru}$	$\mathbf{Fr}_{Gr}$	$\mathbf{Fr}_{Tha}$
Ar + Lat + CJK + Cyr + Gr + Tha	15	1,035	936	2.76	1.24	2.61	2.22	2.54	13.77
Ar + Lat + CJK + Cyr + Gr	17	2,419	1,564	2.34	1.24	2.61	2.22	2.54	–
Ar + Lat + CJK + Cyr	19	3,806	1,870	2.16	1.24	2.61	2.22	–	–
Ar + Lat + CJK	21	10,268	2,451	1.85	1.24	2.61	–	–	–
Ar + Lat	37	18,019	2,857	1.71	1.24	–	–	–	–
Ar + Lat + Gr	35	16,632	2,790	1.73	1.24	–	–	2.54	–

Table 2: Ablation study on adapting the **LLaMA** tokenizer using R-BPE. Fertility scores (Fr) are reported for Arabic (Ar), English (En), Korean (Ko), Russian (Ru), Greek (Gr), and Thai (Tha). Fertility scores for T_O (constant across setups) are: Ar=2.28, En=1.24, Ko=2.61, Ru=2.22, Gr=2.54, Tha=13.77. Empty cells indicate that the corresponding language was not preserved in the resulting tokenizer. Language/script abbreviations: Lat = Latin, CJK = Chinese/Japanese/Korean, Cyr = Cyrillic, Gr = Greek, Tha = Thai.

token IDs in V_O that yield the Unicode replacement character (�) when decoded in isolation. These problematic tokens include both raw byte tokens and tokens created through BPE merges over byte sequences that individually do not form valid UTF-8 characters. Since GPT-style BPE tokenizers (Radford et al., 2019) initially include all 256 byte tokens, all byte tokens are in S_C and thus within $\mathrm{dom}(M_O)$. Naive segmentation risks isolating these tokens incorrectly.

We address this by modifying $\tau(y)$ so that each problematic token ID $y_j \in S_U$ is grouped with up to three subsequent tokens, forming a decoding window of at most four tokens. If any token ID in the window lies outside $\mathrm{dom}(M_O)$, the window is decoded using T_O. Otherwise, we apply M_O and decode with T_N. If the decoded result contains no replacement characters, the window is valid. The final output concatenates all decoded segments.

7 Experiments

We evaluate R-BPE in the context of adapting LLMs to better support Arabic as the target language, l_t = Arabic. We preserve support for Latin and Greek, $L_P = \{\text{Arabic}, \text{Latin}, \text{Greek}\}$. Latin coverage maintains English performance and enables knowledge transfer, and Greek retains accurate tokenization of common mathematical symbols.

7.1 Training Dataset

We curated a 1GB Arabic corpus with contemporary books published in social sciences and humanities. These cover society, governance, and public policy topics including cultural, economic, and sociological perspectives. It also includes school curricula and educational materials from various grade levels. We applied Unicode normalization to the corpus as explained in Appendix A to ensure consistency.

7.2 Tokenizer Efficiency

We evaluate our method on three open-weight BPE-based LLM tokenizers: ***LLaMA*** 3.1 8B[1], ***Command R*** 35B[2], and ***Qwen*** 3 8B[3], and refer to them by these abbreviations throughout the paper. For each original T_O, we construct a corresponding T_{R-BPE} that combines T_O with a T_N trained on the corpus introduced in Section 7.1. We compare Arabic and English fertility scores, defined as the average number of subword units per word (Rust et al., 2021), for T_O and T_{R-BPE}. We use the Aya dataset (Singh et al., 2024) to measure English and Arabic fertility. For Arabic, we also use the ANTCorpus (Chouigui et al., 2021). Table 1 reports tokenizer statistics and fertility scores. On average, T_{R-BPE} reduces Arabic fertility by 24.42% while maintaining English fertility scores. This is achieved by the mapping layer that routes Arabic text to T_N and all other text to T_O.

7.3 Parity and Continued Words

Since the models we evaluate are English-centric, we report tokenization parity with respect to English in Table 3. Specifically, we compute the premium for Arabic relative to English on the FLORES-200 dataset (NLLB Team et al., 2024), following the definition in (Petrov et al., 2023). To further assess segmentation quality improvements, we also report the proportion of continued words for Arabic on the Universal Dependencies v2.7 PADT treebank (Zeman et al., 2020), following (Rust et al., 2021). As shown in Table 3, The Arabic premium relative to English is reduced by 24.6-32%, indicating more balanced segmentation. Continued words drop by over 50%, reflecting

