Happiness is Sharing a Vocabulary: A Study of Transliteration Methods

Anonymous ACL submission

Abstract

Transliteration has emerged as a promising means to bridge the gap between various languages in multilingual NLP, showing promising results especially for languages using non-Latin scripts. We investigate the degree to which shared script, overlapping token vocabularies, and shared phonology contribute to performance of multilingual models. To this end, we conduct controlled experiments using three kinds of transliteration (romanization, phonemic transcription, and substitution ciphers) as well as orthography. We evaluate each model on two downstream tasks—named entity recognition (NER) and natural language inference (NLI)—and find that romanization significantly outperforms other input types in 7 out of 8 evaluation settings, largely consistent with our hypothesis that it is the most effective approach. We further analyze how each factor contributed to the success, and suggest that having longer (subword) tokens shared with pre-trained languages leads to better utilization of the model.

1 Introduction

Multilingual language modeling has drawn significant attention from researchers seeking to support diverse languages and promote fairness in AI. One crucial problem in improving multilingual language modeling is the *script barrier*, a phenomenon in which models struggle to share knowledge between languages written in different scripts, due to mismatched input representations.

Transliteration, which maps a given string to another sequence of characters (e.g., Cyrillic script → Latin script) has been explored as a potential solution to the problem of script barriers. In multilingual NLP, transliteration typically maps text to the Latin script or the International Phonetic Alphabet (IPA), giving various languages a shared input representation. Both representations encode linguistic information—specifically phonetic and phonological information—across languages. Here,

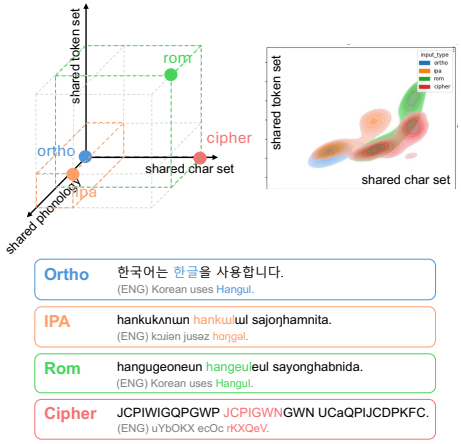


Figure 1: *Top left:* Conceptual visualization of the transliteration analysis schema, positioning input types (Ortho, IPA, Rom, Cipher) based on shared character set, token set, and phonology. *Top right:* KDE plot showing empirical distribution of overlap ratios for each quantifiable component. *Bottom:* Transliteration examples generated with each method.

we pose a question: *Is it the shared script itself or the linguistic information encoded in the scripts that helps the models adapt to other languages?*

To investigate this question, we define three key factors in transliteration—(i) shared character set, (ii) shared token set, and (iii) shared phonology of synonymous lexical items—that influence how a model processes and generalizes across languages. We then designate four input types varying in the degree to which these factors are present: Orthography (Ortho), IPA, Romanized (Rom), and Substitution Ciphered text (Cipher). In the context of a script barrier—where two languages are written in different scripts—each input type can be characterized in terms of the three factors as illustrated in Figure 1 and detailed in Section 3.

We then conduct controlled experiments to understand how each factor of transliteration contributes to success. First, we pre-train Transformer-based multilingual language models from scratch

with each of these four input types, and then fine-tune them on target languages—including both seen and unseen languages—to examine how well each input type adapts to new languages. The languages used for pre-training are selected based on our defined language similarity scores as detailed in Section 4.1. We constructed four language sets to account for various scenarios in which language similarity or script familiarity may vary.

We hypothesize that Rom yields the best performance in handling diverse languages with diverse scripts, as it improves input representations across all three dimensions. Based on this assumption, IPA is expected to follow, as it enhances two out of three dimensions (shared phonology and tokens) while ciphered text only shares the character set and lacks other shared representations. Throughout the paper, we evaluate our hypothesis by comparing downstream task performance specifically on unseen languages and analyzing each method in terms of the defined factors, with an emphasis on shared tokens. We summarize our contribution as follows:

- We define three factors—shared character set, shared token set, and shared phonology—to explain the effect of transliteration on multilingual inputs.
- We conduct controlled pre-training across four language sets and input types, evaluating two downstream tasks to assess how each factor contributes to the success of transliteration.
- We analyze vocabulary overlap by token length, revealing how different transliteration methods yield distinct overlap patterns that contribute to cross-lingual adaptation.

2 Related Work

2.1 Script Barrier and Transliteration

Script barrier refers to the phenomenon in which multilingual language models struggle to share knowledge between languages written in different scripts, due to mismatched input representations. Even massively multilingual models face persistent challenges in handling languages whose scripts were unseen or severely underrepresented during pre-training (de Vries et al., 2022; Fujinuma et al., 2022; Muller et al., 2021; Pfeiffer et al., 2021).

To overcome this limitation, *transliteration* has been explored as an efficient means to improve cross-lingual transfer. Among transliteration methods, **romanization**—the conversion of non-Latin scripts into the Latin alphabet—has gained popularity due to the dominance of Latin scripts in pre-trained language models (Muller et al., 2021; Purkayastha et al., 2023; Husain et al., 2024). Another popular method is **grapheme-to-phoneme** (G2P) conversion, which converts text into phonemes represented in International Phonetic Alphabet (IPA). G2P has been applied either by replacing input representations (Bharadwaj et al., 2016; Sohn et al., 2024) or by integrating phonemic information as an auxiliary signal (Nguyen et al., 2023, 2025).

While these approaches demonstrate transliteration's practical benefits, relatively little is known about *why* it works. Moosa et al. (2023) examined how transliteration benefits multilingual models by analyzing representational similarity and tokenizer fertility. However, their study mainly focuses on closely related Indic languages, leaving open questions about how transliteration contributes to cross-lingual transfer in a broader, typologically diverse setting. Similarly, Jung et al. (2024) investigated phonemic representations in terms of representational similarity across languages, but without clarifying how transliterated inputs lead to improved cross-lingual performance.

Our work advances these studies by systematically isolating and analyzing the underlying factors—shared character set, shared token set, and shared phonology—that explain when and why transliteration is effective.

2.2 Vocabulary Overlap in Cross-lingual Transfer

Lexical overlap has been explored in various works in multilingual NLP. Prior studies have shown that shared lexical or subword units across languages allow models to reuse learned representations, thereby improving transferability (Pires et al., 2019; Chang et al., 2024; Limisiewicz et al., 2023). In contrast, a high proportion of unknown tokens—tokens that the model fails to segment due to their absence from the pre-training corpus—has been shown to hinder transfer and degrade downstream performance (Pfeiffer et al., 2021).

However, Philippy et al. (2023) put former literature together, acknowledging conflicting findings regarding how lexical overlap benefits cross-lingual transfer. They emphasized the need for task-specific analyses and a more comprehensive understanding of this factor.

