Scripts Through Time: A Survey of the Evolving Role of Transliteration in NLP

Thanmay Jayakumar[1,2], Deepon Halder[1,3], Raj Dabre[1,2]

[1]Nilekani Centre at AI4Bharat, [2]Indian Institute of Technology Madras, India,
[3]Indian Institute of Engineering, Science and Technology, Shibpur

Abstract

Cross-lingual transfer in NLP is often hindered by the "script barrier" where differences in writing systems inhibit transfer learning between languages. Transliteration, the process of converting the script, has emerged as a powerful technique to bridge this gap by increasing lexical overlap. This paper provides a comprehensive survey of the application of transliteration in cross-lingual NLP. We present a taxonomy of key motivations to utilize transliterations in language models, and provide an overview of different approaches of incorporating transliterations as input. We analyze the evolution and effectiveness of these methods, discussing the critical trade-offs involved, and contextualize their need in modern LLMs. The review explores various settings that show how transliteration is beneficial, including handling code-mixed text, leveraging language family relatedness, and pragmatic gains in inference efficiency. Based on this analysis, we provide concrete recommendations for researchers on selecting and implementing the most appropriate transliteration strategy based on their specific language, task, and resource constraints.

1 Introduction

Cross-lingual transfer, a major driving factor in multilingual Natural Language Processing (NLP), has enabled significant advancements for numerous languages, especially low-resource languages, by leveraging related high-resource language capabilities, predominantly English (Johnson et al., 2017; Zoph et al., 2016; Conneau and Lample, 2019). However, the efficacy of this transfer is often impeded by the script barrier: When a low-resource target language is written in a different script from the high-resource source language, transfer performance is hindered, even if the languages are related. Lexical overlap, a key factor for successful cross-lingual transfer, is minimal between different scripts (Pires et al., 2019; Anastasopoulos and Neubig, 2019; Muller et al., 2021). Token representations from different scripts can be almost perfectly linearly separated, indicating that models struggle to learn a common representation space (Wen-Yi and Mimno, 2023).

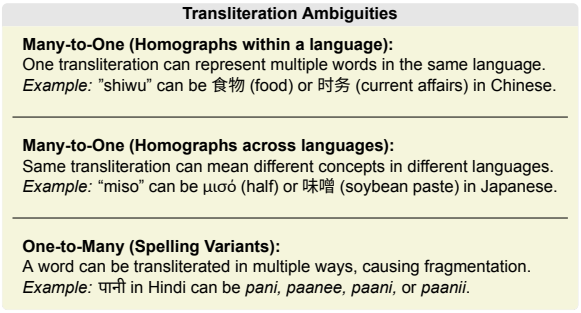


Figure 1: Illustration of common transliteration ambiguities

Transliteration, the process of converting one writing system to another, thus has emerged as a practical solution for mitigating cross-script incompatibility in NLP. By converting text into a common script, transliteration increases the lexical overlap between languages, thereby facilitating knowledge transfer (Pires et al., 2019; Amrhein and Sennrich, 2020). It is a fast, accurate, and data-efficient method, and does not require parallel corpora (Liu et al., 2024b).

Apart from the direct application of increasing lexical overlap, transliterations improve cross-lingual transfer through several deeper mechanisms that have been explored in the literature. The foundational concept lies in the idea of anchor points, which are identical strings that appear in multiple languages and directly tie meaning across different languages. Transliteration can be seen as a method to artificially create a much larger set of these shared tokens or subwords that serve a similar anchoring function, especially for related languages whose similarities are obscured by different

Findings of the Association for Computational Linguistics: ACL 2026, pages 23511–23524

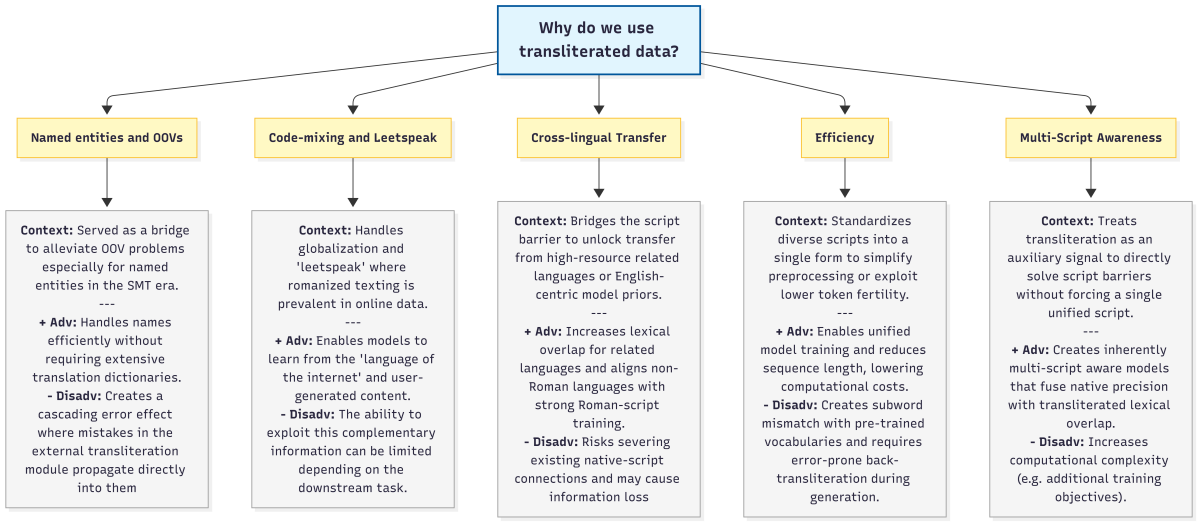


Figure 2: A taxonomy of the key motivations as to why transliterated data may be useful.

scripts (Conneau et al., 2020).

However, transliteration is not a universal solution and can degrade performance. This is particularly evident with logographic languages like Chinese, where transliterating into a phonetic script like Latin removes crucial semantic and contextual nuances, leading to ambiguity. A summary of information loss that arises due to transliteration is presented in Figure 1.