[1]https://huggingface.co/meta-llama/Llama-3.1-8B

[2]https://huggingface.co/CohereLabs/c4ai-command-r-v01

[3]https://huggingface.co/Qwen/Qwen3-8B

Tokenizer	$\mathbf{P}_{Ar}$			$\mathbf{PCW}_{Ar}$		
	T_O	$T_{\text{R-BPE}}$	↓(%)	T_O	$T_{\text{R-BPE}}$	↓(%)
LLaMA	1.67	1.26	24.6	0.55	0.27	50.9
Command R	1.50	1.10	26.7	0.46	0.15	67.4
Qwen	1.65	1.12	32.1	0.50	0.18	64.0

Table 3: Premium ($\mathbf{P}_{Ar}$) w.r.t. English and proportion of continued-words ($\mathbf{PCW}_{Ar}$) for Arabic (↓% = percentage reduction from T_O to T_{R-BPE}).

cleaner and more complete Arabic word boundaries.

7.4 Token Reuse vs. Language Coverage

To examine how the extent of token reuse (i.e., the size of S_R) affects Arabic fertility while maintaining support for languages in L_P, we adapt the ***LLaMA*** tokenizer with different L_P configurations, varying the size and composition of S_R. Results in Table 2, computed on the FLORES-200 dataset (NLLB Team et al., 2024), show that R-BPE enables flexible trade-offs between enhancing support for l_t and preserving multilingual coverage. The configuration used in our main experiments corresponds to the table's last row.

7.5 Language Adaptive Pretraining

To evaluate the effectiveness of our R-BPE framework for language adaptation (Chau et al., 2020), we conduct continued pretraining using the **LLaMA 3.1 8B** model as our base, denoted **Base$_{\text{LLaMA}}$**. We train two variants on our dataset: **Trained$_{\text{LLaMA}}$** using the original tokenizer T_O, and **R-BPE$_{\text{LLaMA}}$** using our modified tokenizer T_{R-BPE}. For **R-BPE$_{\text{LLaMA}}$**, embeddings of the new tokens that are mapped to reused tokens are initialized by averaging the embeddings of their constituent subwords from T_O. This setup allows us to assess whether the expanded Arabic vocabulary in **R-BPE$_{\text{LLaMA}}$** leads to improved performance on Arabic, while maintaining the same model size and benefiting from longer sequence lengths.

Following the pretraining setup of the LLaMA 3 models (Grattafiori et al., 2024), we apply sample packing, with a maximum sequence length of 2048. This yields 49,600 sequences using T_{R-BPE} and 68,629 using T_O, as T_{R-BPE}'s lower fertility allows more tokens per sequence. Both models are trained for 8 epochs on 8×H100 SXM5 GPUs. Model hyperparameters are listed in Appendix B.

We evaluate on ArabicMMLU (Koto et al., 2024) and EnglishMMLU (Hendrycks et al., 2021), reporting average accuracy every two epochs in Table 4. We also report accuracy on ArabicMMLU subsets aligned with our training domain: social science, political science, philosophy, law, and civics which we denote by DataMMLU.

Model	Ep	ArMMLU	DataMMLU	EnMMLU
Base$_{\text{LLaMA}}$	–	35.44	36.12	44.33
Trained$_{\text{LLaMA}}$	2	36.64	36.84	44.54
	4	37.45	36.50	44.80
	6	37.69	36.84	44.70
	8	37.89	36.72	44.80
R-BPE$_{\text{LLaMA}}$	2	29.71	32.73	44.35
	4	33.25	38.38	44.22
	6	34.61	39.91	44.19
	8	35.40	41.21	44.04

Table 4: Comparison with **LLaMA** models across training epochs (Ep) on ArabicMMLU and EnglishMMLU benchmarks. DataMMLU is a subset of ArabicMMLU with topics aligned with R-BPE's training dataset.

Training **Trained$_{\text{LLaMA}}$** took 10 hours, while **R-BPE$_{\text{LLaMA}}$** took 7 hours and 20 minutes. Using T_{R-BPE} yields a 7.33% reduction in training time thanks to the shorter sequences produced. At the end of training, **Trained$_{\text{LLaMA}}$** scored 37.89% on ArabicMMLU, 36.72% on DataMMLU, and 44.80% on EnglishMMLU. **R-BPE$_{\text{LLaMA}}$** scored 35.40%, 41.21%, and 44.04%, respectively, while **Base$_{\text{LLaMA}}$** achieved 35.44%, 36.12%, and 44.33%.

Although **R-BPE$_{\text{LLaMA}}$** underperformed **Trained$_{\text{LLaMA}}$** on ArabicMMLU by 2.49%, it matched the base model and outperformed both on DataMMLU by 5.09%. English performance remained stable across all models. For a lower compute budget, R-BPE provides stronger in-domain accuracy and shorter sequences. In future work, we will perform more continued pre-training steps with more diverse topics to fully adapt the reused embeddings. We hypothesize that this will compensate for the remaining 2.5-point gap on the ArabicMMLU out-of-domain topics.

