Alternative Input Signals Ease Transfer in Multilingual Machine Translation

Simeng Sun[1*] **Angela Fan**[2] **James Cross**[2]
Vishrav Chaudhary[3†] **Chau Tran**[2] **Philipp Koehn**[2] **Francisco Guzmán**[2]
University of Massachusetts Amherst[1] Meta AI[2] Microsoft Turing[3]
simengsun@umass.edu
{angelafan,jcross,chau,pkoehn,fguzman}@fb.com
vchaudhary@microsoft.com

Abstract

Recent work in multilingual machine translation (MMT) has focused on the potential of positive transfer between languages, particularly cases where higher-resourced languages can benefit lower-resourced ones. While training an MMT model, the supervision signals learned from one language pair can be *transferred* to the other via the tokens shared by multiple source languages. However, the transfer is inhibited when the token overlap among source languages is small, which manifests naturally when languages use different writing systems. In this paper, we tackle inhibited transfer by augmenting the training data with alternative signals that unify different writing systems, such as phonetic, romanized, and transliterated input. We test these signals on Indic and Turkic languages, two language families where the writing systems differ but languages still share common features. Our results indicate that a straightforward *multi-source self-ensemble* — training a model on a mixture of various signals and ensembling the outputs of the same model fed with different signals during inference — outperforms strong ensemble baselines by 1.3 BLEU on both language families. Further, we find that incorporating alternative inputs via self-ensemble can be particularly effective in low-resource settings, leading to +5 BLEU when only 5% of the total training data is accessible. Finally, our analysis demonstrates that including alternative signals yields more consistency and translates named entities more accurately, which is crucial for increased factuality of automated systems.

1 Introduction

Machine translation has seen great progress, with improvements in quality and successful commercial applications. However, the majority of this improvement benefits languages with large quantities of high-quality training data (high-resource languages). Recently, researchers have focused on the development of multilingual translation models (Aharoni et al., 2019; Fan et al., 2020) capable of translating between many different language pairs rather than specialized models for each translation direction. In particular, such multilingual models hold great promise for improving translation quality for *low-resource* languages, as grouping languages together allows them to benefit from linguistic similarities as well as shared data between related languages. For example, training a translation system with combined Assamese and Bengali data would enable transfer learning between the two languages.

We investigate how to enable multilingual translation models to optimally learn these similarities between languages and leverage this similarity to improve translation quality. The fundamental unit representing lingual similarity is the token — languages that are similar often have similar words or phrases — and during training, translation models can learn strong representations of tokens in low-resource languages if they are also present in high-resource languages. However, a challenge arises when similar languages share only a small amount of tokens, which inhibits the transfer to limited and trivial cases of token sharing, e.g., punctuation marks and digits. This is particularly clear in cases where similar languages are written in different scripts, as the amount of shared tokens is small compared to languages using the same written script. An example would be Hindi and Gujarati, which have phonetic similarity but are written in their own native scripts.

To tackle inhibited transfer due to distinct writing systems, we transform the original input via *transliteration*, the process of converting text from one script to another, to get alternative signal from the original source sentences. Transliteration has

[*] Work done during an internship at Meta AI.
[†] Work done while at Meta AI.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 5291 - 5305

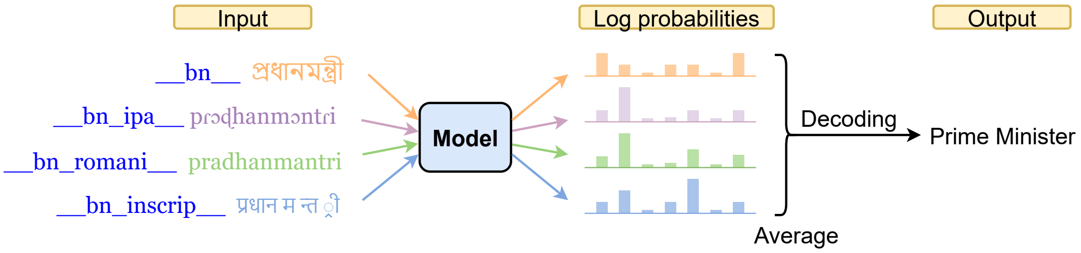

Figure 1: A generic illustration of self-ensemble for a multilingual translation system while translating Bengali to English. The input contains different signals, each preceded by a special language token ('__bn__' indicates input in original Bengali script, '__bn_ipa__' the phonetic version of the same Bengli input, '__bn_romani__' the romanized version and '__bn_inscrip__' the same input but written in the script of Hindi, a language within the same language family). The log probabilities output by the model given each type of input are averaged for subsequent decoding process.