This survey paper examines the multifaceted role of transliteration in language models and NLP. We explore the evolution of methods that leverage transliteration, analyze the conditions when it is beneficial and when it is not, and review its impact on various downstream tasks and models.

Our contributions are summarized as follows:

- We provide a taxonomy of the key motivations for using transliteration, from overcoming poor vocabulary coverage for unseen scripts to leveraging linguistic relatedness between languages.
- We present a comprehensive overview of different approaches for incorporating transliteration, including its use as a data preprocessing step, as a parallel auxiliary input, or in advanced multi-script architectures that aim to combine its strengths with other methods.
- We offer concrete recommendations for NLP researchers, guiding the choice between these strategies based on factors like language relatedness, resource availability, and the specific downstream NLP task.
- We discuss the growing role of romanization in particular as opposed to other scripts. Beyond in-context learning, we situate transliteration in current LLM paradigms, and address whether they are still necessary.

2 Taxonomy of Transliteration-based Strategies to Improve NLP Tasks

The literature for this survey was gathered through an iterative search process[1], beginning with the keywords "transliteration" and "romanization" across academic databases like the ACL Anthology, Semantic Scholar, and arXiv.org. This initial set of papers was then expanded by following their citation graphs.

To maintain a focused review, we emphasize on works whose novelty lie in applying transliterations to improve language models rather than those which propose methods to improve transliteration itself.

It is worth highlighting the role of code-mixing as the phenomenon frequently co-occurs with transliteration, however, the two remain fundamentally orthogonal phenomena. For a comprehensive overview of code-mixing, we refer the reader to dedicated surveys in the field (Winata et al., 2023; Sheth et al., 2026), as we omit intra-script code-mixing (e.g., between two Latin-script languages)

[1]We thank Jaavid Aktar Husain for providing the initial collection of papers for this study.

from this paper and maintain a strict focus on cross-script conversion.

Table 1 summarizes the papers covered in this survey.

2.1 Motivations to Integrate Transliterations

We identify five broad, chronologically-evolving motivations for utilizing transliteration in NLP as summarized in Figure 2.

2.1.1 Named entities and OOVs

The earliest applications arose in the pre-neural era, particularly within Statistical Machine Translation (SMT). These systems were highly susceptible to out-of-vocabulary (OOV) errors, a problem especially frequent with named entities which are rarely found in translation dictionaries (Kashani et al., 2007; Durrani et al., 2014). One approach to address this was by integrating a transliteration component into SMT systems to handle these names and this yielded significant performance improvements. This core utility persisted into the early neural era, where named entities remained a challenge that transliteration was well-suited to address (Du and Way, 2017; Aqlan et al., 2019).

2.1.2 Code-mixing and Leetspeak

A second major driver emerged with globalization and the rise of code-mixing. Especially in bilingual environments, users increasingly began writing their native languages in romanized forms (Kim and Shin, 2013), often mixed with English (van der Wees et al., 2016; Mukherjee et al., 2019). Consequently, researchers started using transliteration, specifically to English (Latin script), as a primary method to normalize this code-mixed data, enabling models to learn from this new linguistic phenomenon.

2.1.3 Cross-lingual Transfer

With the advent of multilingual pretrained models, the motivation shifted again. The community began to see transliteration as a key technique to increase lexical overlap between typologically related languages written in different scripts (like Turkish and Uyghur) (Muller et al., 2021; Khemchandani et al., 2021), thereby unlocking cross-lingual transfer capabilities that were previously inhibited. Furthermore, romanization allows even transfer from typologically unrelated languages (e.g., Hindi) by aligning with the strong-script priors of English-centric models, effectively capitalizing on the abundance of incidental romanized content present in pretraining corpora (Husain et al., 2024).

2.1.4 Efficiency in Training and Inference

This led to a fourth, more pragmatic motivation: unification and efficiency. Having a common script simplifies preprocessing, facilitates uniform model training, reduces the complexity of handling multiple native script conversions, and overcomes orthographic differences (Soni and Bhattacharyya, 2024). Further, using scripts that have lower fertility can significantly reduce the computational overhead and time during inference (Nag et al., 2024). This means that it applies even if the languages corresponding to the scripts are unrelated.

2.1.5 Multi-script Awareness

Most recently, the focus has evolved once more. Instead of merely circumventing the script barrier, the latest research aims to solve it directly. This involves treating transliteration as an auxiliary signal within sophisticated architectures, designing models that are inherently multi-script aware rather than dependent on a single, unified script (Liu et al., 2024a; Xhelili et al., 2024).

2.2 Approaches to Integrate Transliterations

Various approaches have been proposed to integrate transliteration into models. These methods differ based on how transliterated forms are introduced: at the data-level, input-level, architecture-level, or at the inference-level as presented in Figure 3.

For simplicity, the figures show only the original and transliterated form. We stress that this is not a restriction as these methods can accommodate multiple alternative scripts in a similar fashion.

2.2.1 Data-level Integration

Direct Transliteration A rather direct approach is to transliterate the entire training corpus or a portion of it without any change to the architecture or model. In some cases, only transliterations are used as the data mixture **(Transliterated Corpora)**, and in some other, the transliterations are augmented with the original data mixture **(Transliteration Augmented Data)**. The data can then be used to continue training the model, or it can be used more specifically to adapt the model's tokenizer by expanding its vocabulary (i.e., **Vocabulary Augmentation**).