8 Conclusion

In this paper, we presented R-BPE, a lightweight vocabulary reuse framework designed specifically for BPE-based tokenizers. Our experiments demonstrate that R-BPE can effectively reduce subword fertility for a specified target language like Arabic without compromising English performance. Continued pretraining confirms that R-BPE maintains baseline capabilities while improving in-domain accuracy and computational efficiency. Future work should explore scaling the method to larger datasets to further enhance out-of-domain performance.

Limitations

R-BPE uses Unicode-based vocabulary language classification. As such, when a target language is specified (e.g., Arabic), all text written in the corresponding script is routed to T_N to ensure correct tokenization. Therefore, if support for languages sharing the same script (e.g., Urdu) is needed using T_O, R-BPE cannot be used, as this may lead to unexpected tokenization behavior. We leave the exploration of more fine-grained vocabulary language classification methods to future work.

R-BPE requires the preserved set L_P to omit at least some pretrained languages, as it is not feasible to preserve all of them. Thus, trade-offs naturally arise between token reuse and multilingual support, as demonstrated in the ablation study in Section 7.4.

Our downstream evaluations are conducted primarily on Arabic, leveraging a corpus focused on social sciences and humanities, providing robust results for this domain. However, additional evaluations on languages from diverse domains, scripts, and typologies would help substantiate the general applicability of R-BPE. We also evaluate R-BPE only on BPE-based decoder-only PLMs. In principle, the token reuse approach could be adapted to other model types, such as BERT-style encoders, or to non-BPE tokenization schemes, but we leave this to future work.

Furthermore, we initialized embeddings of reused tokens by averaging subword embeddings; alternative initialization approaches may yield improved performance. Finally, our experiments used moderately sized models (LLaMA 3.1-8B) and datasets (1GB), and further work is needed to evaluate how R-BPE performs at larger scales.

Ethics Statement

Our method reuses token IDs from PLMs, which may carry forward biases present in the original training data. This can potentially lead to model behavior that does not align with the cultural or linguistic norms of the target language community. We emphasize the importance of conducting thorough bias evaluations and incorporating human oversight prior to deploying R-BPE adapted models in downstream applications.

Acknowledgements

We thank our colleagues Qusai Abu-Obaida for the initial suggestion to explore token-ID mapping between new and original tokenizers, and Hadi Hamoud for assisting with curating the training dataset used in Section 7.1. We are also grateful to the anonymous reviewers, whose constructive feedback helped improve this paper.

References

M Saiful Bari, Yazeed Alnumay, Norah A. Alzahrani, Nouf M. Alotaibi, Hisham A. Alyahya, Sultan AlRashed, Faisal A. Mirza, Shaykhah Z. Alsubaie, Hassan A. Alahmed, Ghadah Alabduljabbar, Raghad Alkhathran, Yousef Almushayqih, Raneem Alnajim, Salman Alsubaihi, Maryam Al Mansour, Majed Alrubaian, Ali Alammari, Zaki Alawami, Abdulmohsen Al-Thubaity, and 6 others. 2024. Allam: Large language models for arabic and english. *Preprint*, arXiv:2407.15390.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.

Pavel Chizhov, Catherine Arnett, Elizaveta Korotkova, and Ivan P. Yamshchikov. 2024. BPE gets picky: Efficient vocabulary refinement during tokenizer training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16587–16604, Miami, Florida, USA. Association for Computational Linguistics.

Amina Chouigui, Oussama Ben Khiroun, and Bilel Elayeb. 2021. An arabic multi-source news corpus: Experimenting on single-document extractive summarization. *Arabian Journal for Science and Engineering*, 46(4):3925–3938.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yiming Cui, Ziqing Yang, and Xin Yao. 2024. Efficient and effective text encoding for chinese llama and alpaca. *Preprint*, arXiv:2304.08177.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Konstantin Dobler and Gerard de Melo. 2023. FOCUS: Effective embedding initialization for monolingual specialization of multilingual models. In *Proceedings of the 2023 Conference on Empirical Methods in*

Natural Language Processing, pages 13440–13454, Singapore. Association for Computational Linguistics.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

HuggingFace. 2021. Tokenizers. `https://github.com/huggingface/tokenizers`.

Albert Q Jiang, A Sablayrolles, A Mensch, C Bamford, D Singh Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, and 1 others. 2023. Mistral 7b. arxiv. *arXiv preprint arXiv:2310.06825*, 10.

Teruno Kajiura, Shiho Takano, Tatsuya Hiraoka, and Kimio Kuramitsu. 2023. Vocabulary replacement in SentencePiece for domain adaptation. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 645–652, Hong Kong, China. Association for Computational Linguistics.

