One Script Instead of Hundreds?
On Pretraining Romanized Encoder Language Models

Benedikt Ebing*, **Lennart Keller*** and **Goran Glavaš**
Center for Artificial Intelligence and Data Science
University of Würzburg
`{firstname}.{lastname}@uni-wuerzburg.de`

Abstract

Exposing latent lexical overlap, script romanization has emerged as an effective strategy for improving cross-lingual transfer (XLT) in multilingual language models (mLMs). Most prior work, however, focused on setups that favor romanization the most: **(1)** transfer from high-resource Latin-script to low-resource non-Latin-script languages and/or **(2)** between genealogically closely related languages with different scripts. It thus remains unclear whether romanization is a good representation choice for *pretraining* general-purpose mLMs, or, more precisely, if information loss associated with romanization harms performance for high-resource languages. We address this gap by pretraining encoder LMs from scratch on both romanized and original texts for six typologically diverse high-resource languages, investigating two potential sources of degradation: **(i)** loss of script-specific information and **(ii)** dilution of language-specific representations from increased subword overlap. Using two romanizers with different fidelity profiles, we observe negligible performance loss for languages with segmental scripts, whereas languages with morphosyllabic scripts (Chinese and Japanese) suffer degradation that higher-fidelity romanization mitigates but cannot fully recover. Importantly, comparing monolingual LMs with their mLM counterpart, we find no evidence that increased subword overlap dilutes language-specific representations. We further show that romanization improves encoding efficiency (i.e., fertility) for segmental scripts at a negligible performance cost.

1 Motivation and Background

Encoder LMs still outperform their comparable-size (or even larger) decoder-only counterparts on many natural language understanding (NLU) tasks, like classification, clustering, or retrieval (Weller et al., 2025; Gisserot-Boukhlef et al., 2025); and training both monolingual (Portes et al., 2023; Breton et al., 2025; Ehrmanntraut et al., 2025; Warner et al., 2025) and multilingual encoder LMs has recently regained traction (Marone et al., 2025; Boizard et al., 2025).

Multilingual LMs (mLMs) have long been used as primary vehicles for downstream cross-lingual transfer (XLT) (Dufter and Schütze, 2020; Muller et al., 2021b): fine-tuned on task data in a source language, they perform the same task in a target language with few (Xu and Murray, 2022; Schmidt et al., 2022a, *few-shot XLT*) or no labeled task examples (Pires et al., 2019; Lauscher et al., 2020; Schmidt et al., 2023, *zero-shot XLT*).

More recently, *romanization*—converting non-Latin scripts to the Latin alphabet—has emerged as a tool to break the script barrier in multilingual language learning (Ma et al., 2025; Xhelili et al., 2024; J et al., 2024). Mapping diverse scripts to a common symbolic representation, romanization exposes latent lexical overlap and thus directly facilitates XLT with mLMs (Amrhein and Sennrich, 2020; Dabre et al., 2022; Moosa et al., 2023). Romanization is particularly suited for encoder LMs, as we can, unlike in text generation with decoders, avoid the non-trivial additional step of re-mapping the romanized text back to the native script. Still, being non-injective, romanization is inherently lossy: multiple native-script symbols may map to the same (sequences of) Latin characters. This creates a trade-off between enhanced cross-lingual transfer through script unification and the potential loss of script-specific information.

Prior research on romanization in LMs shows promising results, but studies have predominantly focused on XLT and two scenarios that arguably favor romanization the most: **(1)** transfer within a language group with known (lexical) similarities, such as genealogically related languages with different scripts (e.g., languages of the Indo-Aryan

---

*Equal contribution.

Findings of the Association for Computational Linguistics: ACL 2026, pages 38291–38307

family) (Khemchandani et al., 2021; Dhamecha et al., 2021; Liu et al., 2025b) , or **(2)** transfer from a high-resource Latin-script language (usually English) to the text-lean languages from the long tail with non-Latin scripts (Muller et al., 2021a; Purkayastha et al., 2023; Liu et al., 2024, 2025a). These are precisely the setups in which the benefit of unlocking lexical overlap outweighs the drawback of information loss *the most*. What remains unclear, however, is how romanization affects the high-resource languages, for which training data is abundant and thus **(a)** the benefits of exposing lexical links to other languages much smaller and **(b)** total information loss much larger, as it scales with the training data. This gap is critical for assessing whether romanization is a good representation choice for pretraining *general-purpose* mLMs, which need to exhibit strong in-language performance for high-resource languages as much as robust XLT performance.

Focused Contribution. In this controlled study, we address this gap by quantifying the downstream performance gaps for high-resource languages between encoder LMs pretrained on romanized vs. native-script data. We (pre)train from scratch both monolingual LMs, to isolate the impact of the loss of script-specific information, and corresponding mLMs, to assess whether increased vocabulary overlap leads to weaker language-specific representations. We experiment with six typologically diverse high-resource languages that span six writing systems, and investigate two romanization variants, produced by two popular romanization tools with different fidelity characteristics. Results on five established downstream tasks reveal that romanized LMs show *small to negligible performance losses* compared to native script counterparts, in most cases within one standard deviation. While the losses are more pronounced for Chinese and Japanese, two of our experimental languages with morphosyllabic scripts, we show that higher-fidelity romanization can partially mitigate them. Finally, we analyze the trade-off between information loss and tokenizer's fertility gain: we find that romanized models yield substantially better fertility, while performing only marginally worse than their native-script counterparts.

2 Romanization Approaches

Most existing work pre-selects the romanization tool. In contrast, to isolate the downstream effects of the romanizer choice, we compare two widely-used romanizers, which give rise to a trade-off between romanization fidelity and script coverage:

URoman. `URoman` (Hermjakob et al., 2018) aims to maximize script coverage by combining (i) parsing character descriptions from the Unicode codepoint database to derive phonetic Latin correspondences, with (ii) manual corrections where this heuristic fails and (iii) established script-specific mappings for scripts where Unicode descriptions do not provide reliable cues for Latin equivalents (e.g., Chinese characters). As a result, `URoman` offers a near-universal, ASCII-constrained romanization.

UConv. `UConv` bundles standardized romanization schemes, incorporating both transliteration and transcription-focused approaches. While its script coverage is narrower than that of `URoman`, its adherence to standards involves the use of diacritics, thereby enhancing romanization fidelity by increasing phonemic and orthographic distinctions. We provide further details and romanization examples in Appendix A.

3 Experiments

We isolate the downstream effects of romanization for six typologically diverse high-resource languages with different writing systems: Arabic, Chinese, Hindi, Japanese, Russian, and Vietnamese.[1] Compared to native scripts, romanization introduces two possible causes of degradation: **(i)** loss of script-specific information and **(ii)** dilution of language-specific representations from increased subword overlap. Note that the nature of the effect of larger vocabulary overlap likely depends on the *resourceness* of involved languages: while it drives the positive XLT from high- to low-resource languages, it may weaken language-specific representations of high-resource languages, where semantically divergent yet frequently occurring tokens are forced into shared representations. We train the following LM variants.

Monolingual. We train monolingual encoders from scratch to isolate the effect of the script-specific information loss, e.g., the loss of tonal information. We obtain two romanized LMs for each language: **MonoUroman** and **MonoUconv**, trained on corpora romanized with `URoman` and

---

[1] Vietnamese uses Latin script but encodes tonal information through diacritics, which `URoman` removes.

	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Seq. Avg.	Token Avg.	Avg.
MonoNat	$80.3_{\pm 0.6}$	$78.4_{\pm 0.7}$	$81.0_{\pm 1.6}$	$74.9_{\pm 1.2}$	$81.8_{\pm 0.6}$	$82.0_{\pm 0.7}$	$82.2_{\pm 1.0}$	$76.6_{\pm 1.0}$	$79.7_{\pm 1.0}$
MonoUroman	$80.7_{\pm 0.8}$	$74.1_{\pm 1.0}^{*}$	$80.7_{\pm 0.8}$	$71.7_{\pm 1.1}$	$81.6_{\pm 0.6}$	$81.9_{\pm 0.7}$	$81.5_{\pm 0.8}$	$74.7_{\pm 0.8}$	$78.4_{\pm 0.8}$
MonoUconv	$80.3_{\pm 0.8}$	$76.3_{\pm 1.0}$	$80.9_{\pm 0.5}$	$73.6_{\pm 0.9}$	$82.0_{\pm 0.5}$	$82.0_{\pm 0.7}$	$81.9_{\pm 0.8}$	$75.8_{\pm 0.8}$	$79.2_{\pm 0.8}$
MultiNat	$77.5_{\pm 1.2}$	$76.5_{\pm 0.6}$	$77.6_{\pm 1.6}$	$72.0_{\pm 1.3}$	$78.8_{\pm 1.2}$	$79.3_{\pm 1.1}$	$79.3_{\pm 1.2}$	$74.0_{\pm 1.2}$	$76.9_{\pm 1.2}$
MultiRom	$77.3_{\pm 1.0}$	$74.5_{\pm 1.1}$	$76.5_{\pm 1.0}$	$69.3_{\pm 0.9}^{*}$	$78.2_{\pm 2.0}$	$79.0_{\pm 1.4}$	$78.6_{\pm 1.3}$	$72.3_{\pm 1.3}$	$75.8_{\pm 1.3}^{*}$

Table 1: In-language performance averaged across five tasks with standard deviation (±). *Seq. Avg.* and *Token Avg.* columns aggregate sequence and token classification results. * indicates a statistical signifcant difference to MonoNat and MultiNat at $p < 0.05$.

`UConv`, respectively (for each, we also train a dedicated tokenizer). For a fair comparison, we train a corresponding model on the original, native-script data of each language: **MonoNat**. Securing *information parity* in pretraining is key here: for each language, all models are exposed to exactly the same documents and pretrained with the same, fixed computational budget. We report token counts in Appendix A.

Multilingual. To test if romanization's increase in lexical overlap leads to weaker language-specific representations, we train multilingual LMs. The results of monolingual models (see §4) informed the choice of romanization tool for each language: we train **MultiRom** on mixed romanized data, using the better-performing romanizer per language: `UConv` for Chinese, Japanese, and Vietnamese, and `URoman` for Arabic, Hindi, and Russian. We compare **MultiRom** against **MultiNat**, trained on the concatenated original, native-script corpora.