	Integration Approach										Motivation							Script					Architecture		Task	
	Transliterated Corpora	Transliteration Augmented Corpora	Vocabulary Augmentation	Embedding Concat/Fusion	Prompting Strategies	Multi-ensemble	Self-ensemble	Multi-encoder	Adapters	Additional Objective	Named-entity and OOV	Code-mixed and Leetspeak	Cross-script Similarity	Transfer from English-centric	Unifying Preprocessing	Multi-script Awareness	Inference Efficiency	Latin	Non-Latin	Enc-only	Dec-only (LLM)	Enc-Dec	NLU	NLG
(Kashani et al., 2007)						✓					✓							✓				✓		✓
(Nakov and Ng, 2009)	✓												✓					✓				✓		✓
(Bhargava and Kondrak, 2011)						✓					✓							✓				✓		✓
(Semmar and Saadane, 2013)											✓							✓		✓			✓	
(Kim and Shin, 2013)		✓										✓						✓		✓			✓	
(Durrani et al., 2014)	✓					✓					✓							✓				✓		✓
(Durrani and Koehn, 2014)	✓					✓					✓		✓					✓				✓		✓
(Sadamitsu et al., 2016)	✓					✓					✓							✓				✓		✓
(Lin et al., 2016)	✓					✓					✓			✓				✓		✓			✓	
(van der Wees et al., 2016)		✓										✓						✓				✓		✓
(Du and Way, 2017)			✓	✓							✓							✓				✓		✓
(He et al., 2017)						✓					✓							✓		✓			✓	
(Guellil et al., 2018)		✓				✓						✓						✓		✓			✓	
(Mrini and Bond, 2018)	✓												✓					✓		✓			✓	
(Dabre et al., 2018)	✓												✓		✓				✓			✓		✓
(Aqlan et al., 2019)			✓								✓							✓				✓		✓
(Mukherjee et al., 2019)		✓										✓						✓		✓			✓	
(Gheini and May, 2019)	✓														✓			✓				✓		✓
(Johnson et al., 2019)		✓						✓						✓				✓		✓			✓	
(Rijhwani et al., 2019)													✓						✓	✓			✓	
(Briakou and Carpuat, 2019)	✓		✓										✓		✓			✓				✓		✓
(Vania et al., 2019)	✓												✓		✓			✓		✓			✓	
(Liu et al., 2019)	✓			✓									✓					✓				✓		✓
(Khakhmovich et al., 2020)	✓										✓			✓					✓	✓			✓	
(Amrhein and Sennrich, 2020)	✓												✓		✓			✓				✓		✓
(Song et al., 2020)			✓										✓						✓			✓		✓
(Murikinati et al., 2020)	✓												✓	✓				✓	✓			✓		✓
(Goyal et al., 2020)	✓												✓		✓				✓			✓		✓
(Sai and Sharma, 2021)		✓										✓						✓		✓			✓	
(Koneru et al., 2021)	✓												✓		✓			✓				✓		✓
(Khemchandani et al., 2021)	✓												✓						✓			✓		✓
(Dhamecha et al., 2021)	✓												✓		✓				✓	✓			✓	
(Muller et al., 2021)	✓												✓	✓				✓				✓		✓
(Khatri et al., 2021)	✓												✓		✓				✓			✓		✓
(Chau and Smith, 2021)	✓		✓										✓	✓				✓		✓			✓	
(Sun et al., 2022)	✓			✓		✓	✓	✓			✓		✓		✓			✓	✓			✓		✓
(Palanikumar et al., 2022)		✓										✓						✓		✓			✓	
(Das et al., 2022)		✓										✓						✓		✓			✓	
(Laskar et al., 2022)		✓													✓				✓			✓		✓
(Roychoudhury et al., 2022)	✓														✓		✓	✓		✓			✓	
(Dabre et al., 2022)	✓												✓		✓				✓			✓		✓
(Purkayastha et al., 2023)	✓								✓					✓	✓		✓	✓		✓			✓	
(Moosa et al., 2023)													✓		✓		✓	✓		✓			✓	
(Micallef et al., 2023)	✓												✓				✓		✓	✓			✓	
(Doddapaneni et al., 2023)		✓											✓				✓		✓	✓			✓	
(Rajalakshmi et al., 2024)		✓										✓						✓		✓			✓	
(Tabassum et al., 2024)		✓										✓						✓		✓			✓	
(Soni and Bhattacharyya, 2024)	✓												✓		✓		✓	✓				✓		✓
(Zhou et al., 2024)	✓												✓	✓	✓			✓		✓			✓	
(Husain et al., 2024)	✓											✓	✓	✓	✓		✓	✓			✓		✓	✓
(Ma et al., 2025)					✓									✓				✓			✓		✓	
(Al Ghanim et al., 2024)	✓				✓							✓		✓				✓			✓			✓
(Nag et al., 2024)												✓		✓			✓	✓			✓			✓
(Salehi and Jacobs, 2024)		✓	✓										✓					✓		✓			✓	
(Lee et al., 2024)	✓		✓						✓							✓		✓		✓			✓	
(Liu et al., 2024a)										✓				✓		✓		✓		✓			✓	
(Xhelili et al., 2024)				✓						✓				✓		✓		✓		✓			✓	
(Liu et al., 2024b)	✓									✓				✓		✓	✓	✓		✓			✓	
(Chari et al., 2025)	✓													✓		✓		✓		✓			✓	
(Liu et al., 2025)			✓											✓	✓	✓		✓		✓			✓	
(Zhuang et al., 2025)	✓														✓		✓	✓	✓		✓		✓	✓
(Jung et al., 2026)	✓												✓		✓			✓	✓	✓			✓	

Table 1: An overview of papers applying transliteration to improve NLP task performance.