In this work, we examine vocabulary overlap to

	Shared Char. Set	Shared Token Set	Shared Phonology
Ortho	–	–	–
IPA	±	±	+
Rom	+	+	±
Cipher	+	–	–

Table 1: Input types characterized by the three factors for transliteration.

understand its contribution to the success of transliteration. By analyzing overlap by length, we provide deeper insights into how vocabulary overlap shapes multilingual adaptation.

3 Input Types

Input types are selected to vary according to three different dimensions:

- **Shared Character Set.** Transliteration usually enforces a shared character set across languages. For example, applying romanization to English (e.g., "hello" → "hello") and Korean (e.g., "안녕하세요" → "annyeonghaseyo") produces only Latin characters, which significantly reduces the number of unique characters and patterns that a tokenizer must capture.
- **Shared Token Set.** Transliteration also yield shared subword tokens across languages. Here, we specifically distinguish *tokens* from *characters*. By tokens we refer to subword tokens longer than one (more than one character). With English and Korean example again, "Canada" → "canada" and "캐나다" → "kaenada" produce "ada" as their shared token. Since a sequence of characters is more likely to contain semantic meaning than a single character, this distinction is designed to decompose the effect of sharing the surface form and sharing their associated meanings.
- **Shared Phonology.** Widely used transliteration methods (e.g., G2P and romanization) encode phonological information in their representations. The main motivation for this is that they are more likely to have similar representations for lexical items that sound similar. We consider the extent to which they capture the phonology of each language as one of the main factors behind the effectiveness of transliteration, since it enables the recognition of cognates and borrowed vocabulary shared across languages. For example, G2P conversion captures similarities in pronunciation between borrowed words: English "smartphone" → "smɑɹtfown" and Korean "스마트폰" → "sɯmat^{h}ɯp^{h}on".

To explore these different dimensions of transliteration, we employ four distinct input types: Orthography (Ortho), IPA, Romanized text (Rom), and Substitution Ciphered text (Cipher). How each input type is associated with each factors is summarized in Table 1. The following subsections provide a detailed explanation of each input type and the process of converting written text data (Ortho) into the other input types.

3.1 IPA: G2P Conversion

Based on Latin scripts, IPA symbols are designed to represent pronunciations of human language as phonemes. It is therefore considered to be the most accurate method at representing shared phonology. While transliteration into IPA enables some degree of character set sharing, differences in phonemic inventories and phonotactic structures across languages cause each language to use its own distinct set of characters and subword tokens. For example, difference in their phonemic inventories cause 'hotel' to be converted into /otel/ in Spanish and /howtɛl/ in English.

To convert orthographic data into IPA symbols, we use Epitran (Mortensen et al., 2018). It is a rule-based G2P tool that supports more than a hundred languages, and is widely adopted in transliteration-based methods (Bharadwaj et al., 2016; Chaudhary et al., 2018; Leong and Whitenack, 2022; Sohn et al., 2024; Zhu et al., 2024). We chose a rule-based conversion framework to ensure both consistency and practicality, as such rules are well-established for many languages and the conversion process is much faster.

3.2 Rom: Romanization

Romanization is the process of converting various scripts into Latin letters, enforcing a stricter limit in character sets used across languages. Additionally, unlike G2P, which often adds language-specificity to languages originally using Latin scripts, romanization preserves the written form of Latin script. This allows the resulting text to retain the advantages of a shared script, increasing the likelihood of sharing more subword tokens. Since Latin scripts encode sound—though not as precisely as IPA—

romanization produces phonologically informed representations for each language.

For romanization, we employ Uroman (Hermjakob et al., 2018) which supports conversion of any UTF-8-encoded script into Latin script. The tool has been widely used in prior work on romanization and shown to be robust (Amrhein and Sennrich, 2020; Liu et al., 2024; Purkayastha et al., 2023).

3.3 Cipher: Substitution Cipher

A substitution cipher is a method from cryptography where units of plaintext are replaced with ciphertext according to a predefined rule or key. We apply substitution cipher to the romanized text of each language—with different rules for each—to remove encoded phonological information. While this allows multilingual text to share the same character space as Rom, it no longer encodes cross-lingual phonological patterns and prevents the sharing of meaningful subword tokens across languages.

We employ a Caesar cipher, a simple substitution encryption technique that shifts each letter in the text by a fixed number of positions in the Latin alphabet. For each language, we assign an integer that determines the shift from the current position of each letter. For example, if English is assigned the integer 4, the word 'apple' would be represented as 'ettpi', with each letter replaced by the one that is four positions ahead in the alphabet. Details are provided in Appendix A.4.

4 Experimental Setup

4.1 Language Selection

To examine how different input types impact multilingual adaptation, we selected languages to form four language sets: (i) typologically similar languages using the same script (sim-same), (ii) similar languages using diverse scripts (sim-div), (iii) dissimilar languages using the same script (dissim-same), and (iv) dissimilar languages using diverse scripts (dissim-div). Similar to Chang et al. (2024), we utilized lang2vec (Littell et al., 2017)[1] to compute language similarity. We extracted syntactic, geographic, and genetic feature vectors from lang2vec to obtain cosine similarities, and also defined lexical similarity based on word overlap ratio between training corpora of each language[2]. We assigned eight languages to each set (see Table 2) by sampling languages based on the aggregated scores. More details are provided in Appendix A.1.

Similarity	Script Diversity	Language	Language Family	Script
similar	same	swe	Indo-European	Latn
		por	Indo-European	Latn
		lij	Indo-European	Latn
		cat	Indo-European	Latn
		ron	Indo-European	Latn
		spa	Indo-European	Latn
		sqi	Indo-European	Latn
		fra	Indo-European	Latn
similar	div	fra	Indo-European	Latn
		ben	Indo-European	Cyrl
		hin	Indo-European	Beng
		hrv	Indo-European	Deva
		ori	Indo-European	Odia
		rus	Indo-European	Cyrl
		srp	Indo-European	Cyrl
		urd	Indo-European	Arab
dissimilar	same	ilo	Austronesian	Latn
		sna	Niger-Congo	Latn
		lav	Indo-European	Latn
		uzb	Turkic	Latn
		deu	Indo-European	Latn
		fin	Uralic	Latn
		som	Afroasiatic	Latn
		swa	Niger-Congo	Latn
dissimilar	div	amh	Afroasiatic	Ethi
		ben	Indo-European	Beng
		tel	Dravidian	Telu
		fra	Indo-European	Latn
		tha	Tai-Kaidai	Thai
		kat	Caucasian	Geor
		kor	Koreanic	Hang
		mya	Sino-Tibetan	Mymr

Table 2: Languages selected for each language set.