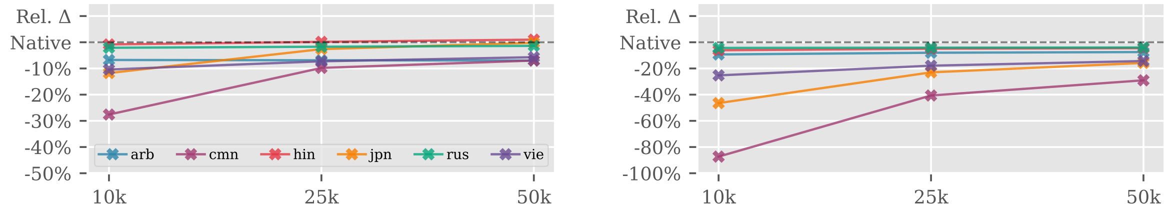
Model Training. We sample pretraining data from the Fineweb-2 corpus (Penedo et al., 2025), containing cleaned, deduplicated, and filtered CommonCrawl texts. We train all LMs with the ModernBERT architecture (Warner et al., 2025) in `Base` size (149M parameters). We train BPE tokenizers using the `sentencepiece` library (Kudo and Richardson, 2018) on 100k randomly sampled documents from the pretraining corpus, with a vocabulary size of 50 048 (unless stated otherwise). We provide further pretraining details in Appendix A.

Fine-Tuning. We fine-tune for (and evaluate on) five standard tasks, mixing sequence classification: natural language inference (XNLI) (Conneau et al., 2018), topic classification (SIB200) (Adelani et al., 2024), and intent classification (MASSIVE) (FitzGerald et al., 2023; Bastianelli et al., 2020); with token classification: named entity recognition (WikiAnn) (Pan et al., 2017) and slot filling (MASSIVE) (FitzGerald et al., 2023). For each experiment, we execute five fine-tuning runs with distinct seeds and report the mean performance. We provide further fine-tuning details in Appendix B.

4 Results and Analyses

Monolingual Models. Table 1 outlines our main results. On average, the information loss introduced by romanization has a limited downstream effect: compared to the native-script models (MonoNat), MonoUroman counterparts display a marginal average performance drop (-1.3%), whereas MonoUconv variants perform essentially on par (-0.5%, less than one standard deviation). Our results reveal two groups of writing systems. For languages with largely segmental scripts (Arabic, Hindi, Russian, and Vietnamese), our romanized models perform on par with MonoNat for both romanizers. Interestingly, we do not see a notable performance drop for Vietnamese for MonoUroman, despite `URoman` removing suprasegmental tonal information. These results are encouraging, as they suggest that romanization is a safe choice for segmental scripts (i.e., Alphabets, Abjads, and Abugidas), which account for the largest number of world languages, by far.

In the second group, we find Japanese and Chinese, two writing systems with substantial morphosyllabic components: here, romanization introduces larger downstream losses, as well as more pronounced differences between the two types of romanization: MonoUconv outperforms MonoUroman by 2.2% for Chinese (1.9% for Japanese), but falls behind MonoNat by 2.1% for Chinese (1.3% for Japanese). Here, the extent of performance loss seems also dependent on the type of task, with romanized models trailing by a wider margin on token classification tasks. This aligns with Schmidt et al. (2022b), who show that token classification models derive useful features from tokens themselves rather than from the context—an effect that is amplified in WikiAnn, where con-



(a) (b)

Figure 1: Fertility vs. token merging for monolingual models with `URoman`: **(a)** relative change in fertility (lower is better) for MonoUroman compared to MonoNat; **(b)** relative change in vocabulary size (higher is better) of MonoNat when romanizing its vocabulary with `URoman` (i.e., subwords conflated due to same romanization).

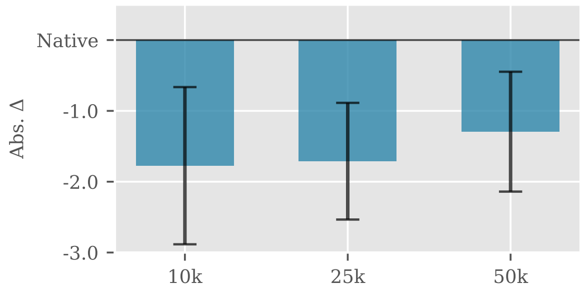


Figure 2: Absolute performance difference (avg. across all six languages and five tasks) between MonoNat and MonoUroman for different vocabulary sizes.

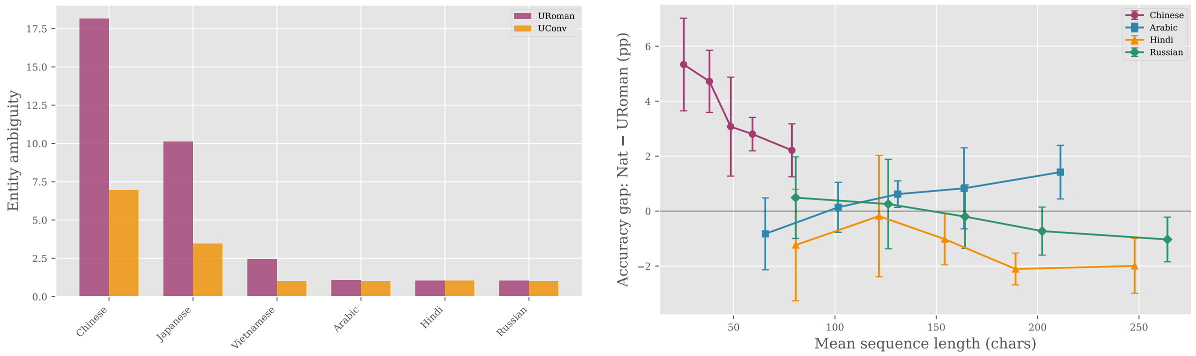
texts are often very short. Furthermore, Chinese and Japanese are typically evaluated at the character level, where romanization introduces far more ambiguity than for longer character sequences. For sequence classification, however, we observe that both types of romanization perform comparably to MonoNat: we attribute this to the fact that longer sequences provide sufficient contextual information to resolve token-level ambiguities introduced by romanization.

Multilingual Models. The trends observed for the monolingual models largely hold: **(i)** MultiRom performs within one standard deviation of MultiNat, with on par performance for **(ii)** Arabic, Hindi, Russian, and Vietnamese; **(iii)** gaps between MultiNat and MultiRom are consistently smaller on sequence classification tasks than on token classification tasks. In sum, we observe no evidence that increased subword overlap, introduced by romanization, leads to weaker language-specific representations in our multilingual models. The in-language performance differences are similar to those observed in the monolingual setting.

Tokenizer Analysis. Looking through the lens of tokenization, romanization entails a trade-off too. On the one hand, it reduces character diversity, which allows the tokenizer to learn longer subwords, thereby yielding lower fertility (i.e., fewer tokens for the same text). On the other hand, information loss incurred by romanization may adversely affect the model's downstream performance by increasing ambiguity. To approximate the information loss of romanization, we measure *token collapse*, the number of tokens in the vocabulary of MonoNat that share the romanized form with some other token(s). Figure 1a illustrates this trade-off between fertility and ambiguity for MonoUroman across vocabulary sizes: for larger vocabularies, romanized models suffer a smaller information loss w.r.t. the corresponding native-script model, but also approach the fertility of the native-script model; conversely, smaller vocabularies imply lower fertility but also larger information loss. For Arabic, Hindi, and Russian, both fertility and information loss are relatively stable across vocabulary sizes; for Chinese and Japanese, however, we see significantly larger information loss and fertility gains with vocabulary reduction; Vietnamese falls in between the two groups. These patterns largely hold for multilingual models too (see Appendix G), albeit with script-specific fertility differences.

Crucially, as illustrated in Figure 2, the downstream performance seems to be quite robust to vocabulary reduction (i.e., insensitive to increased information loss): even the models with a vocabulary of 10K tokens exhibit only marginal performance degradation compared to their native-script counterparts, despite substantial token collapse due to romanization (see Figure 1a). Arabic and Vietnamese illustrate this most clearly: both achieve substantially lower fertility (-7.0% and -5.7%, respectively), while matching MonoNat on downstream tasks ($+0.4\%$ and -0.1%). This suggests that for segmental scripts, romanization offers a favorable trade-off: improved encoding efficiency at a negligible cost to task performance.

Error Analysis. To understand the stark degradation for morphosyllabic scripts (i.e., Chinese and Japanese), we analyze romanization ambiguity on



(a)

(b)

Figure 3: Error analysis for Chinese and Japanese: **(a)** Average WikiAnn entity-token ambiguity: the number of distinct native script forms that collapse onto the same romanized form. **(b)** XNLI accuracy gap (MonoNat vs. MultiRom) across input-length bins, averaged over 5 seeds.

the WikiAnn NER test sets. For each entity token, we measure *entity ambiguity*: the average number of distinct native-script characters that share the same romanized form (Figure 3a). Entity tokens average 18.1 (Chinese) and 10.1 (Japanese) collisions per form under `URoman`, dropping to 7.0 and 3.5 under `UConv`, consistent with the token collapse observed in Figure 1a. This reflects the many-to-one nature of romanizing morphosyllabic characters: e.g., 78 Chinese characters in the WikiAnn test set collapse onto *ji* under romanization with `URoman`. In contrast, segmental scripts exhibit near-zero ambiguity, as their characters romanize nearly one-to-one. The higher entity ambiguity of Chinese relative to Japanese aligns with the larger downstream gap in Table 1, and `UConv`'s lower ambiguity mirrors its smaller performance penalty. We hypothesize that token classification is more sensitive to this ambiguity, as it requires per-token decisions; in sequence classification, longer inputs may provide sufficient context to compensate. Figure 3b supports this: for Chinese XNLI, the accuracy gap between MonoNat and MonoUroman narrows monotonically from +5.3% for the shortest sequences to +2.2% for the longest, whereas segmental scripts do not show a consistent pattern.

5 Conclusion

We studied the impact of romanization on encoder LM performance across six typologically diverse languages under controlled pretraining conditions. We find that romanization incurs negligible performance loss for segmental scripts while offering substantial fertility gains—a favorable trade-off for efficient multilingual LM pretraining. For morphosyllabic scripts (Chinese, Japanese), higher-fidelity romanizers mitigate but do not fully eliminate the performance gap, indicating that such writing systems require special attention. Importantly, we find that increased subword overlap due to romanization does not lead to degradation of in-language performance. Combined with known benefits for low-resource XLT, this warrants pretraining of massively multilingual *romanized* LMs as they can improve cross-lingual sharing without sacrificing monolingual performance for high-resource languages.