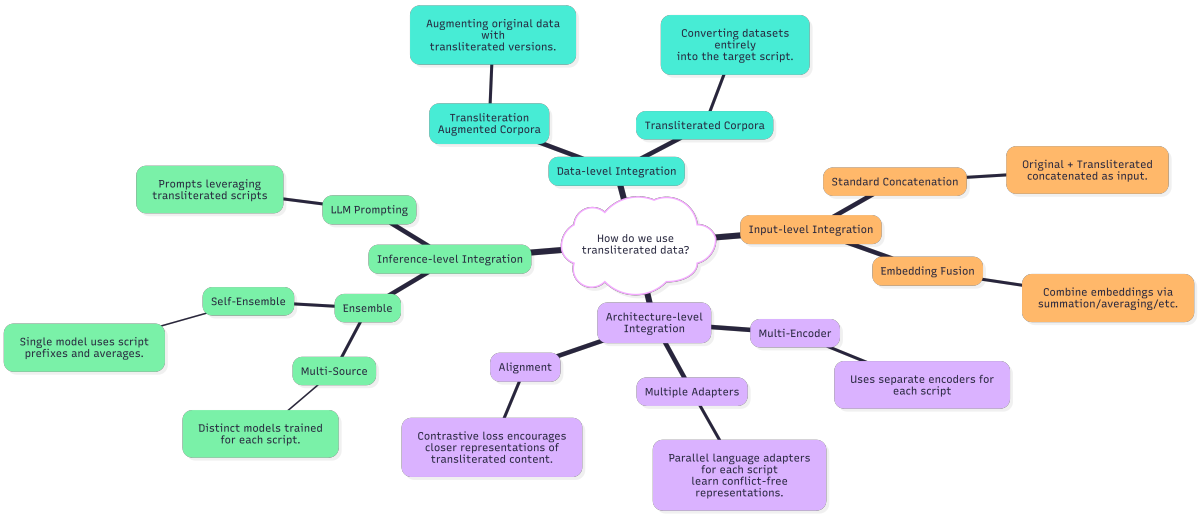


Figure 3: A taxonomy of the key approaches as to how transliterated data may be integrated.

Training Mix

Original Data ⊕ Transliterated Data
(Or only Transliterated Data)

2.2.2 Input-level Integration

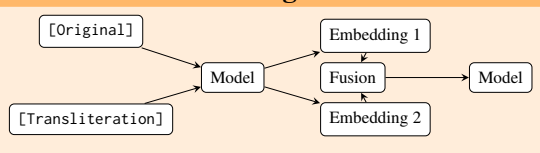
Direct Concatenation Another technique is to concatenate the original sentence with its transliterated counterpart before passing it to the model. Similar to the previous method, no architectural change is required, however, as the concatenated input becomes longer, the computation becomes more expensive.

Input

`[Original] <SEP> [Transliteration]`

Embedding Fusion Unlike concatenation, which extends the sequence length, embedding fusion integrates transliteration at the vector level. In this approach, the model retrieves embeddings for both the original script tokens and their transliterated counterparts simultaneously. These vectors are then combined - typically via summation or averaging. This method maintains the original sequence length, thereby avoiding increased computational cost during self-attention, while still enriching the input representation with cross-script features.

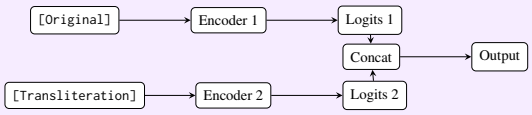
Embedding Fusion



2.2.3 Architecture-level Integration

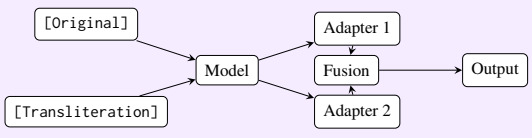
Multi Encoder Some works employ a multi-encoder strategy where each encoder attends to one type of input script. Different attention mechanisms can be seen in such cases based on the interaction between the encoders as illustrated by Libovický et al. (2018). However, since this involves non-trivial changes to the architecture, it is not feasible when employed to existing models.

Multi Encoder



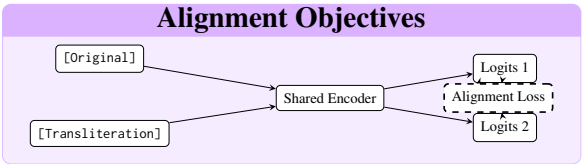
Script Adapters This approach introduces separate, parallel language adapters on top of a multilingual model, training one adapter exclusively on the native script and another on its transliteration. This language module separation allows each adapter to learn a conflict-free representation for its specific script. A fusion mechanism then combines the outputs from these two adapters, leveraging the complementary knowledge from both scripts for the task.

Script Adapters



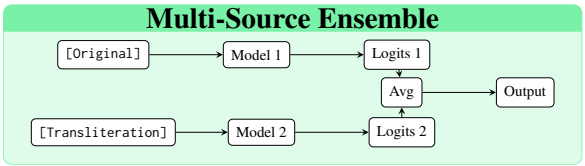
Alignment Objectives This method has been shown to encourage transliterated and original representations to have closer representations generally using a contrastive loss, typically in addition

to the language modeling objective. This regularization helps the model become invariant to orthographic differences while still leveraging both input forms during training. During inference, any input form may be used keeping in mind that same content in similar script are made to have closer representations.

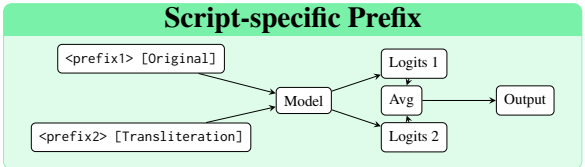


2.2.4 Inference-level Integration

Multi-source Ensemble An alternative line of work adapts an ensemble paradigm to a multi-source setting, where each model is trained using a distinct transliteration form of the input. At test time, the corresponding transliteration is fed to each model, and the resulting log probabilities are averaged prior to decoding. This strategy relies on a common target vocabulary across models to ensure compatibility during the averaging step. Although it can enhance translation quality, the approach is computationally demanding due to the need to train and store several trained models.



Script-specific Prefix Some approaches opt to train a single model on a mixture of different input variants, each prefixed with a special token indicating the script or transliteration type (e.g., `<prefix1>` or `<prefix2>`). This acts as a signal for the model to condition on the script-specific form. At inference time, the same model is used to process each variant of the test sentence separately, and their output log probabilities are averaged before decoding, in a fashion similar to multi-source ensembling. This avoids training multiple models while still leveraging diverse input signals, and thus may also be referred to as multi-source **self ensemble** in certain works.



LLM Prompting With the advent of large language models (LLMs), in-context learning and other prompting strategies has emerged as a strong paradigm, enabling models to adapt to new tasks at inference time using only a few examples or instructions. Building on this, recent work explores multi-script prompting, where prompts leverage both the original script and its transliterations. Further, models often treat different scripts as distinct distributions, leading to divergent behavioral patterns that can undermine semantic alignment or inadvertently circumvent safety guardrails trained on formal scripts.