been used in many real world cases, such as converting Cyrillic Serbian to Latin Serbian, as the language is commonly written with both scripts, or typing in romanized Hindi for convenience on a Latin-script keyboard. To unify various writing scripts to increase token overlap, we experiment with three types of transliteration: **(1)** transliterate into phonemes expressed by international phonetic alphabet (IPA), **(2)** transliterate into Latin script (ROMANI), and **(3)** transliterate into a script used by another language within the same language family (INSCRIP). Beyond training on alternative inputs created through transliteration, we also systematically examine approaches to combining different signals. Our experimental results on Indic and Turkic datasets demonstrate that **(i)** a *self-ensemble* (Figure 1) – training a model on the mixture of different signals and using an ensemble of the *same* model given different input signals during inference time, outperforms other methods such as multi-source ensemble and multi-encoder architecture, which require training multiple models or significant architectural changes. **(ii)** Further, without the need for additional bitext, a self-ensemble over the original and transliterated input consistently outperforms baselines, and is particularly effective when the training set is small (e.g. low-resource languages) with improvements of up to +5 BLEU. **(iii)** Finally, the improvements in BLEU originate from clear gain in the accuracy and consistency in the translation of named entities, which has strong implications for increased factuality of automated translation systems.

2 Method

Multilingual translation models enable languages to learn from each other, meaning low-resource languages can benefit from similarities to high-resource languages where data is plentiful. However, surface-level differences between languages, such as writing system, can obscure semantic similarities. We describe an approach to transliterating input sentences to various alternative forms that maximize transfer learning between different languages, and various modeling approaches to incorporating such varied inputs.

2.1 Alternative Inputs Bridge the Gap between Surface Form and Meaning

While training a multilingual translation system, tokens shared by multiple source languages serve as anchors to transfer information obtained from learning one language pair to the other. For example, the translation of 'terisini' in low-resourced Uzbek data can benefit from the word 'derisinin' in relatively high-resourced Turkish data after tokenizing into sub-word units. However, the transfer is hindered when the amount of shared tokens is small — exacerbated by cases where the source and target languages are written in different scripts.[1] To alleviate the issue of various writing systems and encourage languages to transfer, we focus on alternative signals that unify the script of source languages and have larger token overlap. The core concept we explore is how to best leverage *transliteration*, or the process of converting the text from one script to the other. We demonstrate that transliteration can be an effective data augmentation approach that improves the translation performance *without the need of acquiring additional parallel data.* We explore three alternative inputs that allow

[1] Muller et al. (2021) show that the discrepancy in scripts causes failure to transfer in multilingual models and further hurts performance in downstream tasks.

models to share information more easily across languages with low token overlap but high semantic similarity. Figure 4 in Appendix C shows example alternative signals of the same Oriya sentence.

Phonetic Input. Related languages in the same language family usually sound similar, such as languages in the Romance language family and those in the Indo-Aryan language family. Although cognates can be captured to some degree for Romance languages on subword-level, it is difficult for the Indo-Aryan family as those languages use different writing systems. Therefore, to fully exploit shared information, we transform the original textual input (BASE) into the phonetic space, where the basic units are phonemes expressed in international phonetic alphabet (IPA). For example, 'প্‌রধানমন্‌ত্‌রী' in Bengali looks like 'pɾɔɖhanmɔntɾi' in IPA form.

Romanized Input. Many languages use Latin alphabet (or Roman alphabet) in their default writing system, if not, they more or less have romanization of their default script in order to accommodate conventional keyboards, e.g., Chinese can be typed on U.S. keyboards through Pinyin, the romanization of Chinese. To utilize this existing form of alternative input, the romanized input is another signal we explore in this work. For example, 'প্‌রধানমন্‌ত্‌রী' looks like 'pradhanmantri' in romanized form.