6 Limitations

Language and Script Coverage. Given our focus on high-resource languages and computational budget constraints, we investigate only six of the world's approximately 293 writing systems. While the six corresponding languages greatly vary in terms of linguistic typology and account for a substantial portion of the world's population, many script families remain unexplored. Furthermore, for each writing system, we examine only a single (arguably most representative) language. We thus do not study the potentially interesting variation in how romanization affects languages of the same writing system; a comparison between typologically different and genealogically unrelated languages that share the same script and are ideally of "similar *resourceness*" (e.g., Persian or Urdu vs. Arabic for the Arabic script, or Kazakh vs. Mongolian for the Cyrillic script).

Scale. Our models are trained at the `Base` size with fewer pretraining tokens than state-of-the-art multilingual models (Marone et al., 2025; Boizard et al., 2025; Conneau et al., 2020). While this setup allows for controlled comparisons within our computational budget, scaling behavior at larger model sizes and with more training data may, in principle, differ. However, we think it is unlikely that training at larger scales would fundamentally alter our findings, as our scale already suffices to expose nuanced differences in romanization effects (e.g., for segmental vs. morphosyllabic scripts; or for `Uroman` vs. `UConv` romanization of morphosyllabic scripts).

Cross-lingual transfer. Our analysis focuses on whether language-specific representations are diluted to an extent that harms in-language performance, i.e., we compare each language's performance in MultiRom against its native script counterpart MultiNat. We do not evaluate cross-lingual transfer, and therefore cannot speak to whether the representations learned for high-resource languages remain effective as sources for zero-shot transfer to other languages. We consider this an acceptable scope restriction: high-resource languages, by definition, come with sufficient training data, and the practical motivation for using them as transfer sources—rather than training directly in the target setting—is weaker than for low-resource scenarios, which have been the primary focus of prior romanization work. We leave a systematic study of how romanization affects the transfer properties of high-resource source languages to future work.

7 Acknowledgements

Pretraining experiments were conducted using computing resources provided by WestAI on the supercomputer JURECA (Thörnig, 2021) at Jülich Supercomputing Centre (JSC).

Fine-tuning experiments were performed using Julia 2. Julia 2 was funded as DFG project as "Forschungsgroßgerät nach Art 91b GG" under INST 93/1145-1 FUGG.

This work was supported by the Alcatel-Lucent Stiftung and Deutsches Stiftungszentrum through the grant "Equitably Fair and Trustworthy Language Technology" (EQUIFAIR, Grant Nr. T0067/43110/23).

References

David Ifeoluwa Adelani, Hannah Liu, Xiaoyu Shen, Nikita Vassilyev, Jesujoba O. Alabi, Yanke Mao, Haonan Gao, and En-Shiun Annie Lee. 2024. SIB-200: A simple, inclusive, and big evaluation dataset for topic classification in 200+ languages and dialects. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 226–245, St. Julian's, Malta. Association for Computational Linguistics.

Chantal Amrhein and Rico Sennrich. 2020. On Romanization for model transfer between scripts in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2461–2469, Online. Association for Computational Linguistics.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. SLURP: A spoken language understanding resource package. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7252–7262, Online. Association for Computational Linguistics.

Nicolas Boizard, Hippolyte Gisserot-Boukhlef, Duarte M. Alves, André Martins, Ayoub Hammal, Caio Corro, Céline Hudelot, Emmanuel Malherbe, Etienne Malaboeuf, Fanny Jourdan, Gabriel Hautreux, João Alves, Kevin El-Haddad, Manuel Faysse, Maxime Peyrard, Nuno M. Guerreiro, Patrick Fernandes, Ricardo Rei, and Pierre Colombo. 2025. Eurobert: Scaling multilingual encoders for european languages. *Preprint*, arXiv:2503.05500.

Lola Le Breton, Quentin Fournier, Mariam El Mezouar, John X. Morris, and Sarath Chandar. 2025. Neobert: A next-generation bert. *Preprint*, arXiv:2502.19587.

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

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh Khapra, and Pratyush Kumar. 2022. IndicBART: A pre-trained model for indic natural language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1849–1863, Dublin, Ireland. Association for Computational Linguistics.

Tejas Dhamecha, Rudra Murthy, Samarth Bharadwaj, Karthik Sankaranarayanan, and Pushpak Bhattacharyya. 2021. Role of Language Relatedness in

Multilingual Fine-tuning of Language Models: A Case Study in Indo-Aryan Languages. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8584–8595, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Philipp Dufter and Hinrich Schütze. 2020. Identifying elements essential for BERT's multilinguality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4423–4437, Online. Association for Computational Linguistics.

Anton Ehrmanntraut, Julia Wunderle, Jan Pfister, Fotis Jannidis, and Andreas Hotho. 2025. Moderngbert: German-only 1b encoder model trained from scratch. *arXiv preprint arXiv:2505.13136*.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2023. MASSIVE: A 1M-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4277–4302, Toronto, Canada. Association for Computational Linguistics.

Hippolyte Gisserot-Boukhlef, Nicolas Boizard, Manuel Faysse, Duarte M. Alves, Emmanuel Malherbe, André F. T. Martins, Céline Hudelot, and Pierre Colombo. 2025. Should we still pretrain encoders with masked language modeling? *Preprint*, arXiv:2507.00994.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train BERT with an academic budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10644–10652, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jaavid J, Raj Dabre, Aswanth M, Jay Gala, Thanmay Jayakumar, Ratish Puduppully, and Anoop Kunchukuttan. 2024. RomanSetu: Efficiently unlocking multilingual capabilities of large language models via Romanization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15593–15615, Bangkok, Thailand. Association for Computational Linguistics.

Yash Khemchandani, Sarvesh Mehtani, Vaidehi Patil, Abhijeet Awasthi, Partha Talukdar, and Sunita Sarawagi. 2021. Exploiting language relatedness for low web-resource language model adaptation: An Indic languages study. 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 1312–1323, Online. Association for Computational Linguistics.

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.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Yihong Liu, Chunlan Ma, Haotian Ye, and Hinrich Schuetze. 2024. TransliCo: A contrastive learning framework to address the script barrier in multilingual pretrained language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2476–2499, Bangkok, Thailand. Association for Computational Linguistics.

Yihong Liu, Chunlan Ma, Haotian Ye, and Hinrich Schütze. 2025a. TransMI: A framework to create strong baselines from multilingual pretrained language models for transliterated data. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 469–495, Abu Dhabi, UAE. Association for Computational Linguistics.

Yihong Liu, Mingyang Wang, Amir Hossein Kargaran, Ayyoob ImaniGooghari, Orgest Xhelili, Haotian Ye, Chunlan Ma, François Yvon, and Hinrich Schütze. 2025b. How transliterations improve crosslingual alignment. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2417–2433, Abu Dhabi, UAE. Association for Computational Linguistics.

Ruixuan Luo, Jingjing Xu, Yi Zhang, Zhiyuan Zhang, Xuancheng Ren, and Xu Sun. 2019. Pkuseg: A toolkit for multi-domain chinese word segmentation. *CoRR*, abs/1906.11455.

Chunlan Ma, Yihong Liu, Haotian Ye, and Hinrich Schuetze. 2025. Exploring the role of transliteration in in-context learning for low-resource languages written in non-Latin scripts. In *Proceedings of the 5th Workshop on Multilingual Representation Learning (MRL 2025)*, pages 397–410, Suzhuo, China. Association for Computational Linguistics.

Marc Marone, Orion Weller, William Fleshman, Eugene Yang, Dawn Lawrie, and Benjamin Van

Durme. 2025. mmbert: A modern multilingual encoder with annealed language learning. *Preprint*, arXiv:2509.06888.

Ibraheem Muhammad Moosa, Mahmud Elahi Akhter, and Ashfia Binte Habib. 2023. Does transliteration help multilingual language modeling? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 670–685, Dubrovnik, Croatia. Association for Computational Linguistics.

Benjamin Muller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021a. When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 448–462, Online. Association for Computational Linguistics.

Benjamin Muller, Yanai Elazar, Benoît Sagot, and Djamé Seddah. 2021b. First align, then predict: Understanding the cross-lingual ability of multilingual BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2214–2231, Online. Association for Computational Linguistics.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. Fineweb2: One pipeline to scale them all – adapting pre-training data processing to every language. *Preprint*, arXiv:2506.20920.

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

Jacob Portes, Alexander Trott, Sam Havens, Daniel King, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. 2023. Mosaicbert: A bidirectional encoder optimized for fast pretraining. *NeuRIPS https://openreview.net/pdf?id=5zipcfLC2Z*.

Sukannya Purkayastha, Sebastian Ruder, Jonas Pfeiffer, Iryna Gurevych, and Ivan Vulić. 2023. Romanization-based large-scale adaptation of multilingual language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7996–8005, Singapore. Association for Computational Linguistics.

Fabian David Schmidt, Ivan Vulić, and Goran Glavaš. 2022a. Don't stop fine-tuning: On training regimes for few-shot cross-lingual transfer with multilingual language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10725–10742, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Fabian David Schmidt, Ivan Vulić, and Goran Glavaš. 2022b. SLICER: Sliced fine-tuning for low-resource cross-lingual transfer for named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10775–10785, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Fabian David Schmidt, Ivan Vulić, and Goran Glavaš. 2023. Free lunch: Robust cross-lingual transfer via model checkpoint averaging. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5712–5730, Toronto, Canada. Association for Computational Linguistics.

Kazuma Takaoka, Sorami Hisamoto, Noriko Kawahara, Miho Sakamoto, Yoshitaka Uchida, and Yuji Matsumoto. 2018. Sudachi: a japanese tokenizer for business. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Philipp Thörnig. 2021. Jureca: Data centric and booster modules implementing the modular supercomputing architecture at jülich supercomputing centre. *Journal of large-scale research facilities JLSRF*, 7.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.

Orion Weller, Kathryn Ricci, Marc Marone, Antoine Chaffin, Dawn Lawrie, and Benjamin Van Durme. 2025. Seq vs seq: An open suite of paired encoders and decoders. *Preprint*, arXiv:2507.11412.

Orgest Xhelili, Yihong Liu, and Hinrich Schuetze. 2024. Breaking the script barrier in multilingual pre-trained language models with transliteration-based post-training alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11283–11296, Miami, Florida, USA. Association for Computational Linguistics.

Haoran Xu and Kenton Murray. 2022. Por qué não utiliser alla språk? mixed training with gradient optimization in few-shot cross-lingual transfer. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2043–2059, Seattle, United States. Association for Computational Linguistics.