Prompt

Respond based on the following examples in different scripts:

```
{{Prompt in Script 1}}
{{Prompt in Script 2}}
```

3 Which is the Best Approach to Integrate Transliterated Data?

To determine the most effective way to incorporate transliterated data, Sun et al. (2022) conduct a systematic comparison of multiple architectures presented in Section 2.2, namely **Straight Concatenation**, **Multi-Encoder**, **Multi-source Ensemble**, **Self-Ensemble**, and **Direct Transliteration** taking the task of multilingual machine translation. In addition to comparing these different architectures, they also investigate four primary methods for combining the original input script with alternative signals like IPA, romanization, or in-family script representations.

Their findings showed a clear winner. Both Direct Concatenation and Multi-Encoder Architectures provided little to no benefit, with concatenation bringing only marginal gains and the multi-encoder models achieving similar performance to the much smaller self-ensemble model. The Multi-Source Self-Ensemble consistently outperformed other methods, improving over strong ensemble baselines on both Indic and Turkic language families on in-family scripts. The authors highlight that this approach is particularly effective due to its architectural simplicity, as it requires no complex changes and can be added to any existing translation system.

In the context of **Adapters** (Section 2.2.3), Lee et al. (2024) utilize separate script adapters to navigate the trade-off between increasing lexical

overlap (via transliteration) and maintaining semantic precision (via native vocabularies). **Direct Transliteration** facilitates transfer by creating shared tokens, but risks introducing "false positives" as seen in Figure 1. In contrast, **Vocabulary Augmentation** preserves script integrity but fails to leverage shared semantics effectively.

Put differently, transliteration adjusts the data to the model's needs and vocabulary augmentation adjusts the model to the data's needs. A naive combination of the two therefore often leads to negative interference as transliteration already functions as a form of implicit vocabulary augmentation (Chau and Smith, 2021). This occurs when a single model attempts to optimize for distinct script representations simultaneously causing failure due to the emergence of ambiguities (Lee et al., 2024). Thus, using script-based adapters prevents this gradient interference and allows the model to subsequently fuse the strengths of both the transliterated and native representations.

Alternatively, recent work suggests that conflict between scripts can be mitigated without architectural separation by employing contrastive **Alignment Objectives** (Section 2.2.3). Xhelili et al. (2024) propose improving cross-lingual alignment by explicitly minimizing the distance between native text and its transliteration. They achieve this through a combined objective: a sequence-level contrastive loss that aligns the global representations of the two scripts, and a token-level Transliteration Language Modeling (TLM) loss.

Crucially, this gives a newer perspective to transliteration concatenation (Section 2.2.2). While standard concatenation acts as a static input, here it serves as a dynamic training signal via the TLM objective. By leveraging TLM to force cross-script attention, the model achieves alignment during training without the inference-time computational overhead of processing concatenated sequences.

While the architectural comparisons (Sun et al., 2022) were performed on encoder-decoder models, and the adapter (Lee et al., 2024) and alignment studies (Liu et al., 2024b) used encoder-only models, both of these experimental setups lack a comparison for transliteration integration in decoder-only models that are prevalent today.

Further, the evaluations of these techniques have been concentrated on a narrow set of tasks. The work on encoder-only models has largely focused on demonstrating performance on NER, Dependency Parsing, and POS tagging, while studies on encoder-decoder models have primarily experimented with Neural Machine Translation (NMT).

4 Insights from Literature

Although there is no single 'best' integration method that applies to every scenario, we can derive a few practical guidelines based on the trade-offs discussed above. First, we recommend eliminating modified architectures in favor of standard architectures whenever possible; this ensures model reusability and compatibility with existing systems. This preference for architectural minimalism explains the success of prior works utilizing direct transliteration, alignment objectives, self-ensembles, and LLM prompting. With the architecture fixed, the critical question then becomes one of utility: *"For which languages and tasks is transliteration beneficial?"*

4.1 When is Transliterated Corpora Useful?

As demonstrated by Muller et al. (2021), converting a low-resource language into the script of a high-resource relative (e.g., Uyghur to Latin to match Turkish) can significantly boost mBERT performance by maximizing vocabulary overlap. Conversely, transliteration can be actively harmful if it breaks an existing link between related languages that a model has already learned. For instance, transliterating Mingrelian into the Latin script severs its connection to Georgian (which uses the Georgian script and is present in mBERT's pretraining), thereby harming performance.

Consequently, the decision to transliterate for NLU should be governed by whether the script conversion strengthens or disrupts the linguistic signals available to the model. For encoder-based architectures, utility is maximized when transliteration bridges the gap to a high-resource anchor language without discarding essential semantic features. However, these insights are largely derived from NLU tasks where the model's output is a class label. This changes significantly when we move to generative tasks.

4.2 Transliteration: Solution or Shortcut?

While transliteration bridges the script barrier for understanding, it creates a usability hurdle for generation: end-users typically require native script

output, thus necessitating a post-processing step to restore the original script, which can be susceptible to information loss (Soni and Bhattacharyya, 2024). To address this, we recommend more sophisticated methods that aim to make the model itself multi-script aware, still keeping the core architecture, input, and output processing unchanged. Script adapters or alignment objectives are some ways to solve this. Alternately, Zhuang et al. (2025) propose a Huffman-based framework that guarantees 100% lossless back-transliteration via a greedy mapping strategy, offering an architecture-agnostic alternative to more complex interventions. Compared to traditional vocabulary adaptation, this strategy proves highly effective for generation, achieving superior BLEU scores in low-resource NMT.

4.3 Why Transliteration Works

While it is empirically established that transliteration can help improve task performance due to lexical overlap and shared script/phonology, the degree to which these factors contribute at a tokenizer level is studied by Jung et al. (2026). In this work, a controlled study is performed on the effect of three types of transliteration (complemented with orthography) - romanization, IPA, and substitution cipher - and pretraining an encoder model on each of these four input types, giving various sets of language families, similarity scores, and scripts a shared input representation. Importantly, the substitution cipher serves as a base case to bring about character overlap, but not encode semantic or linguistic information like IPA or romanization.