4.2 Vocabulary Overlap Ratio

As discussed in Section 3, different transliteration methods are likely to produce different patterns of token overlap. We therefore focus on analyzing overlaps between target language and pretraining languages to understand how each transliteration method benefits. Throughout the analysis, we compute the overlap ratio as follows:

$$\text{OverlapRatio}(l_t, L_s) = \max_{l \in L_s} \frac{|S_l \cap S_{l_t}|}{|S_{l_t}|} \quad (1)$$

where L_s is a set of pre-trained languages, l_t the target language, and S_l the set of tokens from the language l. We suppose all sets S that are used to compute overlap ratio is based on the same tokenizer. We take overlap ratio with that of the

[1] Utilizing https://github.com/antonisa/lang2vec

[2] Words are segmented by white spaces.

We further break down the overlaps by length to examine which lengths contribute most to the results through shared tokens. The overlap ratio for tokens of length m is computed as

$$\frac{|\{x \in S_{l_s} \cap S_{l_t} \mid \text{len}(x) = m\}|}{|S_{l_t}|} \quad \text{where} \quad l_s = \arg\max_{l \in L_s} \frac{|S_l \cap S_{l_t}|}{|S_{l_t}|}. \tag{2}$$

We compute the overlap ratio using the pre-trained language that exhibits the greatest overlap. Counting token overlaps across all pre-trained languages would, in most cases, cover nearly all tokens from the target language, since the tokenizer is trained on those languages. For a more meaningful analysis, we therefore focus on the single pre-trained language that the target language is most likely to leverage for our main analysis. Analyses based on overlaps with all pre-trained languages are provided in the Appendix A.6.

4.3 Datasets

For pre-training, we utilize sampled version of a preprocessed Wikipedia corpus from Hugging Face.[3] For downstream tasks, we utilized WikiAnn (Pan et al., 2017; Rahimi et al., 2019) dataset for NER and XNLI (Conneau et al., 2018) for sentence classification (NLI) task. More details on preprocessing and dataset statistics can be found in Appendix A.10. In order to train the model with different input types, we converted all datasets into each input type using the aforementioned tools.

4.4 Model Training

To investigate the impact of different input types, we pre-train 16 models from scratch using four input types and four language sets. We avoid using publicly available pre-trained models to ensure a controlled experimental setup, as most such models are trained on orthographic data, which would confound the analysis of transliteration effects.

For our controlled experiments, we first trained a SentencePiece tokenizer (BPE) for each model with fixed vocabulary size of 30K for all tokenizers. For model architecture, we employed Transformer encoder architecture as in XLM-R (Conneau et al., 2020)—a widely known model for its compelling multilingual performances in various tasks. As with XLM-R, we follow the training regime used for RoBERTa (Liu et al., 2019) with masked language modeling on each multilingual corpus. After pre-training, we fine-tune each model on target language dataset to evaluate its downstream performance. For details on the model configurations and training, refer to Appendix A.2 and Appendix A.3.

[3]https://huggingface.co/datasets/wikimedia/wikipedia

5 Results

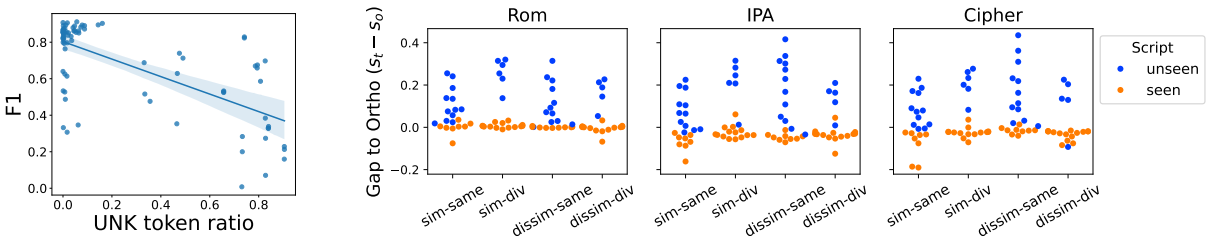
Trained Lang. Set	Input Type	NER			NLI		
		All	Seen	Unseen	All	Seen	Unseen
sim-same	# Lang.	28	8	20	11	2	9
	Ortho	0.7141	**0.8466**	0.6611	0.5847	**0.6768**	0.5642
	IPA	0.7168	0.8085	0.6801	0.6057	0.6605	0.5936
	Rom	**0.7589**	0.8395	**0.7267**	**0.6276**	0.6755	**0.6170**
	Cipher	0.7210	0.8173	0.6824	0.6133	0.6658	0.6017
sim-div	# Lang.	28	8	20	11	4	7
	Ortho	0.6917	0.8409	0.6321	0.6051	0.6242	0.5942
	IPA	0.7202	0.8239	0.6787	0.6171	0.6192	0.6160
	Rom	**0.7522**	**0.8451**	**0.7151**	**0.6248**	**0.6273**	**0.6233**
	Cipher	0.7200	0.8270	0.6772	0.6128	0.6104	0.6142
dissim-same	# Lang.	28	7	21	11	2	9
	Ortho	0.6935	0.7860	0.6626	0.6000	0.6243	0.5947
	IPA	0.7534	0.7732	0.7468	0.6093	**0.6307**	0.6045
	Rom	0.7455	**0.7981**	0.7280	**0.6155**	0.6195	**0.6146**
	Cipher	**0.7592**	0.7725	**0.7547**	0.5995	0.6296	0.5928
dissim-div	# Lang.	28	8	20	11	2	9
	Ortho	0.7436	0.7402	0.7450	0.5670	**0.6481**	0.5490
	IPA	0.7524	0.7524	0.7524	0.6220	0.6039	0.6260
	Rom	**0.7748**	**0.7538**	**0.7832**	**0.6327**	0.6172	**0.6361**
	Cipher	0.7502	0.7518	0.7496	0.6268	0.6176	0.6288

Table 3: Downstream task performances for each model—F1 scores for NER and accuracy scores for NLI. Average scores for all, seen, and unseen languages are reported for each task. **Bold** indicates best performing input type per language set.

Table 3 presents average scores across target languages for the two downstream tasks—F1 scores for NER and average accuracies on XNLI—for seen and unseen languages[4]. P-values obtained from paired t-tests on F1 scores across input types can be found in Appendix A.5.

NER Results We observe that Rom outperforms other input types on average for unseen languages, which simulate the script barrier. Even for seen languages, Rom performs comparably to Ortho. To further verify the significance of the results, we conduct paired t-tests to examine the statistical significance of differences between input types. We find that Rom yields a significant advantage over all other input types for unseen languages ($p < 0.05$), whereas Ortho performs worse than all other input types ($p < 0.05$). Interestingly, IPA and Cipher do not differ significantly, despite Cipher encoding no shared linguistic information across languages.

[4]Unseen languages refer to languages not included in pre-training of each model.



(a) (b)

Figure 2: (a) Negative correlation between UNK token ratio and F1 score. (b) Performance gains of transliterated input types compared to orthography-based models, where s_t and s_o denote scores with transliterated and orthographic inputs, respectively. Performance gains appear primarily in target languages whose original scripts are unseen during pre-training.