A Pretraining

Data. We source the pretraining data from the Fineweb2 corpus (Penedo et al., 2025) through the Hugging Face library. The dataset is already cleaned, deduplicated, and filtered. We do not apply further preprocessing. For a fair comparison, we expose all models within a language to the exact same information (i.e., the same documents). We randomly sample documents and ensure that the number of training tokens is at most equal to the total number of tokens seen during training (i.e., each model trains at least one epoch on all documents). The resulting number of training tokens per model is shown in Table 2; the last two lines additionally report the per-language token counts for the multilingual models, which we construct by concatenating the monolingual corpora and randomly sampling batches without any over- or undersampling. We aim for a roughly uniform distribution across languages, but ensuring document-level parity in combination with the differing fidelity of tokenizers across scenarios makes this a non-trivial problem. For example, in Chinese, our document selection yields 21.61 billion training tokens for MonoNat 10K, no-ws, but only 13.27 billion for MonoUroman, ws, 50K. Fineweb2 is licensed under ODC-By 1.0.

Romanization Tools. To efficiently romanize pretraining-scale data, we use a Rust reimplementation of `URoman`, which yields a roughly 27x speedup in throughput compared with the original Python implementation. Both versions are licensed under the Apache License Version 2.0.

`UConv` is part of the International Components for Unicode (ICU) (https://icu.unicode.org), which is open-source-licensed under the Unicode License. For `UConv` romanization, we use the default `any-to-Latin` preset that dynamically selects a suitable romanization scheme. The scheme selection of `UConv` generally follows the recommendations of the Working Group on Romanization Systems of the United Nations Group of Experts on Geographical Names (UNGEGN), and thus utilizes both transcription- and transliteration-focused approaches. Transcription-focused Romanization generally assumes a reference pronunciation of Latin characters, whereas transliteration aims to create invertible character mappings between the source script and the Latin alphabet. Thus, the Russian name “**Чайковский**” appears as “**Ch**aĭkovskiĭ” under the ALA–LC transcription scheme aimed at English speakers, whereas the ISO 9 transliteration scheme for Cyrillic yields “**Č**ajkovskij”. In this work, the specific schemes employed via `UConv` were: ADEGN (Arabic), Pinyin (Chinese), ISO 15919 (Hindi), Hepburn (Japanese), and ISO 9 (Russian). Table 3 presents example outputs for both romanizers.

Model. We train ModernBert models (Warner et al., 2025) in their `Base` variant with 149M parameters. We adopt our implementation from the ModernBert GitHub repository: https://github.com/AnswerDotAI/ModernBERT which is licensed under the Apache License 2.0. The training details are displayed in Table 4. We train all models with bfloat16 mixed precision and distributed data parallelism. Our training setup is adapted from ModernBert (Warner et al., 2025) and Izsak et al. (2021), who focus on training BERT models under constrained budget (e.g., 24 GPU hours). The model design hyperparameters are displayed in Table 5. A single pretraining run for a monolingual model required 34 GPU hours (204 GPU hours for a multilingual model). Overall, we estimate the total pretraining time to be 3000 GPU hours.

Tokenizer. Our tokenizers are trained with the sentencepiece library (Kudo and Richardson, 2018), which is licensed under Apache-2.0. We provide the used hyperparameters in Table 6. The tokenizers for the multilingual models are trained on 100 000 documents per language (i.e., 600 000 in total). The models in the main body of the paper are trained with the *Split by Whitespace*-flag set to *true*. For `UConv`, this hyperparameter choice is required as the romanizer introduces additional whitespaces during transcription. This is particularly relevant for Chinese and Japanese, where whitespaces are inserted around the romanization of almost every character. Therefore, not setting the flag would result in training a romanized model that can merge tokens only up to the level of a Chinese or Japanese character. Uroman, by contrast, does not introduce additional whitespace. Hence, we ablate the impact of this hyperparameter for Uroman models. Furthermore, we ablate pre-segmenting the Chinese and Japanese text

	Tok.	Vocab	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn
MonoNat	no-ws	10k	24.71	21.61	12.49	17.15	18.27	13.92
MonoNat	ws	10k	24.81	21.61	13.69	17.15	18.80	16.44
MonoNat	no-ws	25k	20.89	15.69	10.33	13.50	15.13	11.77
MonoNat	ws	25k	21.17	15.69	12.12	13.50	15.85	15.75
MonoNat	no-ws	50k	18.65	14.18	9.15	11.96	13.28	10.58
MonoNat	ws	50k	19.24	14.17	11.43	11.96	14.17	15.19
MonoUroman	no-ws	10k	22.96	15.77	12.34	15.13	17.85	12.45
MonoUroman	ws	10k	23.46	15.96	13.52	15.42	18.38	15.70
MonoUroman	no-ws	25k	19.41	14.09	10.30	13.11	14.83	10.89
MonoUroman	ws	25k	20.27	14.28	12.04	13.38	15.54	15.23
MonoUroman	no-ws	50k	17.32	13.09	9.20	11.90	13.07	9.95
MonoUroman	seg&ws	50k	-	19.21	-	20.92	-	-
MonoUroman	ws	50k	18.57	13.27	11.38	12.14	13.94	14.99
MonoUconv	no-ws	50k	18.60	13.43	9.37	12.00	13.12	10.58
MultiNat	no-ws	50k	25.70	16.81	13.12	15.59	18.88	14.00
MultiRom	no-ws	50k	22.79	16.25	12.15	15.59	17.41	13.51

Table 2: Pretraining token counts (in billions) for the romanized and native script models.

Arabic
الناس يولدون أحرارًا ومتساوين.
`UR`: alnas ywldwn ahrara wmtsawyn.
`UC`: ạlnạs ywldwn ảḥrạraⁿạ wmtsạwyn.

Devanagari
मनुष्य जन्म से स्वतंत्र और समान होते हैं।
`UR`: manussya janma se svatamtra aur samaan hote haim.
`UC`: manuṣya janma sē svatantra aura samāna hōtē haiṁ.

Japanese
すべての人は、生まれながら自由で平等である。
`UR`: subetenorenha, shengmarenagaraziyoudepingdengdearu.
`UC`: subeteno rénha, shēngmarenagara zì yóude píng děngdearu.

Chinese
人人生而自由平等。
`UR`: renrenshengerziyoupingdeng.
`UC`: rén rén shēng ér zì yóu píng děng.

Cyrillic
Все люди рождаются свободными и равными.
`UR`: Vse lyudi rozhdayutsya svobodnymi i ravnymi.
`UC`: Vse lûdi roždaûtsâ svobodnymi i ravnymi.

Table 3: Romanization of *"All human beings are born free and equal"* using `URoman` (`UR`) and `UConv` (`UC`).

prior to tokenizer training. For pre-segmentation, we use `pkuseg` (Luo et al., 2019) for Chinese and `Sudachi` (Takaoka et al., 2018) for Japanese (licensed under the MIT License and Apache License 2.0, respectively), both with their default configurations. After training, we convert the tokenizers to the Hugging Face format using the tokenizers library to facilitate integration into the model training pipeline. The library is licensed under Apache License 2.0.

B Downstream Fine-Tuning

	SIB200	WikiAnn	XNLI	MASSIVE
Lang	All	Other (hin)	All	All
Train. Size	700	20k (5k)	393k	11.5k
Val. Size	99	10k (1k)	2.5k	2k
Test Size	204	10k (1k)	5k	3k
Metric	Acc.	F1	Acc.	Acc./F1
License	cc-by-sa-4.0	-	-	cc-by-4.0

Table 7: Dataset sizes, metrics, and licenses for downstream fine-tuning

We begin downstream fine-tuning from the final pretraining checkpoint. For each task, we train models for 10 epochs (5 epochs for XNLI), with a batch size of 32. We use a linear learning rate schedule with 10% warmup and a weight decay of 0.01. The used optimizer is AdamW with a learning rate of 2e-5. We pick the best checkpoint based on the in-language validation performance. We employ the Hugging Face Transformers library for our downstream experiments, which is licensed under the Apache 2.0 license. For MASSIVE, we use the official downstream evaluation repository: https://github.com/alexa/massive licensed under Apache License 2.0. All fine-tuning experiments were run on L40 GPUs with 40GB of VRAM. We estimate the total compute to 2000 GPU hours. We comply with the licenses governing the downstream datasets used. Details on the dataset sizes and eval-

	Mono	Multi
Training Batches	8000	48000
Max Sequence Lenght	1024	
Batch Size	4608	
Micro Batch Size	96	
Learning Rate	1e-3	
Schedule	Linear	
Warmup	6%	
Decay	94%	
Weight Decay	1e-5	
Model Initialization	Megatron	
Dropout (attn out)	0.1	
Dropout (all other layers)	0.0	
Optimizer	Stable AdamW	
Betas	(0.9, 0.98)	
Epsilons	1e-6	
Training Hardware	4x H100	

Table 4: Training details adopted from Warner et al. (2025) and Izsak et al. (2021)

uation metrics for each dataset are provided in Table 7.

C Details on Fertility

We compute the fertility of each tokenizer on the respective test sets from Fineweb2 using the following formula:

$$Fertility_{MonoNat} = \frac{\#Tokens}{\#Words}$$

, where $\#Tokens$ refers to the total number of tokens used by the respective tokenizer (e.g., MonoNat, MonoUroman, ...) to encode the documents in the test set and $\#Words$ refers to the number of words in the native test set. Using the number of words in the original data test set ensures that the fertility scores are comparable across tokenizers. For Arabic, Hindi, Russian, and Vietnamese, we compute $\#Words$ by splitting on whitespace, whereas for Chinese and Japanese, we use the number of characters as the denominator. Finally, we compute the relative difference in fertility by:

$$1 - \frac{Fertility_{MonoUroman}}{Fertiltity_{MonoNat}}$$

D Details on Loss of information

The loss of information is computed as the difference between the number of tokens remaining after romanizing each entry in the native script tokenizer and the total number of tokens before romanization. We compute the measure by encoding the respective test sets of the Fineweb2 dataset with the native script tokenizers (i.e., MonoNat and MultiNat). We identify the unique tokens yielding $\#UniqueOrigTokens$. Then, we romanize each unique original token identified in the first step to compute $\#UniqueRomanizedTokens$. Finally, we compute the relative loss in performance by:

$$1 - \frac{\#UniqueRomanizedTokens}{\#UniqueOrigTokens}$$

	Value(s)
Vocabulary	10112 | 25088 | 50048
Unused Tokens	111 | 87 | 47
Layers	22
Hidden Size	768
Transformer Block	Pre-Norm
Activation Function	GeLU
Linear Bias	False
Attention	Multi-head
Attention Heads	12
Global Attention	Every three layers
Local Attention Window	128
Intermediate Size	1 152
GLU Expansion	2 304
Normalization	Layer Norm
Norm Epsilon	1e-5
Norm Bias	False
RoPE theta	10 000

Table 5: Model design adopted from Warner et al. (2025)

E Details on Statistical Test

We evaluate statistical significance using a one-sample t-test on per-language mean deltas against zero. For each language, we compute the mean performance difference between MonoNat (MultiNat) and the proposed romanization methods averaged across tasks, yielding six language-level deltas. We apply a one-sample t-test on these six deltas. We additionally report per-language significance by averaging each method's scores across seeds and applying a one-sample t-test across the resulting per-task deltas. Significance markers denote $p < 0.05$. We note the following limitation of this evaluation. With only six languages as independent units, the test operates at low statistical

	Value(s)
Model Type	BPE
Vocab Size	10k | 25k | 50k
Input Sentence Size	100 000
Byte Fallback	True
Normalization	NFKC
Character Coverage	0.9999
Max. Sentence Length	4192
Split by Whitespace	False | True
Split by Unicode Script	False
Split by Number	True
Split by Digits	False
Hard Vocab Limit	True

Table 6: Sentencepiece hyperparameter choices

power and is sensitive to outliers at this sample size.