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

Artur Kiulian, Anton Polishko, Mykola Khandoga, Yevhen Kostiuk, Guillermo Gabrielli, ukasz Gagała, Fadi Zaraket, Qusai Abu Obaida, Hrishikesh Garud, Wendy Wing Yee Mak, and 1 others. 2024. From english-centric to effective bilingual: Llms with custom tokenizers for underrepresented languages. *arXiv preprint arXiv:2410.18836*.

Fajri Koto, Haonan Li, Sara Shatnawi, Jad Doughman, Abdelrahman Boda Sadallah, Aisha Alraeesi, Khalid Almubarak, Zaid Alyafeai, Neha Sengupta, Shady Shehata, Nizar Habash, Preslav Nakov, and Timothy Baldwin. 2024. Arabicmmlu: Assessing massive multitask language understanding in arabic. *Preprint*, arXiv:2402.12840.

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.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, and 13 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haoran Lian, Yizhe Xiong, Jianwei Niu, Shasha Mo, Zhenpeng Su, Zijia Lin, Hui Chen, Jungong Han, and Guiguang Ding. 2025. Scaffold-bpe: Enhancing byte pair encoding for large language models with simple and effective scaffold token removal. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24539–24548.

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.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2024. Scaling neural machine translation to 200 languages. *Nature*, 630(8018):841–846.

Yuta Nozaki, Dai Nakashima, Ryo Sato, Naoki Asaba, and Shintaro Kawamura. 2025. VRCP: Vocabulary replacement continued pretraining for efficient multilingual language models. In *Proceedings of the Second Workshop on Scaling Up Multilingual & Multi-Cultural Evaluation*, pages 48–59, Abu Dhabi. Association for Computational Linguistics.

Vaidehi Patil, Partha Talukdar, and Sunita Sarawagi. 2022. Overlap-based vocabulary generation improves cross-lingual transfer among related languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 219–233, Dublin, Ireland. Association for Computational Linguistics.

Aleksandar Petrov, Emanuele La Malfa, Philip H.S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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.

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

Shivalika Singh, Freddie Vargus, Daniel D'souza, Börje Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura O'Mahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergun, Ifeoma Okoh, and 14 others. 2024. Aya dataset: An open-access collection for multilingual instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11521–11567, Bangkok, Thailand. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Elia Ackermann, Noëmi Aepli, Hamid Aghaei, Željko Agić, Amir Ahmadi, Lars Ahrenberg, Chika Kennedy Ajede, Gabrielė Aleksandravičiūtė, Ika Alfina, Lene Antonsen, Katya Aplonova, Angelina Aquino, Carolina Aragon, Maria Jesus Aranzabe, Hórunn Arnardóttir, Gashaw Arutie, and 397 others. 2020. Universal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL).

A Arabic Unicode Normalization

We applied a custom text normalization procedure to our Arabic training corpus introduced in Section 7.1. We began by extracting all characters in the Arabic Unicode range, along with their frequencies and the top 10 words in which they appeared. Characters were sorted by code point to distinguish base forms from homoglyphs. For each character, we assigned a normalized form, either the character itself or its base form, based on its usage patterns. If replacing a character with its base form altered the visual or orthographic integrity of common words, we preserved the original character; otherwise, we adopted the base form. We also corrected cases where visually similar non-Arabic characters were used in Arabic contexts, replacing them with their appropriate Arabic equivalents.

Although such replacements may introduce errors in non-Arabic contexts, our focus is on maximizing consistency within Arabic text. To evaluate the quality of our normalization, we compared it against Unicode normalization forms NFC, NFD, NFKC, and NFKD. Our method aligned most closely with NFKC. In cases of disagreement, we manually reviewed the affected characters, prioritizing linguistic and visual consistency in Arabic usage, especially where NFKC failed to collapse characters to a canonical base form.

B Model Training Hyperparameters

We conducted all experiments on 8 Nvidia H100 SXM5 GPUs. We used the HuggingFace transformers (Wolf et al., 2020), tokenizers (HuggingFace, 2021), and datasets (Lhoest et al., 2021) libraries. We used the same hyperparameters for all experiments as detailed in Table 5.

Hyper-parameter	**Value**
Model precision	bfloat16 (bf16)
Optimizer	AdamW
Peak learning rate	2.0×10^{-5}
LR scheduler	Cosine decay
Warm-up ratio	10%
Gradient accumulation steps	64
Gradient checkpointing	Enabled (non-reentrant)
Sequence length	2048 tokens
Train batch size (per device)	4

Table 5: Key training hyper-parameters for the continued-pretraining of Llama-3.1-8B.