XNLI Results For XNLI, we evaluated on 11 languages, most of which were unseen during pre-training. Similar to NER, transliteration does not improve performance on seen languages, with no statistically significant differences between input types. However, for unseen languages, all transliterated input types outperform Ortho, with Rom consistently showing the strongest performance across all language sets. Paired t-tests examining the differences between input types support these findings (See Appendix A.5).

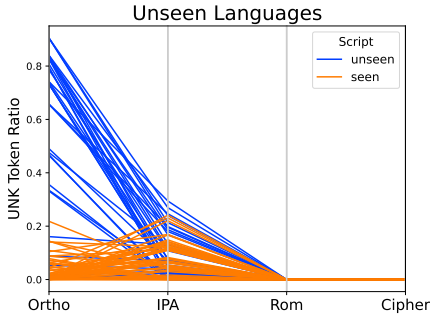


Figure 3: Unknown (UNK) token ratio for unseen languages across different input types.

6 Analysis and Discussion

In this section, we analyze how each factor of transliteration—shared characters, shared (subword) tokens, and shared phonology—is associated with the downstream performance. Given that transliteration benefits unseen languages rather than seen ones, our analysis focuses on unseen languages unless otherwise noted.

6.1 Shared Character Set: Overcoming UNK Tokens

Figure 2a shows that the script barrier is closely related to the proportion of unknown (UNK) tokens, which we compute as the number of UNK tokens divided by the total number of tokens in the evaluation data. A UNK token occurs when the tokenizer fails to segment a sequence of characters using its learned vocabulary, and high UNK ratios arise primarily in languages written in unseen scripts, where the tokenizer has no prior exposure to the characters.

Consequently, applying transliteration and sharing characters across languages greatly reduces UNK ratio in unseen languages (See Figure 3), leading to clear performance gains over Ortho as shown in Figure 2b. To this end, **shared character set** serves as the initial means by which transliteration overcomes the script barrier, reducing the proportion of UNK tokens. Results from Cipher further support this interpretation: despite containing no semantic or linguistic information shared across languages, its simple character sharing yields notable performance gains for languages written in unseen scripts compared to Ortho.

6.2 Sharing Longer Token Matters

While reducing the UNK ratio substantially mitigates the script barrier for languages written in unseen scripts, the correlation between UNK ratio and downstream performance becomes less pronounced across different transliteration methods—IPA, Rom, and Cipher. To further investigate what drives these remaining differences, we analyze token overlaps across languages under each transliteration method, examining how lengths of overlapping tokens relate to downstream performance.

Effect of Longer Shared Tokens Figure 4 shows the Pearson correlation between token-overlap ratios of different lengths and downstream performance. The overlap ratio by length is computed

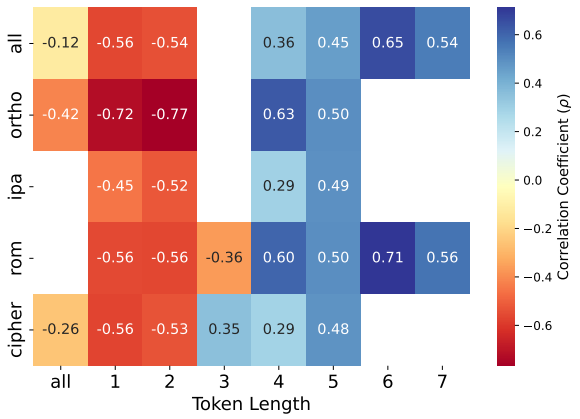


Figure 4: Correlation between token overlap ratio by length and downstream performance. Correlations with $p > 0.05$ are masked.

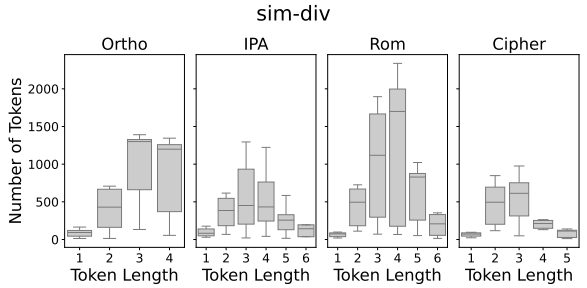


Figure 5: Number of unique tokens by length for unseen target languages, using models trained on sim-div languages.

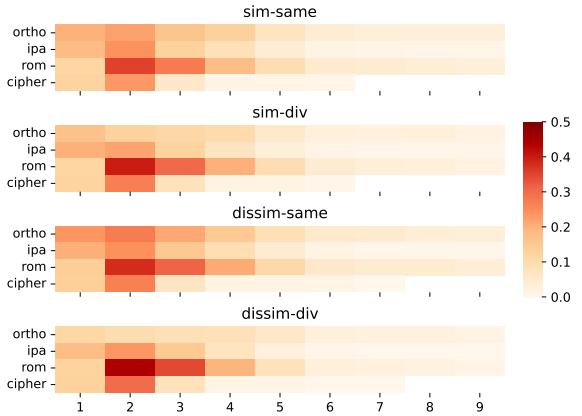


Figure 6: Average vocabulary coverage of each model on unseen languages. The x-axis indicates token length. Romanization shows higher coverage for tokens of length 2-4, whereas other input types exhibit more limited utilization of their vocabulary space.

as described in Equation (2). As shown in Figure 4, overlaps of relatively shorter tokens (including a single character) correlate negatively with performance, whereas overlaps of longer tokens show positive correlations. Notably, negative correlation between character-level overlaps and the performances indicates that while sharing characters alleviates the script barrier by reducing UNK tokens, excessive character-level overlap can be detrimental to downstream performance. One possible explanation is that shorter tokens, which tend to vary more in meaning across contexts, may confuse the model when shared across languages, whereas longer tokens are more likely to provide more stable and consistent semantic cues across languages.

Romanization and Token Length We next examine how this observation relates to the notable success of romanization. Our hypothesis is that romanization produces longer tokens, which in turn positively affect performance due to their better consistency across languages. To test this hypothesis, we examine the distribution of unique token lengths produced by each model and find that Rom generates the largest proportion of longer tokens. Figure 5 shows the distribution of token lengths for models trained on sim-div languages, with results for other language sets provided in the Appendix A.7. We attribute this to the shared phonology introduced by transliteration, which we discuss further in Section 6.3.

Vocabulary Coverage To better understand the impact of longer shared tokens, we additionally analyze what we define as *vocabulary coverage* and how each token length contribute to it. We define vocabulary coverage as the ratio of unique tokens produced to the total vocabulary size of the tokenizer. A higher coverage indicates that a larger portion of the model's embedding space is being used, allowing more effective utilization of its capacity through a greater share of token embeddings.

Figure 6 shows the average vocabulary coverage across unseen languages. It clearly illustrates that Rom achieves higher coverage than other input types, especially for tokens longer than two characters. Figure 7 further presents how the contribution of each token length to overall vocabulary coverage varies across input types. Rom clearly produces a greater variety of longer tokens than any other input type. This broader token usage ensures greater vocabulary coverage and thereby better leverages the model's capacity.