F Detailed Results: Main Results and Impact of Tokenization

	Tok.	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Seq. Avg.	Token Avg.	Avg.
MonoNat	no-ws	$80.3_{\pm 0.6}$	$78.4_{\pm 0.7}$	$81.0_{\pm 1.6}$	$74.9_{\pm 1.2}$	$81.8_{\pm 0.6}$	$82.0_{\pm 0.7}$	$82.2_{\pm 1.0}$	$76.6_{\pm 1.0}$	$79.7_{\pm 1.0}$
MonoNat	ws	$79.8_{\pm 0.9}$	$78.4_{\pm 1.0}$	$81.3_{\pm 1.0}$	$74.6_{\pm 1.7}$	$81.8_{\pm 0.8}$	$82.4_{\pm 0.5}$	$82.0_{\pm 1.0}$	$76.9_{\pm 1.1}$	$79.7_{\pm 1.1}$
MonoUroman	no-ws	$80.7_{\pm 0.8}$	$74.1_{\pm 1.0}$	$80.7_{\pm 0.8}$	$71.7_{\pm 1.1}$	$81.6_{\pm 0.6}$	$81.9_{\pm 0.7}$	$81.5_{\pm 0.8}$	$74.7_{\pm 0.8}$	$78.4_{\pm 0.8}$
MonoUroman	ws	$80.1_{\pm 1.1}$	$74.0_{\pm 0.8}$	$80.8_{\pm 0.8}$	$72.1_{\pm 0.8}$	$81.7_{\pm 0.9}$	$81.3_{\pm 0.8}$	$81.1_{\pm 0.9}$	$74.9_{\pm 0.9}$	$78.3_{\pm 0.9}$
MonoUroman	seg&ws	-	$74.6_{\pm 0.8}$	-	$72.2_{\pm 1.5}$	-	-	$81.1_{\pm 1.1}$	$64.1_{\pm 1.2}$	$73.4_{\pm 1.2}$
MonoUconv	no-ws	$80.3_{\pm 0.8}$	$76.3_{\pm 1.0}$	$80.9_{\pm 0.5}$	$73.6_{\pm 0.9}$	$82.0_{\pm 0.5}$	$82.0_{\pm 0.7}$	$81.9_{\pm 0.8}$	$75.8_{\pm 0.8}$	$79.2_{\pm 0.8}$
MultiNat	no-ws	$77.5_{\pm 1.2}$	$76.5_{\pm 0.6}$	$77.6_{\pm 1.6}$	$72.0_{\pm 1.3}$	$78.8_{\pm 1.2}$	$79.3_{\pm 1.1}$	$79.3_{\pm 1.2}$	$74.0_{\pm 1.2}$	$76.9_{\pm 1.2}$
MultiRom	no-ws	$77.3_{\pm 1.0}$	$74.5_{\pm 1.1}$	$76.5_{\pm 1.0}$	$69.3_{\pm 0.9}$	$78.2_{\pm 2.0}$	$79.0_{\pm 1.4}$	$78.6_{\pm 1.3}$	$72.3_{\pm 1.3}$	$75.8_{\pm 1.3}$

Table 8: Main results averaged across our 5 evaluation tasks for monolingual and multilingual romanized models. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*, and `seg&ws` refers to pre-segmented data for tokenizer training in combination with *Split by Whitespace* set to *true*.

	Tok.	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	$86.2_{\pm 0.9}$	$89.4_{\pm 1.0}$	$85.9_{\pm 3.0}$	$87.3_{\pm 2.0}$	$86.8_{\pm 0.8}$	$87.2_{\pm 0.9}$	$87.1_{\pm 1.7}$
MonoNat	ws	$83.0_{\pm 1.2}$	$88.3_{\pm 1.3}$	$84.9_{\pm 1.7}$	$86.6_{\pm 3.3}$	$85.8_{\pm 1.3}$	$88.8_{\pm 0.7}$	$86.2_{\pm 1.8}$
MonoUroman	no-ws	$88.2_{\pm 1.3}$	$84.5_{\pm 1.7}$	$84.5_{\pm 1.1}$	$84.7_{\pm 2.0}$	$85.8_{\pm 0.7}$	$89.6_{\pm 1.1}$	$86.2_{\pm 1.4}$
MonoUroman	ws	$85.3_{\pm 2.2}$	$84.7_{\pm 1.4}$	$84.6_{\pm 1.2}$	$84.7_{\pm 1.3}$	$85.6_{\pm 1.5}$	$87.9_{\pm 1.1}$	$85.5_{\pm 1.5}$
MonoUroman	seg&ws	-	$87.1_{\pm 1.0}$	-	$84.8_{\pm 2.8}$	-	-	$85.9_{\pm 2.1}$
MonoUconv	no-ws	$85.5_{\pm 1.1}$	$85.7_{\pm 2.1}$	$85.8_{\pm 0.8}$	$86.4_{\pm 1.3}$	$86.8_{\pm 0.5}$	$87.2_{\pm 0.9}$	$86.2_{\pm 1.2}$
MultiNat	no-ws	$81.9_{\pm 2.3}$	$83.0_{\pm 1.0}$	$78.8_{\pm 3.3}$	$84.7_{\pm 1.8}$	$78.5_{\pm 2.2}$	$82.5_{\pm 1.9}$	$81.6_{\pm 2.2}$
MultiRom	no-ws	$82.3_{\pm 1.3}$	$82.6_{\pm 1.7}$	$76.0_{\pm 1.1}$	$82.3_{\pm 1.3}$	$79.3_{\pm 4.1}$	$81.8_{\pm 2.6}$	$80.7_{\pm 2.3}$

Table 9: SIB200 results for monolingual and multilingual romanized models. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*, and `seg&ws` refers to pre-segmented data for tokenizer training in combination with *Split by Whitespace* set to *true*.

	Tok.	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	$89.7_{\pm 0.1}$	$75.4_{\pm 0.2}$	$90.1_{\pm 0.5}$	$65.9_{\pm 0.2}$	$87.3_{\pm 0.1}$	$90.5_{\pm 0.1}$	$83.2_{\pm 0.2}$
MonoNat	ws	$89.9_{\pm 0.1}$	$75.3_{\pm 0.5}$	$91.3_{\pm 0.5}$	$65.9_{\pm 0.3}$	$87.6_{\pm 0.1}$	$91.0_{\pm 0.2}$	$83.5_{\pm 0.3}$
MonoUroman	no-ws	$89.6_{\pm 0.2}$	$68.1_{\pm 0.4}$	$89.5_{\pm 0.7}$	$59.7_{\pm 0.1}$	$87.1_{\pm 0.3}$	$89.5_{\pm 0.3}$	$80.5_{\pm 0.4}$
MonoUroman	ws	$89.7_{\pm 0.2}$	$67.9_{\pm 0.2}$	$90.4_{\pm 0.4}$	$59.3_{\pm 0.3}$	$87.4_{\pm 0.1}$	$90.2_{\pm 0.1}$	$80.8_{\pm 0.2}$
MonoUroman	seg&ws	-	$68.9_{\pm 0.2}$	-	$60.6_{\pm 0.3}$	-	-	$64.8_{\pm 0.3}$
MonoUconv	no-ws	$89.3_{\pm 0.2}$	$71.4_{\pm 0.2}$	$89.5_{\pm 0.2}$	$63.4_{\pm 0.3}$	$87.3_{\pm 0.2}$	$90.5_{\pm 0.1}$	$81.9_{\pm 0.2}$
MultiNat	no-ws	$87.6_{\pm 0.2}$	$73.8_{\pm 0.3}$	$88.3_{\pm 0.5}$	$62.6_{\pm 0.6}$	$85.8_{\pm 0.1}$	$89.3_{\pm 0.3}$	$81.2_{\pm 0.4}$
MultiRom	no-ws	$87.6_{\pm 0.2}$	$69.3_{\pm 0.5}$	$87.2_{\pm 1.3}$	$59.3_{\pm 0.1}$	$85.2_{\pm 0.2}$	$89.1_{\pm 0.1}$	$79.6_{\pm 0.6}$

Table 10: Wikiann results for monolingual and multilingual romanized models. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*, and `seg&ws` refers to pre-segmented data for tokenizer training in combination with *Split by Whitespace* set to *true*.

	Tok.	arb_Arab	cmn_Hani	hin_Deva	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	$74.6_{\pm0.6}$	$74.8_{\pm0.7}$	$72.0_{\pm1.5}$	$75.7_{\pm1.0}$	$76.7_{\pm1.1}$	$74.8_{\pm1.1}$
MonoNat	ws	$74.2_{\pm1.3}$	$75.3_{\pm1.0}$	$72.8_{\pm1.0}$	$75.6_{\pm0.9}$	$76.4_{\pm0.7}$	$74.9_{\pm1.0}$
MonoUroman	no-ws	$74.2_{\pm0.7}$	$71.2_{\pm1.2}$	$73.3_{\pm0.6}$	$75.9_{\pm0.8}$	$76.3_{\pm0.9}$	$74.2_{\pm0.9}$
MonoUroman	ws	$74.9_{\pm0.9}$	$70.7_{\pm0.8}$	$72.9_{\pm0.7}$	$75.4_{\pm1.1}$	$74.5_{\pm0.7}$	$73.7_{\pm0.9}$
MonoUroman	seg&ws	-	$70.8_{\pm1.2}$	-	-	-	$70.8_{\pm1.2}$
MonoUconv	no-ws	$74.2_{\pm1.2}$	$74.7_{\pm0.8}$	$73.0_{\pm0.7}$	$75.6_{\pm0.8}$	$76.7_{\pm1.1}$	$74.9_{\pm0.9}$
MultiNat	no-ws	$74.3_{\pm0.8}$	$74.6_{\pm0.5}$	$70.0_{\pm0.7}$	$74.9_{\pm0.3}$	$76.7_{\pm0.8}$	$74.1_{\pm0.6}$
MultiRom	no-ws	$74.3_{\pm0.6}$	$75.2_{\pm0.7}$	$70.9_{\pm0.7}$	$74.1_{\pm1.1}$	$75.3_{\pm0.8}$	$73.9_{\pm0.8}$

Table 11: XNLI results for monolingual and multilingual romanized models. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*, and `seg&ws` refers to pre-segmented data for tokenizer training in combination with *Split by Whitespace* set to *true*.