Building on this limitation of the substitution cipher, the study concludes that romanization is the most effective approach that improves task performance, and find it successfully combines a restricted character set with cross-lingual phonological information. This integration allows the tokenizer to form a greater number of longer subword tokens that maintain semantic consistency across diverse languages. These longer shared tokens drive performance gains by significantly increasing the model's vocabulary coverage and maximizing the utilization of its embedding space. Ultimately, transliteration works primarily by reshaping token distributions to enhance overall model adaptability, independent of the inherent linguistic similarity between the pre-trained and target languages.

4.4 Cross-lingual Alignment and Transfer

Research indicates that adding transliterations effectively increases similarity scores across languages, but it often inflates scores for random sentence pairs just as much as correct ones, introducing noise rather than improving the model's ability to align semantic meaning (Liu et al., 2024b). To solve this, the authors employ auxiliary alignment objectives by teaching the model to explicitly differentiate matched translations from random pairings.

Through this objective, the transliterated text acts as a structural intermediary that successfully aligns the original scripts via shared lexical overlap. Furthermore, this relationship is complicated by the finding that even improved sequence-level script alignment does not consistently yield better zero-shot downstream performance, suggesting that the mechanisms driving alignment and effective task transfer are distinct and completely separate.

4.5 Romanization: Form over Fidelity

Beyond leveraging language relatedness, there are pragmatic reasons to use romanization instead of other transliterating to other scripts. We ask *"Why do so many works use romanization when it may not be a relevant script and can cause information loss?"* (see Table 1 for ambiguities and the sheer amount of papers using Latin script in Table 1). Based on our analysis, we present four major reasons.

First, most LLMs are English-centric and are inherently better at processing Latin script, as their training corpora often have limited to non-existent data for non-Latin scripts (Ma et al., 2025). Second, for many languages, text tokenization exhibits high fertility, meaning it breaks into a large number of subwords. Romanizing the text can reduce this token fertility by a factor of 2x-4x (Husain et al., 2024) and this is a significant advantage as it reduces inference time, and in commercial LLMs, directly leads to lower API costs (Nag et al., 2024).

Third, due to globalization and bilingual environments (including creoles), there has been an influx of English loanwords, code-switching, and online data in the Latin script, thus enabling romanization to make most of this overlap (Mukherjee et al., 2019). This phenomenon extends to non-English environments too, such as Arabizi and French (van der Wees et al., 2016). Lastly, the widespread

availability of general-purpose romanization tools, compared to the scarcity of high-quality, language-specific transliterators also makes this a practical choice (Purkayastha et al., 2023; Soni and Bhattacharyya, 2024).

4.6 LLMs Revisited: Are Transliterations Still Necessary?

Despite a scarcity of work applying transliteration for decoder-only models, recent studies suggest this mechanism may already be implicitly present in LLMs.

Specifically, Saji et al. (2025) identify a phenomenon they term "Latent Romanization", where intermediate layers of English-centric models tend to represent non-Latin tokens in Latin script before resolving into the native script. These phonetic approximations typically emerge in the middle-to-top layers, acting as a bridge that connects the model's language-agnostic concept space to language-specific output embeddings. More experiments further confirm that LLMs encode semantic concepts identically regardless of whether the input is in the native or romanized script. Moreover, when generating output directly in the Latin script, the intended representations emerge significantly earlier in the model's layers compared to when using native scripts. These findings suggest that LLMs may have internalized this process of transliteration (with Latin script). This observation, while nascent, is interesting as it gives a new perspective to the underlying working of modern LLMs in handling different scripts.

5 Conclusion

Transliteration is an efficient yet nuanced technique in multilingual models to overcome the "script barrier". It is particularly effective for low-resource languages written in different scripts but closely related to higher-resource ones. The dominance of romanization is driven by practical advantages such as reduced token fertility, cost efficiency, and the need to accommodate code-mixed globalized inputs. Yet, transliteration is not a universal solution. It can introduce ambiguity or harm performance by disrupting learned connections across scripts and add an additional bottleneck of post-hoc back-transliteration. Nevertheless, the fundamental utility of this mechanism is underscored by recent findings in emerging LLM paradigms with in-context learning, reversible transliteration, exploiting romanization, and latent romanization, though this landscape remains nascent and under-explored. As multilingual models evolve, transliteration can be a reliable tool if used properly until the script barrier is fundamentally addressed.

6 Limitations

While this survey provides a comprehensive overview of the application of transliteration in language models, it is subject to several limitations inherent in the current body of research.

First, the scope of our analysis is constrained by the tasks investigated in the literature. Much of the work on transliteration integration has concentrated on a narrow set of downstream tasks, primarily NER, POS tagging, and dependency parsing for encoder models and NMT for encoder-decoder models. The applicability of these findings to a wider array of NLP tasks remains less explored.

Second, this survey, reflecting the available research, primarily discusses encoder-only and encoder-decoder architectures. The impact and optimal application of these transliteration techniques on the now-prevalent decoder-only LLMs are not deeply covered in the literature and thus represent a significant limitation in our current understanding.

Finally, there is an interpretability gap in understanding how transliteration works. Very few works have tried to explain these internal mechanisms (Liu et al., 2024b; Saji et al., 2025). Consequently, this lack of insight limits our ability to predict when transliterations aid in downstream performance, highlighting the need for more research into how such models actually work.

References

Mansour Al Ghanim, Saleh Almohaimeed, Mengxin Zheng, Yan Solihin, and Qian Lou. 2024. Jailbreaking LLMs with Arabic transliteration and Arabizi. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18584–18600, Miami, Florida, USA. 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.

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological in-

flection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 984–996, Hong Kong, China. Association for Computational Linguistics.

Fares Aqlan, Xiaoping Fan, Abdullah Alqwbani, and Akram Al-Mansoub. 2019. Arabic–chinese neural machine translation: Romanized arabic as subword unit for arabic-sourced translation. *IEEE Access*, 7:133122–133135.