Longer Tokens and Vocabulary Coverage Building on the observation of how tokens of different lengths contribute to vocabulary coverage,

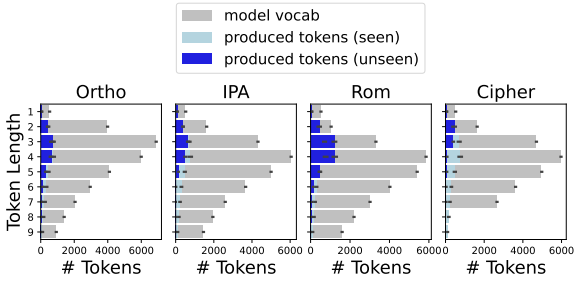


Figure 7: Number of tokens produced by each input type for seen and unseen languages. Rom utilizes a larger proportion of the model's vocabulary, whereas other input types show more limited token usage.

we further interpret the advantage of overlapping longer tokens in terms of their combinatorial potential. While single-character overlaps are bounded by the size of the alphabet or the number of unique characters available, longer tokens allow exponentially more possible combinations. This provides a greater potential to span a larger portion of the vocabulary, thereby enabling more effective utilization of the model's capacity.

We also examine whether tokenizer fertility (Ács, 2019; Rust et al., 2021) explains the gains from transliteration. Fertility, defined as the average number of subword tokens a tokenizer generates per word, is commonly used to assess tokenizer quality. Prior work (Rust et al., 2021) suggests that lower fertility is associated with better multilingual performance. Using our trained tokenizers, We compute fertility scores on the NER datasets and analyze their correlation with downstream performance. As shown in Table 4, fertility only partially explains the performance gains and does not fully account for differences across input types. In contrast, vocabulary coverage exhibits a stronger and more consistent correlation—not only across input types, but also across languages within each input type. These results show that the benefit of sharing longer tokens for unseen languages cannot be explained solely by conventional tokenizer quality.

Input Type	Fertility		Vocab. Coverage	
	Coef.	p	Coef.	p
Ortho	-0.2547	0.0008	0.5236	4.6×10^{-12}
IPA	-0.0176	0.8162	0.3248	1.8×10^{-5}
Rom	-0.0797	0.2927	0.3705	9.9×10^{-7}
Cipher	-0.1145	0.1303	0.3405	6.8×10^{-6}
All	-0.1628	1.2×10^{-5}	0.4240	5.0×10^{-30}

Table 4: Spearman's correlation coefficients between downstream performance and two tokenizer-related factors: fertility and vocabulary coverage.

6.3 Shared Phonology: A Path to Longer Tokens

Here, we examine the role of shared phonology introduced by transliteration through the behavior of Cipher. Cipher is designed to isolate the effect of shared phonology from Rom. It uses the same character set as in Rom, but does not encode phonological—or any linguistic—information shared across languages.

Without shared phonology, Cipher struggles to produce longer tokens as shown in Figure 5, and consequently performs worse than Rom. Nevertheless, by eliminating unknown tokens, it performs comparably to IPA and surpasses Ortho on unseen languages. In contrast, although IPA has a relatively high proportion of UNK tokens compared to other transliteration methods, it produces longer shared tokens in unseen languages. These observations suggest that the shared character set alone is insufficient for effective transliteration; rather, a consistent form-meaning mapping across languages is crucial to enable models to form longer tokens, typically through shared phonology.

7 Conclusion

Transliteration has been used to bring together diverse languages using diverse scripts, but its benefits have not been well understood. We examined three aspects of transliteration—shared characters, shared subword tokens, and shared phonology—and analyzed their roles in adapting to languages and scripts unseen during pre-training, by comparing four different input types: Ortho, IPA, Rom, and Cipher. We find that Rom outperforms all other input types, with a greater number of longer tokens shared across languages. While shared characters largely facilitate the adaptation by preventing unknown tokens, we suggest that sharing longer tokens is crucial for effective transliteration that overcomes the script barrier. Our findings suggest that transliteration is effective not simply because of language similarity with pre-trained languages, but because it reshapes token distributions in ways that make multilingual models more adaptable and capable.

8 Limitation

The results reported here are suggestive, but there are two major limitations which prevent us from generalizing them too broadly. First, we only tested one type of transformer model with only subword tokenization scheme. It is possible, for example, that we would have obtained much different results if we had trained character- or byte-level models. Also, we only tested one romanizer and one G2P transducer. It is possible that the results were influenced by the performance of each tool.

9 Ethics Statement

We believe that this research raises no significant ethical concerns or violations of the code of ethics mandated by the Association for Computational Linguistics. The data used in this study, all of which are publicly available, were collected in accordance with legal and institutional protocols, to the best of our knowledge. Furthermore, our use of these resources is compatible with the uses intended by the creators.

References

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.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472, Austin, Texas. Association for Computational Linguistics.

Tyler A. Chang, Catherine Arnett, Zhuowen Tu, and Ben Bergen. 2024. When is multilinguality a curse? language modeling for 250 high- and low-resource languages. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4074–4096, Miami, Florida, USA. Association for Computational Linguistics.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen, and Jaime Carbonell. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3285–3295, Brussels, Belgium. Association for Computational Linguistics.

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.

Wietse de Vries, Martijn Wieling, and Malvina Nissim. 2022. Make the best of cross-lingual transfer: Evidence from POS tagging with over 100 languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7676–7685, Dublin, Ireland. Association for Computational Linguistics.

Yoshinari Fujinuma, Jordan Boyd-Graber, and Katharina Kann. 2022. Match the script, adapt if multilingual: Analyzing the effect of multilingual pretraining on cross-lingual transferability. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1500–1512, Dublin, Ireland. Association for Computational Linguistics.

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.

Jaavid Husain, 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.

Haeji Jung, Changdae Oh, Jooeon Kang, Jimin Sohn, Kyungwoo Song, Jinkyu Kim, and David R Mortensen. 2024. Mitigating the linguistic gap with phonemic representations for robust cross-lingual transfer. In *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, pages 200–211, Miami, Florida, USA. Association for Computational Linguistics.

David Kahn. 1996. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*, revised edition. Scribner, New York.

Colin Leong and Daniel Whitenack. 2022. Phone-ing it in: Towards flexible multi-modal language model training by phonetic representations of data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5306–5315, Dublin, Ireland. Association for Computational Linguistics.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5661–5681, Toronto, Canada. Association for Computational Linguistics.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. 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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

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.

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Benjamin Muller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021. 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.

Hoang Nguyen, Chenwei Zhang, Tao Zhang, Eugene Rohrbaugh, and Philip Yu. 2023. Enhancing cross-lingual transfer via phonemic transcription integration. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9163–9175, Toronto, Canada. Association for Computational Linguistics.