In-family Script Input. The two previous alternative representations introduce tokens not present in the existing vocabulary, which increases the number of input and output representations the translation models must learn. Further, phonetic input is artificial in the sense that it is not used by people to communicate to each other in written form — and only used for pronunciation. Romanization naturally would introduce many additional tokens if the source language does not use Latin script. A third alternative that does not suffer these drawbacks is transliterate source language into the script of any of the other source languages in the multilingual translation model. To take advantage of language relatedness (Dhamecha et al., 2021), we unify the source languages with the script used by a language within the same language family (INSCRIP). This method has the additional advantage of not needing to learn new subword tokenization models or replace the old vocabulary with a new one since all the inputs are expressed in one of the existing multilingual model's source language scripts. For example, 'প্‌রধানমন্‌ত্‌রী' looks like 'प्‌रधान म न्‌त्‌री' when transliterated into Hindi script.

Advantages of Transliterated Inputs. Various different input representations have been inserted into translation models, from parse trees (Li et al., 2017; Currey and Heafield, 2018) to pretrained embeddings (Artetxe et al., 2018; Conneau et al., 2018). Compared to these alternatives, transliteration has several clear advantages. Most importantly, transliteration is fast and accurate. Several existing alternatives often use other models to produce a different input, such as a parse tree, which cascades error from the first model into the translation model. Comparatively, the alphabet alignment between various writing systems is quite well known, even for many low-resource languages, as alphabet is one of the foundational aspects of studying any new language. Similarly, phonetic pronunciation guides are often widely available. These resources are also easily accessible programmatically, making them ideal for converting large quantities of supervised training data, for instance, the `espkea-ng` tool supports phonemization of more than 100 languages and accents. Beyond the ease of creating transliterations, we emphasize that this technique does not require any data annotation or collection of parallel data. Thus, it can be utilized in any existing translation system.

2.2 Adding Transliterated Input Combinations to Translation Models

How can additional transliterated inputs be incorporated into modern machine translation architectures? Since each alternative signal could capture a different view of the original input, in addition to training on each of the individual alternative signal alone, we investigate different approaches to combining them.

Straight Concatenation The simplest combination strategy is to concatenate different input signals and separate them by a special token. For instance, to combine the original and phonetic input, we re-arrange the input to be of the format: "*[original input]* `[SEP]` *[phonetic input]*". During training, the decoder explicitly attends to tokens in both input signals. The advantage of this method is that no architectural change is required as all modification is operated on the input data. However, as the concatenated input becomes longer, this method requires more computation to train compared to the baseline model trained on the original input only.

Multi-Encoder Architectures Prior works have found multi-encoder architecture to be effective for multi-source machine translation (Nishimura et al., 2018). To cope with input from different sources, each encoder in the multi-encoder architecture deals with one type of input. To attend to multiple encoders on the decoder side, four cross-attention mechanisms can be adopted. We direct the reader to Appendix A for a detailed description of these attention variations. Although prior work investigates the efficacy of this approach, it is a complicated model choice requiring non-trivial architectural changes.

Multi-Source Ensemble Ensembles are usually employed to boost the performance of a translation system. In a standard setup, each ensemble component is trained with identical configuration except for the random seed. We generalize this method to multi-source ensemble, i.e., individual ensemble components are trained on different transliterated inputs. During inference time, each component is fed with the type of transliteration it was trained on and produces the predicted log probabilities, which are averaged over all components for the subsequent decoding process. It is important for models trained on different source signals to have the same target vocabulary so that the average of log probabilities can happen. Unlike the previous two methods, this approach requires training multiple full models, thus requiring even more computation.

Multi-Source Self-Ensemble Ensembling models that are trained on different input transliterations has the advantage that each individual model is maximally *simple* — only the input data for training changes. However, it comes with the downside that multiple different models need to be trained. This creates challenges particularly when models grow in size, as a new model would need to be created for each different transliterated input.