Aditya Bhargava and Grzegorz Kondrak. 2011. How do you pronounce your name? improving G2P with transliterations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 399–408, Portland, Oregon, USA. Association for Computational Linguistics.

Eleftheria Briakou and Marine Carpuat. 2019. The University of Maryland's Kazakh-English neural machine translation system at WMT19. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 134–140, Florence, Italy. Association for Computational Linguistics.

Andreas Chari, Iadh Ounis, and Sean MacAvaney. 2025. Lost in transliteration: Bridging the script gap in neural ir. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2900–2905.

Ethan C. Chau and Noah A. Smith. 2021. Specializing multilingual language models: An empirical study. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 51–61, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034, Online. Association for Computational Linguistics.

Raj Dabre, Anoop Kunchukuttan, Atsushi Fujita, and Eiichiro Sumita. 2018. NICT's participation in WAT 2018: Approaches using multilingualism and recurrently stacked layers. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation: 5th Workshop on Asian Translation: 5th Workshop on Asian Translation*, Hong Kong. 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.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in Bengali. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 286–296, Online only. 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.

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.

Jinhua Du and Andy Way. 2017. Pinyin as subword unit for chinese-sourced neural machine translation.

Nadir Durrani and Philipp Koehn. 2014. Improving machine translation via triangulation and transliteration. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation*, pages 71–78, Dubrovnik, Croatia. European Association for Machine Translation.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014. Integrating an unsupervised transliteration model into statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 148–153, Gothenburg, Sweden. Association for Computational Linguistics.

Mozhdeh Gheini and Jonathan May. 2019. A universal parent model for low-resource neural machine translation transfer. *arXiv preprint arXiv:1909.06516*.

Vikrant Goyal, Sourav Kumar, and Dipti Misra Sharma. 2020. Efficient neural machine translation for low-resource languages via exploiting related languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 162–168, Online. Association for Computational Linguistics.

Imane Guellil, Ahsan Adeel, Faical Azouaou, Fodil Benali, Ala-eddine Hachani, and Amir Hussain. 2018. Arabizi sentiment analysis based on transliteration and automatic corpus annotation. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity,*

Sentiment and Social Media Analysis, pages 335–341, Brussels, Belgium. Association for Computational Linguistics.

Junqing He, Long Wu, Xuemin Zhao, and Yonghong Yan. 2017. HCCL at SemEval-2017 task 2: Combining multilingual word embeddings and transliteration model for semantic similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 220–225, Vancouver, Canada. 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.

Andrew Johnson, Penny Karanasou, Judith Gaspers, and Dietrich Klakow. 2019. Cross-lingual transfer learning for Japanese named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 182–189, Minneapolis, Minnesota. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Haeji Jung, Jinju Kim, Kyungjin Kim, Youjeong Roh, and David R. Mortensen. 2026. Happiness is sharing a vocabulary: A study of transliteration methods. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7797–7816, Rabat, Morocco. Association for Computational Linguistics.

Mehdi M. Kashani, Eric Joanis, Roland Kuhn, George Foster, and Fred Popowich. 2007. Integration of an Arabic transliteration module into a statistical machine translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 17–24, Prague, Czech Republic. Association for Computational Linguistics.

Aleksandr Khakhmovich, Svetlana Pavlova, Kira Kirillova, Nikolay Arefyev, and Ekaterina Savilova. 2020. Cross-lingual named entity list search via transliteration. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4247–4255, Marseille, France. European Language Resources Association.

Jyotsana Khatri, Nikhil Saini, and Pushpak Bhattacharyya. 2021. Language relatedness and lexical closeness can help improve multilingual NMT: IITBombay@MultiIndicNMT WAT2021. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*, pages 217–223, Online. 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.

Youngsam Kim and Hyopil Shin. 2013. Romanization-based approach to morphological analysis in Korean SMS text processing. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 145–152, Nagoya, Japan. Asian Federation of Natural Language Processing.

Sai Koneru, Danni Liu, and Jan Niehues. 2021. Unsupervised machine translation on Dravidian languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 55–64, Kyiv. Association for Computational Linguistics.

Sahinur Rahman Laskar, Pankaj Dadure, Riyanka Manna, Partha Pakray, and Sivaji Bandyopadhyay. 2022. English to Bengali multimodal neural machine translation using transliteration-based phrase pairs augmentation. In *Proceedings of the 9th Workshop on Asian Translation*, pages 111–116, Gyeongju, Republic of Korea. International Conference on Computational Linguistics.

Jaeseong Lee, Dohyeon Lee, and Seung-won Hwang. 2024. ScriptMix: Mixing scripts for low-resource language parsing. 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 6430–6444, Mexico City, Mexico. Association for Computational Linguistics.

Jindřich Libovický, Jindřich Helcl, and David Mareček. 2018. Input combination strategies for multi-source transformer decoder. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 253–260, Brussels, Belgium. Association for Computational Linguistics.

Ying Lin, Xiaoman Pan, Aliya Deri, Heng Ji, and Kevin Knight. 2016. Leveraging entity linking and related language projection to improve name transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 1–10, Berlin, Germany. Association for Computational Linguistics.

Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. 2019. Robust neural machine translation with joint textual and phonetic embedding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3044–3049, Florence, Italy. Association for Computational Linguistics.

Yihong Liu, Chunlan Ma, Haotian Ye, and Hinrich Schuetze. 2024a. 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. 2025. 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 Imani, Orgest Xhelili, Haotian Ye, Chunlan Ma, François Yvon, and Hinrich Schütze. 2024b. How transliterations improve crosslingual alignment. *arXiv preprint arXiv:2409.17326*.

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.

Kurt Micallef, Fadhl Eryani, Nizar Habash, Houda Bouamor, and Claudia Borg. 2023. Exploring the impact of transliteration on NLP performance: Treating Maltese as an Arabic dialect. In *Proceedings of the Workshop on Computation and Written Language (CAWL 2023)*, pages 22–32, Toronto, Canada. Association for Computational Linguistics.

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.