Hoang H Nguyen, Khyati Mahajan, Vikas Yadav, Julian Salazar, Philip S. Yu, Masoud Hashemi, and Rishabh Maheshwary. 2025. Prompting with phonemes: Enhancing LLMs' multilinguality for non-Latin script languages. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11975–11994, Albuquerque, New Mexico. 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.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fred Philippy, Siwen Guo, and Shohreh Haddadan. 2023. Towards a common understanding of contributing factors for cross-lingual transfer in multilingual language models: A review. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5877–5891, Toronto, Canada. Association for Computational Linguistics.

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

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.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

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.

Jimin Sohn, Haeji Jung, Alex Cheng, Jooeon Kang, Yilin Du, and David R Mortensen. 2024. Zero-shot cross-lingual NER using phonemic representations for low-resource languages. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13595–13602, Miami, Florida, USA. Association for Computational Linguistics.

Jian Zhu, Changbing Yang, Farhan Samir, and Jahurul Islam. 2024. The taste of IPA: Towards open-vocabulary keyword spotting and forced alignment in any language. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 750–772, Mexico City, Mexico. Association for Computational Linguistics.

Judit Ács. 2019. Exploring bert's vocabulary. Blog Post.

A Appendix

A.1 Language Selection

To examine the impact on multilingual adaptation that differences in input types have, we selected four language sets : (i) similar languages using the same script (sim-same), (ii) similar languages using diverse scripts (sim-div), (iii) dissimilar languages using the same script (dissim-same), and (iv) dissimilar languages using diverse scripts (dissim-div). These sets were used to train multilingual models with varying linguistic similarities and scripts. For each set, we assigned eight languages based on a computed similarity score as shown in Table 2.

Similar to Chang et al. (2024), we utilized lang2vec (Littell et al., 2017)[5] to compute language similarity. Specifically, we extracted syntactic, geographic, and genetic features from lang2vec and computed cosine similarities, denoted as s_{syn}, s_{geo}, and s_{gen} in Eq. 3. We also defined lexical similarity s_{lex}, which is obtained by calculating the word overlap ratio between training corpora of each language[6]. Finally, we aggregated all similarity scores (i.e., syntactic, geographic, genetic, and lexical) to derive the overall similarity score between two languages:

$$\begin{aligned}\text{sim}_s(x,y) = \text{s}_{syn}(x,y) + \text{s}_{geo}(x,y) \\ + \text{s}_{gen}(x,y) + \text{s}_{lex}(x,y).\end{aligned} \tag{3}$$

With initial set of languages L that are supported by Wikipedia corpus and Epitran, we use average pairwise similarity scores to compute similarity score for a set of languages and obtain an optimal set L_s^*, where $s \in \{\text{sim-same}, \text{sim-div}\}$:

$$\begin{aligned}L_s^* = \arg\max_{\substack{L_s \subset L \\ |L_s|=8}} \Bigg(\frac{1}{|L_s|(|L_s|-1)} \sum_{x \in L_s} \sum_{\substack{y \in L_s \\ y \neq x}} \text{sim}_s(x,y) \\ + \alpha \cdot \Big(\mathbb{1}_{s \in \{\text{sim-div}\}} |\text{SC}_{L_s}| - \mathbb{1}_{s \in \{\text{dissim-div}\}} |\text{SC}_{L_s}| \Big) \Bigg)\end{aligned} \tag{4}$$

.

As for an optimal set L_d^*, where $d \in \{\text{dissim-same}, \text{dissim-div}\}$:

$$\begin{aligned}L_d^* = \arg\min_{\substack{L_d \subset L \\ |L_d|=8}} \Bigg(\frac{1}{|L_d|(|L_d|-1)} \sum_{x \in L_d} \sum_{\substack{y \in L_d \\ y \neq x}} \text{sim}_s(x,y) \\ + \alpha \cdot \Big(\mathbb{1}_{d \in \{\text{sim-div}\}} |\text{SC}_{L_d}| - \mathbb{1}_{d \in \{\text{dissim-div}\}} |\text{SC}_{L_d}| \Big) \Bigg).\end{aligned} \tag{5}$$

To select languages for the sets with same script (i.e., sim-same and dissim-same), we limited the search space to languages that use the Latin script to maximize the number of languages available for similarity-based sampling.

For sets with diverse scripts (i.e., -div), we additionally consider how many different scripts are involved in each set.

A.2 Model Configuration

Table 5 summarizes the key configuration details of our RoBERTa-based model. Number of parameters per model is 109,082,112.

A.3 Training Setup

To investigate the impact of different input types, we pre-trained and fine-tuned a total of 16 models across four distinct input types and language sets. In addition, we trained a SentencePiece BPE tokenizer for each model, fixing the vocabulary size to 30K. Table 6 summarizes the key hyperparameters used in our experiments for both the pretraining phase and the downstream NER task.

[5]Utilizing https://github.com/antonisa/lang2vec

[6]Words are segmented by white spaces.

Parameter	Value
Vocabulary Size	30,000
Hidden Size	768
Hidden Layers	12
Attention Heads	12
Intermediate Size	3072
Activation Function	GELU
Dropout (Hidden/Attention)	0.1
Max Position Embeddings	514

Table 5: Model Configuration

Hyperparameter Sweep We conducted grid search to find learning rates that converges or achieves the best results. For pre-training, the search space was {1e-5, 2e-5, 3e-5, 5e-5, 1e-4, 2e-4, 3e-4} and for NER, it was {3e-5, 5e-5, 1e-4}.

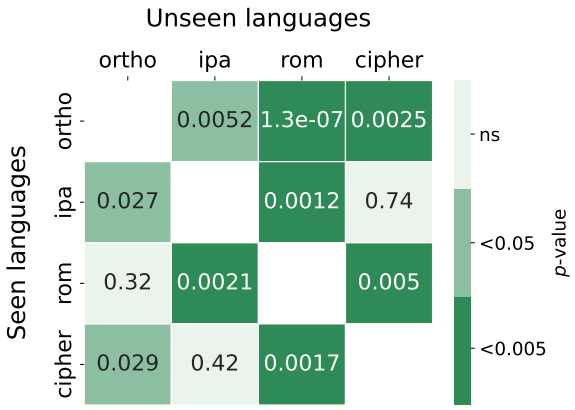
Parameter	Pretraining	NER	XNLI
FP16 Training	True	True	True
Max Seq. Length	512	512	512
Batch Size (per device)	64	64	64
Gradient Accumulation Steps	1	-	-
Warmup Steps	50	-	-
Learning Rate	1e-4	5e-5	3e-5
Weight Decay	0.01	0.01	0.01
LR Scheduler Type	Linear	-	-
MLM Probability	0.15	-	-
Epochs	300	20	5
Log Interval	-	1	1
GPU Resources	4 NVIDIA L40S	2 NVIDIA RTX A6000	1 NVIDIA RTX A6000

Table 6: Training Configurations

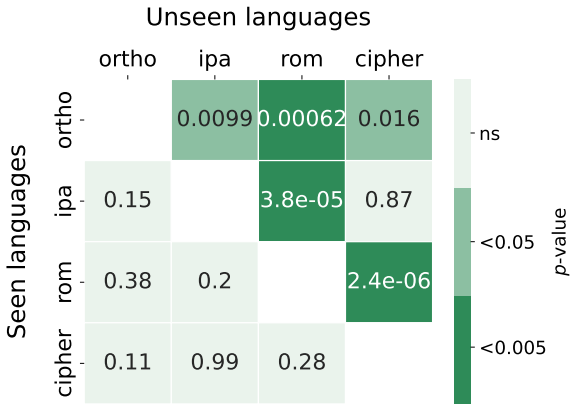
A.4 Substitution Cipher (Cipher)

A substitution cipher is a method from cryptography where units of plaintext are replaced with ciphertext according to a predefined rule or key. We apply substitution cipher to the Romanized text to remove encoded phonological information.