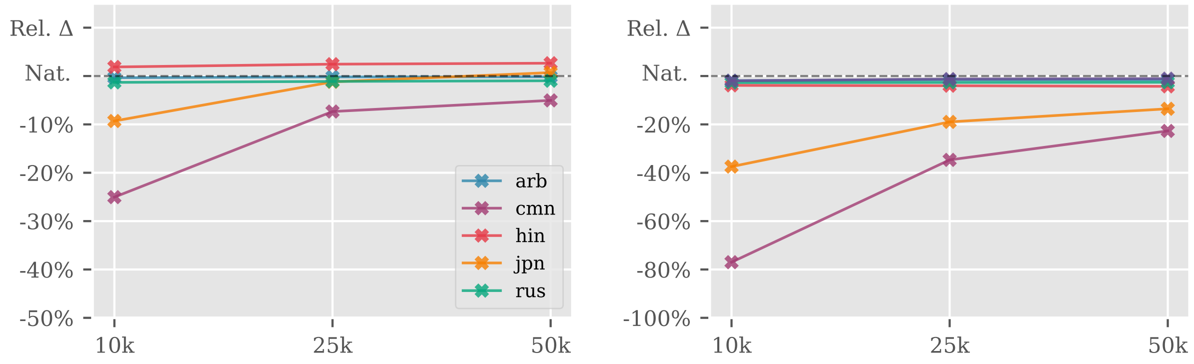
	Tok.	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	$79.8_{\pm0.5}$	$83.6_{\pm0.5}$	$85.1_{\pm0.5}$	$83.2_{\pm0.5}$	$84.8_{\pm0.5}$	$85.0_{\pm0.2}$	$83.6_{\pm0.5}$
MonoNat	ws	$79.9_{\pm0.5}$	$83.3_{\pm1.3}$	$85.1_{\pm0.5}$	$83.3_{\pm0.5}$	$85.3_{\pm0.6}$	$85.3_{\pm0.3}$	$83.7_{\pm0.7}$
MonoUroman	no-ws	$79.4_{\pm0.6}$	$81.4_{\pm0.4}$	$84.5_{\pm0.9}$	$82.1_{\pm0.3}$	$85.3_{\pm0.6}$	$84.2_{\pm0.6}$	$82.8_{\pm0.6}$
MonoUroman	ws	$79.3_{\pm0.3}$	$81.5_{\pm0.6}$	$84.6_{\pm0.6}$	$82.4_{\pm0.4}$	$85.5_{\pm0.6}$	$84.4_{\pm0.8}$	$83.0_{\pm0.6}$
MonoUroman	seg&ws	-	$81.0_{\pm0.1}$	-	$82.0_{\pm0.6}$	-	-	$81.5_{\pm0.5}$
MonoUconv	no-ws	$80.2_{\pm0.6}$	$82.4_{\pm0.3}$	$84.6_{\pm0.1}$	$82.7_{\pm0.5}$	$85.2_{\pm0.4}$	$85.0_{\pm0.2}$	$83.3_{\pm0.4}$
MultiNat	no-ws	$76.2_{\pm0.8}$	$82.5_{\pm0.3}$	$82.0_{\pm0.6}$	$81.3_{\pm0.7}$	$83.1_{\pm0.7}$	$82.6_{\pm0.7}$	$81.3_{\pm0.7}$
MultiRom	no-ws	$75.8_{\pm1.4}$	$80.9_{\pm0.4}$	$81.1_{\pm0.6}$	$79.5_{\pm0.6}$	$82.6_{\pm0.6}$	$82.5_{\pm0.7}$	$80.4_{\pm0.8}$

Table 12: MASSIVE Intent results for monolingual and multilingual romanized models. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*, and `seg&ws` refers to pre-segmented data for tokenizer training in combination with *Split by Whitespace* set to *true*.

	Tok.	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	$71.3_{\pm0.6}$	$68.6_{\pm0.9}$	$71.9_{\pm0.4}$	$63.3_{\pm1.4}$	$74.6_{\pm0.1}$	$70.5_{\pm0.6}$	$70.0_{\pm0.8}$
MonoNat	ws	$72.1_{\pm0.6}$	$69.5_{\pm0.6}$	$72.4_{\pm0.6}$	$62.6_{\pm0.8}$	$74.7_{\pm0.3}$	$70.5_{\pm0.2}$	$70.3_{\pm0.6}$
MonoUroman	no-ws	$72.2_{\pm0.8}$	$65.2_{\pm0.6}$	$71.6_{\pm0.5}$	$60.4_{\pm0.8}$	$74.2_{\pm0.4}$	$69.7_{\pm0.6}$	$68.9_{\pm0.6}$
MonoUroman	ws	$71.5_{\pm0.5}$	$65.2_{\pm0.5}$	$71.5_{\pm0.6}$	$62.0_{\pm0.8}$	$74.4_{\pm0.6}$	$69.6_{\pm0.9}$	$69.0_{\pm0.7}$
MonoUroman	seg&ws	-	$65.4_{\pm0.9}$	-	$61.4_{\pm0.5}$	-	-	$63.4_{\pm0.8}$
MonoUconv	no-ws	$72.2_{\pm0.7}$	$67.3_{\pm0.7}$	$71.7_{\pm0.5}$	$61.8_{\pm1.0}$	$74.9_{\pm0.2}$	$70.5_{\pm0.6}$	$69.7_{\pm0.7}$
MultiNat	no-ws	$67.8_{\pm1.0}$	$68.5_{\pm0.3}$	$68.7_{\pm0.7}$	$59.3_{\pm1.7}$	$71.7_{\pm1.3}$	$65.1_{\pm1.3}$	$66.8_{\pm1.1}$
MultiRom	no-ws	$66.9_{\pm1.2}$	$64.5_{\pm1.4}$	$67.1_{\pm1.1}$	$56.3_{\pm1.3}$	$69.6_{\pm1.1}$	$66.1_{\pm1.4}$	$65.1_{\pm1.2}$

Table 13: MASSIVE Slot Filling results for monolingual and multilingual romanized models. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*, and `seg&ws` refers to pre-segmented data for tokenizer training in combination with *Split by Whitespace* set to *true*.

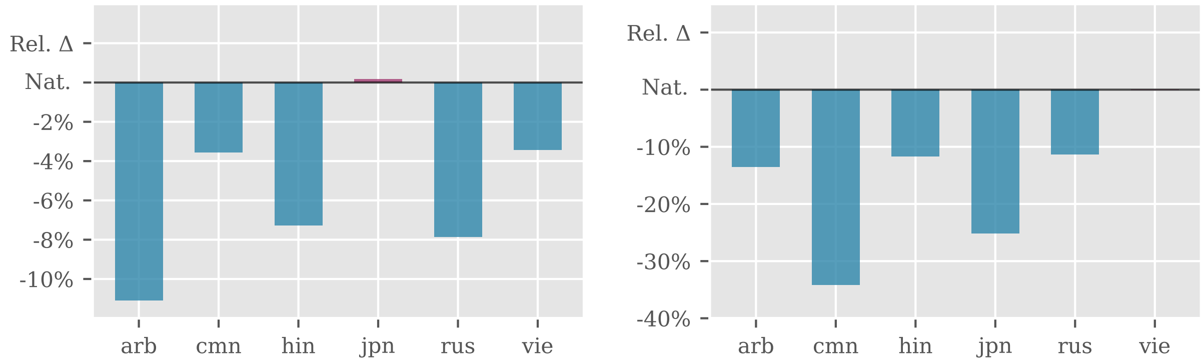
G Detailed Results: Fertility vs. Information Loss



(a)

(b)

Figure 4: Fertility vs. Token Collapse for monolingual models using `UConv`: **(a)** relative change in fertility (lower is better) for MonoUconv compared to MonoNat; **(b)** relative change in the vocabulary size (higher is better) of MonoNat when romanizing each subword in its vocabulary using `UConv` (i.e., subwords get merged due to the same romanization).



(a)

(b)

Figure 5: Fertility vs. token merging for multilingual models: **(a)** relative change in fertility (lower is better) for MonoUroman compared to MonoNat; **(b)** relative change in the vocabulary size (higher is better) of MonoNat when romanizing each subword in its vocabulary (i.e., subwords get merged due to the same romanization).