Khalil Mrini and Francis Bond. 2018. Putting figures on influences on Moroccan Darija from Arabic, French and Spanish using the WordNet. In *Proceedings of the 9th Global Wordnet Conference*, pages 372–377, Nanyang Technological University (NTU), Singapore. Global Wordnet Association.

Siddhartha Mukherjee, Vinuthkumar Prasan, Anish Nediyanchath, Manan Shah, and Nikhil Kumar. 2019. Robust deep learning based sentiment classification of code-mixed text. In *Proceedings of the 16th International Conference on Natural Language Processing*, pages 124–129, International Institute of Information Technology, Hyderabad, India. NLP Association of India.

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.

Nikitha Murikinati, Antonios Anastasopoulos, and Graham Neubig. 2020. Transliteration for cross-lingual morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–197, Online. Association for Computational Linguistics.

Arijit Nag, Animesh Mukherjee, Niloy Ganguly, and Soumen Chakrabarti. 2024. Cost-performance optimization for processing low-resource language tasks using commercial LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15681–15701, Miami, Florida, USA. Association for Computational Linguistics.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1367, Singapore. Association for Computational Linguistics.

Vasanth Palanikumar, Sean Benhur, Adeep Hande, and Bharathi Raja Chakravarthi. 2022. DE-ABUSE@TamilNLP-ACL 2022: Transliteration as data augmentation for abuse detection in Tamil. In *Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages*, pages 33–38, Dublin, Ireland. 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.

Ratnavel Rajalakshmi, Saptharishee M, Hareesh S, Gabriel R, and Varsini Sr. 2024. DLRG-DravidianLangTech@EACL2024 : Combating hate speech in Telugu code-mixed text on social media. In *Proceedings of the Fourth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 140–145, St. Julian's, Malta. Association for Computational Linguistics.

Shruti Rijhwani, Jiateng Xie, Graham Neubig, and Jaime Carbonell. 2019. Zero-shot neural transfer for cross-lingual entity linking. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Rajarshi Roychoudhury, Subhrajit Dey, Md Akhtar,

Amitava Das, and Sudip Naskar. 2022. A novel approach towards cross lingual sentiment analysis using transliteration and character embedding. In *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, pages 260–268, New Delhi, India. Association for Computational Linguistics.

Kugatsu Sadamitsu, Itsumi Saito, Taichi Katayama, Hisako Asano, and Yoshihiro Matsuo. 2016. Name translation based on fine-grained named entity recognition in a single language. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 613–619, Portorož, Slovenia. European Language Resources Association (ELRA).

Siva Sai and Yashvardhan Sharma. 2021. Towards offensive language identification for Dravidian languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 18–27, Kyiv. Association for Computational Linguistics.

Alan Saji, Jaavid Aktar Husain, Thanmay Jayakumar, Raj Dabre, Anoop Kunchukuttan, and Ratish Puduppully. 2025. RomanLens: The role of latent Romanization in multilinguality in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26410–26429, Vienna, Austria. Association for Computational Linguistics.

Ali Salehi and Cassandra L. Jacobs. 2024. The effect of model capacity and script diversity on subword tokenization for Sorani Kurdish. In *Proceedings of the 21st SIGMORPHON workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 51–56, Mexico City, Mexico. Association for Computational Linguistics.

Nasredine Semmar and Houda Saadane. 2013. Using transliteration of proper names from Arabic to Latin script to improve English-Arabic word alignment. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1022–1026, Nagoya, Japan. Asian Federation of Natural Language Processing.

Rajvee Sheth, Samridhi Raj Sinha, Mahavir Patil, Himanshu Beniwal, and Mayank Singh. 2026. Beyond monolingual assumptions: A survey of code-switched nlp in the era of large language models across modalities.

Haiyue Song, Raj Dabre, Zhuoyuan Mao, Fei Cheng, Sadao Kurohashi, and Eiichiro Sumita. 2020. Pre-training via leveraging assisting languages for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 279–285, Online. Association for Computational Linguistics.

Govind Soni and Pushpak Bhattacharyya. 2024. RoMantra: Optimizing neural machine translation for low-resource languages through Romanization. In *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, pages 157–168, AU-KBC Research Centre, Chennai, India. NLP Association of India (NLPAI).

Simeng Sun, Angela Fan, James Cross, Vishrav Chaudhary, Chau Tran, Philipp Koehn, and Francisco Guzmán. 2022. Alternative input signals ease transfer in multilingual machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5291–5305, Dublin, Ireland. Association for Computational Linguistics.

Nafisa Tabassum, Mosabbir Khan, Shawly Ahsan, Jawad Hossain, and Mohammed Moshiul Hoque. 2024. Sandalphon@DravidianLangTech-EACL2024: Hate and offensive language detection in Telugu code-mixed text using transliteration-augmentation. In *Proceedings of the Fourth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 167–172, St. Julian's, Malta. Association for Computational Linguistics.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2016. A simple but effective approach to improve Arabizi-to-English statistical machine translation. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 43–50, Osaka, Japan. The COLING 2016 Organizing Committee.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong Kong, China. Association for Computational Linguistics.

Andrea W Wen-Yi and David Mimno. 2023. Hyperpolyglot LLMs: Cross-lingual interpretability in token embeddings. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1124–1131, Singapore. Association for Computational Linguistics.

Genta Winata, Alham Fikri Aji, Zheng Xin Yong, and Thamar Solorio. 2023. The decades progress on code-switching research in NLP: A systematic survey on trends and challenges. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2936–2978, Toronto, Canada. Association for Computational Linguistics.

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.

Shijia Zhou, Huangyan Shan, Barbara Plank, and Robert Litschko. 2024. MaiNLP at SemEval-2024 task 1: Analyzing source language selection in cross-lingual

textual relatedness. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1842–1853, Mexico City, Mexico. Association for Computational Linguistics.

Wenhao Zhuang, Yuan Sun, and Xiaobing Zhao. 2025. Enhancing cross-lingual transfer through reversible transliteration: A Huffman-based approach for low-resource languages. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16299–16313, Vienna, Austria. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.