Specifically, we use the Caesar cipher (Kahn, 1996), a simple substitution encryption technique that shifts each letter in the text by a fixed number of positions in the Latin alphabet. For each language, we assign an integer that determines the shift from the current position of each letter. For example, if English is assigned the integer 4, the word 'apple' would be represented as 'ettpi', with each letter replaced by the one four positions ahead in the alphabet.



(a) NER (WikiAnn)



(b) XNLI

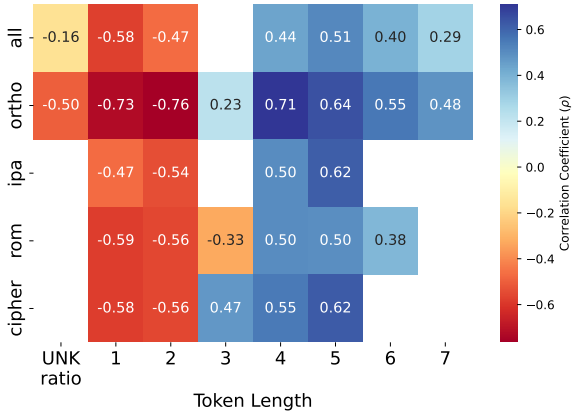
Figure 8: P-values for paired t-tests on performances for both tasks. Upper triangular elements show results across unseen languages and lower triangular elements show those across seen languages.

A.5 P-values of Paired t-tests

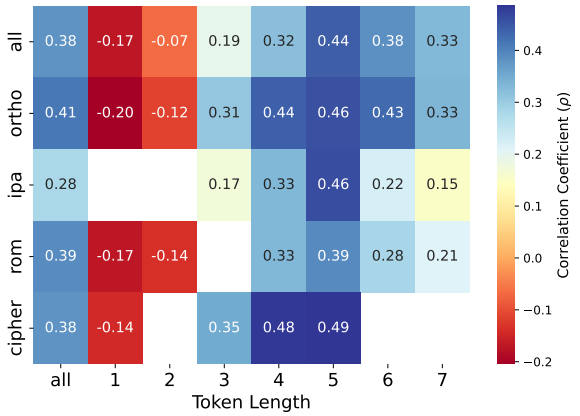
Table 3 presents the scores of NER and XNLI tasks for different input types across various language settings. To assess the significance of the observed differences, we performed paired t-tests. Figure 8 displays the corresponding p-values derived from these tests. The results show that transliteration (IPA, Rom, Cipher) significantly benefits Ortho for unseen languages. For NER, Rom and Ortho perform on par for seen languages, while there is no significant difference across any input types for XNLI.

A.6 Vocabulary Overlap Analysis

Figure 9 shows heatmap of correlation coefficients between overlap ratio and NER F1 score, where overlap ratio is computed differently from Section 4.2. Figure 10 shows the correlation analysis for XNLI task.



(a)



(b)

Figure 9: Spearman's correlation coefficients between overlap ratio and NER F1 score. (a) Computed by type ratio. Leftmost column indicates correlations with UNK token ratio. (b) Computed by overlap ratio across all source languages. Leftmost column indicates correlations with total number of overlaps regardless of token length.

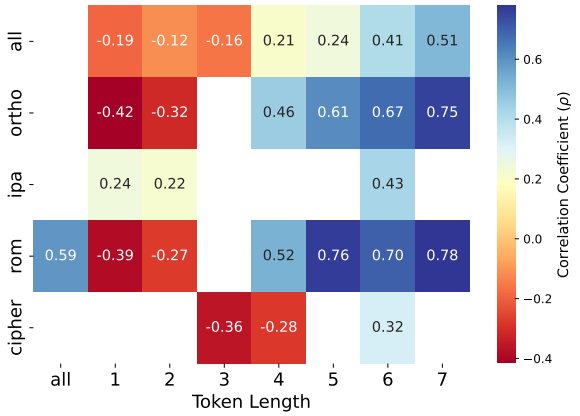
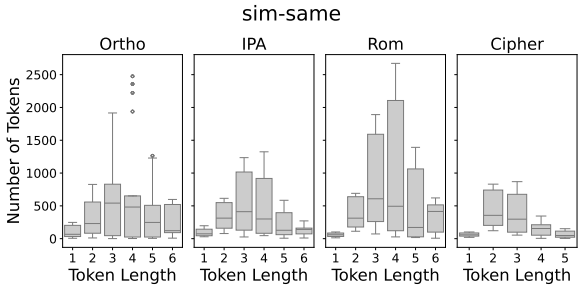
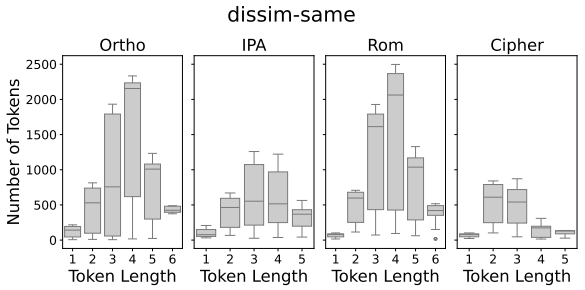


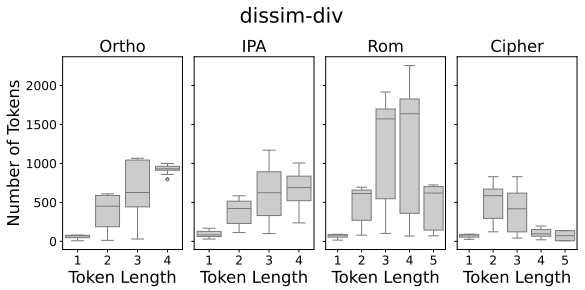
Figure 10: Spearman's correlation coefficients between overlap ratio and XNLI Accuracy. Overlap ratio computed as in Section 4.2.



(a)



(b)



(c)

Figure 11: Number of unique tokens by length for unseen target languages (WikiAnn dataset), using models trained on (a) sim-same (b) dissim-same (c) dissim-div languages.

A.7 Number of Unique Tokens

Figure 11 shows the distribution of token lengths for models trained on each language set.