	Tok.	Vocab	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Seq. Avg.	Token Avg.	Avg.
MonoNat	no-ws	50k	80.3$_{\pm 0.6}$	78.4$_{\pm 0.7}$	81.0$_{\pm 1.6}$	74.9$_{\pm 1.2}$	81.8$_{\pm 0.6}$	82.0$_{\pm 0.7}$	82.2$_{\pm 1.0}$	76.6$_{\pm 1.0}$	79.7$_{\pm 1.0}$
MonoNat	ws	50k	79.8$_{\pm 0.9}$	78.4$_{\pm 1.0}$	81.3$_{\pm 1.0}$	74.6$_{\pm 1.7}$	81.8$_{\pm 0.8}$	82.4$_{\pm 0.5}$	82.0$_{\pm 1.0}$	76.9$_{\pm 1.1}$	79.7$_{\pm 1.1}$
MonoNat	no-ws	25k	80.0$_{\pm 1.2}$	78.1$_{\pm 0.8}$	80.2$_{\pm 0.7}$	75.4$_{\pm 0.8}$	81.5$_{\pm 0.9}$	82.0$_{\pm 0.9}$	81.8$_{\pm 0.9}$	76.8$_{\pm 0.9}$	79.6$_{\pm 0.9}$
MonoNat	ws	25k	79.8$_{\pm 1.3}$	78.1$_{\pm 1.5}$	80.8$_{\pm 0.8}$	74.9$_{\pm 0.7}$	81.7$_{\pm 0.9}$	82.2$_{\pm 0.7}$	81.8$_{\pm 1.1}$	76.8$_{\pm 1.0}$	79.6$_{\pm 1.0}$
MonoNat	no-ws	10k	79.4$_{\pm 0.8}$	79.1$_{\pm 0.9}$	80.8$_{\pm 1.5}$	75.6$_{\pm 1.1}$	80.8$_{\pm 1.2}$	82.0$_{\pm 0.9}$	81.8$_{\pm 1.1}$	76.9$_{\pm 1.1}$	79.6$_{\pm 1.1}$
MonoNat	ws	10k	79.3$_{\pm 1.2}$	79.0$_{\pm 0.9}$	80.3$_{\pm 0.7}$	75.7$_{\pm 0.9}$	81.0$_{\pm 1.6}$	82.1$_{\pm 0.9}$	81.7$_{\pm 1.1}$	76.9$_{\pm 1.1}$	79.6$_{\pm 1.1}$
MonoUroman	no-ws	50k	80.7$_{\pm 0.8}$	74.1$_{\pm 1.0}$	80.7$_{\pm 0.8}$	71.7$_{\pm 1.1}$	81.6$_{\pm 0.6}$	81.9$_{\pm 0.7}$	81.5$_{\pm 0.8}$	74.7$_{\pm 0.8}$	78.4$_{\pm 0.8}$
MonoUroman	ws	50k	80.1$_{\pm 1.1}$	74.0$_{\pm 0.8}$	80.8$_{\pm 0.8}$	72.1$_{\pm 0.8}$	81.7$_{\pm 0.9}$	81.3$_{\pm 0.8}$	81.1$_{\pm 0.9}$	74.9$_{\pm 0.9}$	78.3$_{\pm 0.9}$
MonoUroman	no-ws	25k	79.6$_{\pm 0.6}$	73.6$_{\pm 0.9}$	80.4$_{\pm 1.0}$	71.1$_{\pm 0.6}$	81.5$_{\pm 1.1}$	80.9$_{\pm 0.5}$	80.5$_{\pm 0.8}$	74.6$_{\pm 0.8}$	77.8$_{\pm 0.8}$
MonoUroman	ws	25k	79.5$_{\pm 0.7}$	74.0$_{\pm 1.5}$	80.9$_{\pm 0.9}$	71.4$_{\pm 0.8}$	81.5$_{\pm 0.6}$	81.1$_{\pm 1.1}$	80.7$_{\pm 1.0}$	74.9$_{\pm 1.0}$	78.1$_{\pm 1.0}$
MonoUroman	no-ws	10k	79.7$_{\pm 1.1}$	74.7$_{\pm 1.0}$	80.1$_{\pm 0.8}$	71.8$_{\pm 1.2}$	80.5$_{\pm 1.1}$	80.3$_{\pm 1.2}$	80.6$_{\pm 1.1}$	74.5$_{\pm 1.1}$	77.8$_{\pm 1.1}$
MonoUroman	ws	10k	79.3$_{\pm 1.4}$	73.9$_{\pm 1.1}$	80.8$_{\pm 0.9}$	71.8$_{\pm 0.8}$	80.4$_{\pm 1.1}$	81.0$_{\pm 0.7}$	80.5$_{\pm 1.0}$	74.7$_{\pm 1.0}$	77.9$_{\pm 1.0}$

Table 14: Main results averaged across our 5 evaluation tasks for monolingual and multilingual romanized models and different vocab sizes. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*.

	Tok.	Vocab	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	50k	86.2$_{\pm0.9}$	89.4$_{\pm1.0}$	85.9$_{\pm3.0}$	87.3$_{\pm2.0}$	86.8$_{\pm0.8}$	87.2$_{\pm0.9}$	87.1$_{\pm1.7}$
MonoNat	ws	50k	83.0$_{\pm1.2}$	88.3$_{\pm1.3}$	84.9$_{\pm1.7}$	86.6$_{\pm3.3}$	85.8$_{\pm1.3}$	88.8$_{\pm0.7}$	86.2$_{\pm1.8}$
MonoNat	no-ws	25k	85.0$_{\pm2.6}$	86.2$_{\pm1.5}$	82.3$_{\pm0.9}$	88.0$_{\pm1.4}$	85.6$_{\pm1.6}$	88.7$_{\pm1.7}$	86.0$_{\pm1.7}$
MonoNat	ws	25k	84.7$_{\pm2.5}$	85.4$_{\pm3.1}$	84.1$_{\pm1.4}$	87.9$_{\pm1.2}$	87.1$_{\pm1.3}$	89.8$_{\pm1.3}$	86.5$_{\pm1.9}$
MonoNat	no-ws	10k	84.5$_{\pm1.5}$	88.0$_{\pm1.0}$	87.3$_{\pm3.0}$	86.3$_{\pm1.8}$	84.5$_{\pm2.0}$	89.3$_{\pm1.5}$	86.7$_{\pm1.9}$
MonoNat	ws	10k	83.0$_{\pm2.5}$	88.0$_{\pm1.9}$	85.3$_{\pm0.7}$	87.8$_{\pm1.4}$	85.9$_{\pm3.5}$	89.4$_{\pm1.6}$	86.6$_{\pm2.1}$
MonoUroman	no-ws	50k	88.2$_{\pm1.3}$	84.5$_{\pm1.7}$	84.5$_{\pm1.1}$	84.7$_{\pm2.0}$	85.8$_{\pm0.7}$	89.6$_{\pm1.1}$	86.2$_{\pm1.4}$
MonoUroman	ws	50k	85.3$_{\pm2.2}$	84.7$_{\pm1.4}$	84.6$_{\pm1.2}$	84.7$_{\pm1.3}$	85.6$_{\pm1.5}$	87.9$_{\pm1.1}$	85.5$_{\pm1.5}$
MonoUroman	no-ws	25k	85.0$_{\pm0.7}$	83.5$_{\pm1.7}$	82.7$_{\pm2.0}$	81.8$_{\pm0.4}$	86.6$_{\pm1.9}$	87.2$_{\pm0.7}$	84.5$_{\pm1.4}$
MonoUroman	ws	25k	84.0$_{\pm0.7}$	83.2$_{\pm2.9}$	85.8$_{\pm1.7}$	83.1$_{\pm1.0}$	86.3$_{\pm0.6}$	87.8$_{\pm2.2}$	85.0$_{\pm1.7}$
MonoUroman	no-ws	10k	86.8$_{\pm2.0}$	84.1$_{\pm2.0}$	83.6$_{\pm1.0}$	84.3$_{\pm2.2}$	84.3$_{\pm2.1}$	84.6$_{\pm2.1}$	84.6$_{\pm2.0}$
MonoUroman	ws	10k	83.7$_{\pm2.9}$	84.6$_{\pm1.7}$	85.2$_{\pm1.7}$	83.6$_{\pm1.0}$	83.8$_{\pm2.1}$	87.3$_{\pm0.8}$	84.7$_{\pm1.9}$

Table 15: SIB200 results for monolingual and multilingual romanized models and different vocab sizes. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*.

	Tok.	Vocab	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	50k	89.7$_{\pm0.1}$	75.4$_{\pm0.2}$	90.1$_{\pm0.5}$	65.9$_{\pm0.2}$	87.3$_{\pm0.1}$	90.5$_{\pm0.1}$	83.2$_{\pm0.2}$
MonoNat	ws	50k	89.9$_{\pm0.1}$	75.3$_{\pm0.5}$	91.3$_{\pm0.5}$	65.9$_{\pm0.3}$	87.6$_{\pm0.1}$	91.0$_{\pm0.2}$	83.5$_{\pm0.3}$
MonoNat	no-ws	25k	89.6$_{\pm0.1}$	76.6$_{\pm0.4}$	90.5$_{\pm0.6}$	66.3$_{\pm0.2}$	87.2$_{\pm0.1}$	90.3$_{\pm0.2}$	83.4$_{\pm0.3}$
MonoNat	ws	25k	89.5$_{\pm0.3}$	76.6$_{\pm0.2}$	90.6$_{\pm0.6}$	66.3$_{\pm0.4}$	87.6$_{\pm0.2}$	90.8$_{\pm0.0}$	83.5$_{\pm0.3}$
MonoNat	no-ws	10k	89.0$_{\pm0.2}$	77.6$_{\pm0.4}$	90.2$_{\pm0.5}$	67.7$_{\pm0.2}$	87.1$_{\pm0.2}$	90.3$_{\pm0.2}$	83.6$_{\pm0.3}$
MonoNat	ws	10k	89.5$_{\pm0.2}$	77.6$_{\pm0.4}$	89.9$_{\pm0.7}$	66.9$_{\pm0.3}$	86.9$_{\pm0.1}$	90.7$_{\pm0.2}$	83.6$_{\pm0.4}$
MonoUroman	no-ws	50k	89.6$_{\pm0.2}$	68.1$_{\pm0.4}$	89.5$_{\pm0.7}$	59.7$_{\pm0.1}$	87.1$_{\pm0.3}$	89.5$_{\pm0.3}$	80.5$_{\pm0.4}$
MonoUroman	ws	50k	89.7$_{\pm0.2}$	67.9$_{\pm0.2}$	90.4$_{\pm0.4}$	59.3$_{\pm0.3}$	87.4$_{\pm0.1}$	90.2$_{\pm0.1}$	80.8$_{\pm0.2}$
MonoUroman	no-ws	25k	89.2$_{\pm0.1}$	67.9$_{\pm0.4}$	90.7$_{\pm0.5}$	59.2$_{\pm0.4}$	87.0$_{\pm0.1}$	89.4$_{\pm0.1}$	80.5$_{\pm0.3}$
MonoUroman	ws	25k	89.6$_{\pm0.2}$	68.9$_{\pm0.3}$	90.2$_{\pm0.5}$	59.6$_{\pm0.2}$	87.1$_{\pm0.2}$	90.1$_{\pm0.3}$	80.9$_{\pm0.3}$
MonoUroman	no-ws	10k	88.9$_{\pm0.2}$	69.2$_{\pm0.1}$	89.9$_{\pm0.7}$	60.0$_{\pm0.6}$	86.6$_{\pm0.2}$	89.5$_{\pm0.3}$	80.7$_{\pm0.4}$
MonoUroman	ws	10k	89.1$_{\pm0.3}$	68.7$_{\pm0.3}$	90.7$_{\pm0.5}$	60.5$_{\pm0.5}$	87.0$_{\pm0.2}$	90.2$_{\pm0.1}$	81.0$_{\pm0.3}$

Table 16: Wikiann results for monolingual and multilingual romanized models and different vocab sizes. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*.