A.8 Vocabulary Coverage

Here we provide more plots for vocabulary coverage. Figure 12 shows vocabulary coverage by length over unseen languages using XNLI dataset. The pattern is similar to that of NER (WikiAnn) dataset, where Rom exhibits evidently high coverage compared to other input types. Figure 13 shows vocabulary coverage by length over seen languages using WikiAnn dataset. While Rom still shows the most coverage, Ortho also takes large portion when the languages share same script (i.e., sim-same, dissim-same).

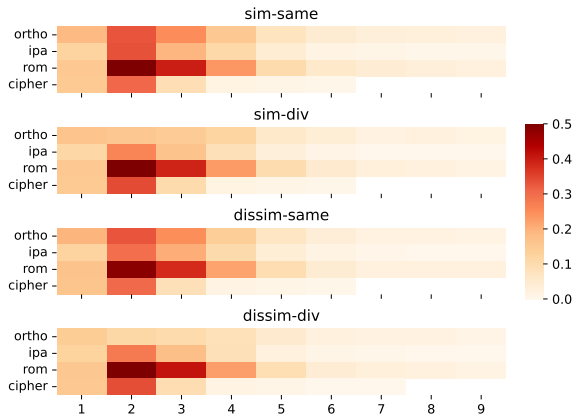


Figure 12: Vocabulary coverage over unseen languages on XNLI dataset computed as in Section 6.2.

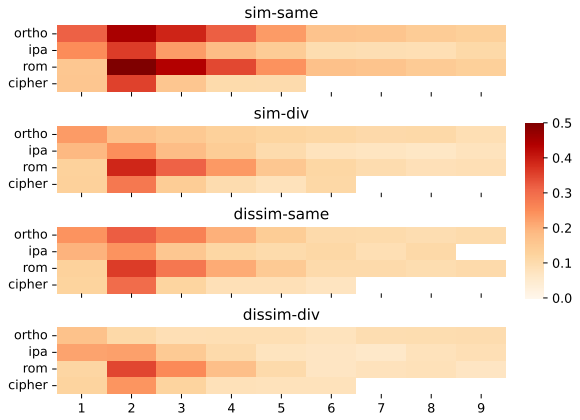


Figure 13: Vocabulary coverage over seen languages on WikiAnn dataset computed as in Section 6.2.

A.9 External Tools for Transliteration

In this study, we used Epitran and Uroman as transliteration tools to unify script and facilitate multilingual processing. These tools are widely used for converting text into standardized phonemic or Romanized forms, which aids in cross-lingual learning and transferability. Below, we describe their functionalities and implementation details.

Epitran(Mortensen et al., 2018) is a tool for grapheme-to-phoneme (G2P) conversion, capable of converting text into the International Phonetic Alphabet (IPA) representations. It can be downloaded from the link below https://github.com/dmort27/epitran

Uroman(Hermjakob et al., 2018) is a universal transliteration tool that converts text from various scripts into a Romanized format. It can be downloaded from the link below https://github.com/isi-nlp/uroman

A.10 Datasets

In Table 7, the specific number of datasets per corresponding language is provided. For pre-training, we utilized sampled version of preprocessed Wikipedia corpus from Huggingface[7].

We limited each language with its number of words around 10M[8]. For those languages with less number of tokens than 10M, we kept all the documents and oversampled during training, to match the model's exposure to all languages. For downstream task, we utilized WikiAnn (Pan et al., 2017; Rahimi et al., 2019) dataset for named entity recognition. In order to train the model with different input types, we converted all datasets into each corresponding input type.

Wikipedia corpora used for pre-training are licensed under the GNU Free Documentation License (GFDL) and the Creative Commons Attribution-Share-Alike 3.0 License. License type for WikiAnn dataset is ODC-BY.

[7]https://huggingface.co/datasets/wikimedia/wikipedia

[8]For each language, we randomly shuffled the order of the documents, and iterated over each document, counting the words segmented by whitespaces. We stop adding the documents when adding the number of words of the last document exceeds 10M.

Lang	Dataset	# Train	# Validate	# Test	Lang	Dataset	# Train	# Validate	# Test
am	wikipedia	5328	-	-	my	wikipedia	34309	-	-
	wikiann	100	100	100		wikiann	100	100	100
ar	wikipedia	-	-	-	or	wikipedia	11018	-	-
	wikiann	20000	10000	10000		wikiann	100	100	100
bn	wikipedia	28496	-	-	pl	wikipedia	-	-	-
	wikiann	10000	1000	1000		wikiann	20000	10000	10000
ca	wikipedia	26031	-	-	pt	wikipedia	26510	-	-
	wikiann	20000	10000	10000		wikiann	20000	10000	10000
ceb	wikipedia	22724	-	-	ro	wikipedia	28890	-	-
	wikiann	100	100	100		wikiann	20000	10000	10000
de	wikipedia	30460	-	-	ru	wikipedia	32636	-	-
	wikiann	20000	10000	10000		wikiann	20000	10000	10000
es	wikipedia	25727	-	-	si	wikipedia	23084	-	-
	wikiann	20000	10000	10000		wikiann	100	100	100
fi	wikipedia	36190	-	-	so	wikipedia	5204	-	-
	wikiann	20000	10000	10000		wikiann	100	100	100
fr	wikipedia	25353	-	-	sq	wikipedia	27406	-	-
	wikiann	20000	10000	10000		wikiann	5000	1000	1000
hi	wikipedia	25492	-	-	sr	wikipedia	29961	-	-
	wikiann	5000	1000	1000		wikiann	20000	10000	10000
hr	wikipedia	30764	-	-	sv	wikipedia	29839	-	-
	wikiann	20000	10000	10000		wikiann	20000	10000	10000
ilo	wikipedia	5828	-	-	sw	wikipedia	25911	-	-
	wikiann	100	100	100		wikiann	1000	1000	1000
ka	wikipedia	33713	-	-	te	wikipedia	28543	-	-
	wikiann	10000	10000	10000		wikiann	1000	1000	1000
ko	wikipedia	38885	-	-	th	wikipedia	76083	-	-
	wikiann	20000	10000	10000		wikiann	20000	10000	10000
lij	wikipedia	4002	-	-	ur	wikipedia	23568	-	-
	wikiann	100	100	100		wikiann	20000	1000	1000
lt	wikipedia	32836	-	-	uz	wikipedia	29833	-	-
	wikiann	10000	10000	10000		wikiann	1000	1000	1000
lv	wikipedia	31152	-	-	-	-	-	-	-
	wikiann	10000	10000	10000		-	-	-	-

Table 7: Statistic of transliterated dataset. All dataset exist in four parallel versions ; original Orthographic, phonemic IPA, Romanized, and Cipher transcribed version. - refers to unavailable values. The wikipedia dataset is used for pre-training without validation or test. Languages ‘ar’ and ‘pl’ do not have available wikipedia dataset for pre-train.