	Tok.	Vocab	arb_Arab	cmn_Hani	hin_Deva	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	50k	74.6$_{\pm0.6}$	74.8$_{\pm0.7}$	72.0$_{\pm1.5}$	75.7$_{\pm1.0}$	76.7$_{\pm1.1}$	74.8$_{\pm1.1}$
MonoNat	ws	50k	74.2$_{\pm1.3}$	75.3$_{\pm1.0}$	72.8$_{\pm1.0}$	75.6$_{\pm0.9}$	76.4$_{\pm0.7}$	74.9$_{\pm1.0}$
MonoNat	no-ws	25k	74.1$_{\pm0.5}$	74.6$_{\pm0.9}$	71.6$_{\pm0.9}$	75.9$_{\pm0.4}$	76.2$_{\pm0.7}$	74.5$_{\pm0.7}$
MonoNat	ws	25k	74.0$_{\pm0.4}$	74.7$_{\pm1.0}$	71.7$_{\pm0.8}$	74.6$_{\pm1.3}$	75.4$_{\pm0.6}$	74.1$_{\pm0.9}$
MonoNat	no-ws	10k	73.8$_{\pm0.7}$	74.0$_{\pm1.4}$	70.9$_{\pm0.8}$	74.0$_{\pm1.3}$	75.9$_{\pm1.1}$	73.7$_{\pm1.1}$
MonoNat	ws	10k	73.8$_{\pm0.7}$	73.9$_{\pm0.5}$	71.1$_{\pm1.0}$	73.9$_{\pm0.9}$	75.0$_{\pm0.6}$	73.5$_{\pm0.8}$
MonoUroman	no-ws	50k	74.2$_{\pm0.7}$	71.2$_{\pm1.2}$	73.3$_{\pm0.6}$	75.9$_{\pm0.8}$	76.3$_{\pm0.9}$	74.2$_{\pm0.9}$
MonoUroman	ws	50k	74.9$_{\pm0.9}$	70.7$_{\pm0.8}$	72.9$_{\pm0.7}$	75.4$_{\pm1.1}$	74.5$_{\pm0.7}$	73.7$_{\pm0.9}$
MonoUroman	no-ws	25k	73.7$_{\pm0.9}$	70.4$_{\pm0.9}$	72.1$_{\pm0.8}$	75.1$_{\pm0.9}$	75.6$_{\pm0.5}$	73.4$_{\pm0.8}$
MonoUroman	ws	25k	73.8$_{\pm1.0}$	70.7$_{\pm1.3}$	72.3$_{\pm0.6}$	74.5$_{\pm0.8}$	74.2$_{\pm0.9}$	73.1$_{\pm0.9}$
MonoUroman	no-ws	10k	73.2$_{\pm1.1}$	71.0$_{\pm0.5}$	72.0$_{\pm1.0}$	74.5$_{\pm0.8}$	75.6$_{\pm0.9}$	73.3$_{\pm0.9}$
MonoUroman	ws	10k	73.8$_{\pm0.6}$	70.8$_{\pm1.0}$	71.8$_{\pm0.7}$	73.4$_{\pm1.1}$	73.9$_{\pm0.6}$	72.8$_{\pm0.8}$

Table 17: XNLI results for monolingual and multilingual romanized models and different vocab sizes. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*.

	Tok.	Vocab	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	50k	79.8$_{\pm0.5}$	83.6$_{\pm0.5}$	85.1$_{\pm0.5}$	83.2$_{\pm0.5}$	84.8$_{\pm0.5}$	85.0$_{\pm0.2}$	83.6$_{\pm0.5}$
MonoNat	ws	50k	79.9$_{\pm0.5}$	83.3$_{\pm1.3}$	85.1$_{\pm0.5}$	83.3$_{\pm0.5}$	85.3$_{\pm0.6}$	85.3$_{\pm0.3}$	83.7$_{\pm0.7}$
MonoNat	no-ws	25k	79.6$_{\pm0.5}$	83.6$_{\pm0.3}$	85.3$_{\pm0.4}$	83.3$_{\pm0.2}$	84.8$_{\pm1.1}$	85.1$_{\pm0.2}$	83.6$_{\pm0.5}$
MonoNat	ws	25k	79.2$_{\pm1.4}$	84.2$_{\pm0.4}$	84.9$_{\pm0.3}$	82.7$_{\pm0.4}$	85.3$_{\pm0.7}$	85.2$_{\pm0.6}$	83.6$_{\pm0.7}$
MonoNat	no-ws	10k	79.1$_{\pm0.5}$	84.8$_{\pm0.4}$	84.1$_{\pm0.5}$	83.6$_{\pm0.5}$	84.9$_{\pm0.8}$	84.8$_{\pm0.1}$	83.5$_{\pm0.5}$
MonoNat	ws	10k	79.1$_{\pm0.7}$	84.7$_{\pm0.1}$	83.7$_{\pm0.5}$	83.1$_{\pm0.6}$	85.2$_{\pm0.3}$	85.4$_{\pm0.3}$	83.5$_{\pm0.5}$
MonoUroman	no-ws	50k	79.4$_{\pm0.6}$	81.4$_{\pm0.4}$	84.5$_{\pm0.9}$	82.1$_{\pm0.3}$	85.3$_{\pm0.6}$	84.2$_{\pm0.6}$	82.8$_{\pm0.6}$
MonoUroman	ws	50k	79.3$_{\pm0.3}$	81.5$_{\pm0.6}$	84.6$_{\pm0.6}$	82.4$_{\pm0.4}$	85.5$_{\pm0.6}$	84.4$_{\pm0.8}$	83.0$_{\pm0.6}$
MonoUroman	no-ws	25k	79.0$_{\pm0.7}$	81.1$_{\pm0.6}$	84.6$_{\pm0.3}$	82.0$_{\pm0.4}$	84.7$_{\pm0.8}$	84.0$_{\pm0.5}$	82.6$_{\pm0.6}$
MonoUroman	ws	25k	79.0$_{\pm0.4}$	81.4$_{\pm0.8}$	84.3$_{\pm0.2}$	81.9$_{\pm0.3}$	84.9$_{\pm0.8}$	84.4$_{\pm0.4}$	82.7$_{\pm0.5}$
MonoUroman	no-ws	10k	78.8$_{\pm0.6}$	82.5$_{\pm0.6}$	84.0$_{\pm0.8}$	82.5$_{\pm0.2}$	84.2$_{\pm0.9}$	83.6$_{\pm1.4}$	82.6$_{\pm0.8}$
MonoUroman	ws	10k	78.9$_{\pm0.5}$	81.2$_{\pm0.7}$	84.8$_{\pm0.6}$	82.3$_{\pm0.3}$	84.3$_{\pm0.6}$	84.6$_{\pm0.5}$	82.7$_{\pm0.5}$

Table 18: MASSIVE Intent Classification results for monolingual and multilingual romanized models and different vocab sizes. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*.

	Tok.	Vocab	arb_Arab	cmn_Hani	hin_Deva	jpn_Jpan	rus_Cyrl	vie_Latn	Avg.
MonoNat	no-ws	50k	71.3$_{\pm0.6}$	68.6$_{\pm0.9}$	71.9$_{\pm0.4}$	63.3$_{\pm1.4}$	74.6$_{\pm0.1}$	70.5$_{\pm0.6}$	70.0$_{\pm0.8}$
MonoNat	ws	50k	72.1$_{\pm0.6}$	69.5$_{\pm0.6}$	72.4$_{\pm0.6}$	62.6$_{\pm0.8}$	74.7$_{\pm0.3}$	70.5$_{\pm0.2}$	70.3$_{\pm0.6}$
MonoNat	no-ws	25k	71.9$_{\pm0.6}$	69.6$_{\pm0.5}$	71.4$_{\pm0.3}$	64.0$_{\pm0.4}$	74.3$_{\pm0.6}$	69.8$_{\pm0.5}$	70.2$_{\pm0.5}$
MonoNat	ws	25k	71.6$_{\pm0.2}$	69.5$_{\pm0.5}$	72.5$_{\pm0.4}$	62.8$_{\pm0.3}$	74.2$_{\pm0.5}$	69.9$_{\pm0.4}$	70.1$_{\pm0.4}$
MonoNat	no-ws	10k	70.6$_{\pm0.5}$	71.1$_{\pm1.0}$	71.4$_{\pm0.9}$	65.0$_{\pm1.0}$	73.6$_{\pm0.8}$	69.8$_{\pm0.4}$	70.2$_{\pm0.8}$
MonoNat	ws	10k	70.9$_{\pm0.7}$	70.8$_{\pm0.8}$	71.7$_{\pm0.5}$	65.0$_{\pm0.8}$	73.4$_{\pm0.8}$	70.0$_{\pm1.0}$	70.3$_{\pm0.8}$
MonoUroman	no-ws	50k	72.2$_{\pm0.8}$	65.2$_{\pm0.6}$	71.6$_{\pm0.5}$	60.4$_{\pm0.8}$	74.2$_{\pm0.4}$	69.7$_{\pm0.6}$	68.9$_{\pm0.6}$
MonoUroman	ws	50k	71.5$_{\pm0.5}$	65.2$_{\pm0.5}$	71.5$_{\pm0.6}$	62.0$_{\pm0.8}$	74.4$_{\pm0.6}$	69.6$_{\pm0.9}$	69.0$_{\pm0.7}$
MonoUroman	no-ws	25k	71.0$_{\pm0.3}$	65.2$_{\pm0.4}$	71.6$_{\pm0.6}$	61.2$_{\pm1.0}$	74.2$_{\pm0.5}$	68.5$_{\pm0.6}$	68.6$_{\pm0.6}$
MonoUroman	ws	25k	71.1$_{\pm0.9}$	65.6$_{\pm0.9}$	71.8$_{\pm0.6}$	60.9$_{\pm1.2}$	74.6$_{\pm0.2}$	69.1$_{\pm0.7}$	68.9$_{\pm0.8}$
MonoUroman	no-ws	10k	71.0$_{\pm0.8}$	66.5$_{\pm0.8}$	70.8$_{\pm0.4}$	60.3$_{\pm0.7}$	72.9$_{\pm0.6}$	68.4$_{\pm0.8}$	68.3$_{\pm0.7}$
MonoUroman	ws	10k	71.0$_{\pm0.4}$	64.0$_{\pm1.0}$	71.4$_{\pm0.4}$	61.0$_{\pm1.0}$	73.5$_{\pm0.6}$	69.3$_{\pm1.0}$	68.4$_{\pm0.8}$

Table 19: MASSIVE Slot Filling results for monolingual and multilingual romanized models and different vocab sizes. We ablate different settings for the tokenizer training flag *Split by Whitespace*: `ws` refers to *Split by Whitespace* set to *true*, `no-ws` refers to *Split by Whitespace* set to *false*.