Instead, we propose the *Multi-Source Self-Ensemble*, which has all the advantages of traditional ensembling, but only requires *one* model to be trained. Previous works in self-ensembles have focused on model robustness (Liu et al., 2018), which is distinct from varying input representations. Other work creates inputs in different languages (Fan et al., 2020), but have to use a translation model to create those inputs first.

In our case, we train the model with different transliterated inputs mapping to the same translated target sentence. Concretely, the model is trained on the mixture of various input signals, each preceded by a special language token indicating which type of signal this input belongs to. At inference time, the alternative transliterated signals of the same test sentence are fed to the same model and the log probabilities produced by these separate passes are averaged as in multi-source ensemble. This approach is simple to implement as it requires no architectural change, meaning the transliterated inputs we propose can be added seamlessly to any existing translation library. Unlike multi-source ensemble, only one model needs to be trained, stored and loaded for inference, greatly simplifying the ensembling process and increasing the scalability of our approach (particularly as translation models increase in size). To enforce fair comparison between multi-source self-ensemble and multi-source ensemble, we *scale* the former so that it has the same number of parameters as that of all ensemble components of the latter. For the purpose of minimally impacting inference speed, the scaling is done only to the encoder embedding dimension so that the decoder remains the same.

3 Experimental setup

Dataset We train our model on two language families: Indic and Turkic. The Indic dataset is from the WAT MultiIndic MT task[2], including 10 Indic languages and in total around 11 million Indic-English bi-texts. Six of the Indic languages are Indo-Aryan languages and the rest are Dravidian languages. All of these languages use a different writing system. The Turkic dataset is collected from the open parallel corpus (Tiedemann, 2012)[3]. For relatively high-resourced language Turkish, we randomly select 4 million subset from the CCAligned (El-Kishky et al., 2020) corpus. Within this dataset, two languages use Cyrillic alphabet (Kazakh and Kyrgyz) and the rest use Latin alphabet. Detailed dataset statistics are displayed in Table 7 in Appendix B.

Single-input model To test the effectiveness of each input signal, we train models on each *single* type of input: original input (BASE), phonetic input (IPA), romanized input (ROMANI) or input all expressed in the script of a language within the same language family (INSCRIP). On the Indic dataset,

[2] https://lotus.kuee.kyoto-u.ac.jp/WAT/indic-multilingual/index.html

[3] https://opus.nlpl.eu/

∼93 M parameters			∼2×93 M parameters		
	Indic	**Turkic**		**Indic**	**Turkic**
Single-input Original			**Standard Ensemble**		
BASE	33.6	20.3	BASE+BASE	34.5	21.1
Single-input Alternative			**Multi-Source Ensemble**		
IPA	32.7	17.9	BASE+IPA	34.3	20.9
ROMANI	32.5	20.7	BASE+ROMANI	34.4	21.4
INSCRIP	33.4	20.5	BASE+INSCRIP	34.5	21.5
Multi-Source Self-Ensemble			**Multi-Source Self-Ensemble**		
BASE+IPA	34.1	20.5	BASE+IPA	35.7	21.9
BASE+ROMANI	33.8	20.9	BASE+ROMANI	35.7	22.2
BASE+INSCRIP	**34.2**	**21.3**	BASE+INSCRIP	**35.8**	**22.4**

Table 1: BLEU scores on Indic test set and FloRes Turkic Devtest set.

for the INSCRIP signal, all Indo-Aryan languages are transliterated into Hindi script, and all Dravidian languages into Tamil script. On the Turkic dataset, all languages in Latin script are transliterated into Cyrillic script.

Multi-Source Ensemble A baseline for ensembling models trained on different signals is the standard ensemble (BASE+BASE) where two BASE models are ensembled, each trained with a different random seed. Although there are multiple combinations of input signals, we only discuss the cases where BASE is combined with one of {IPA, ROMANI, INSCRIP}, since in our preliminary experiments, we found dropping the BASE model leads to significantly degraded performance.

Multi-Source Self-Ensemble Similar to above, we train a single model on the mixture of original input and one of {IPA, ROMANI, INSCRIP} input for multi-source self-ensemble. To enforce fair comparisons with the ensembled models, which have more parameters in total, we train two sizes of the self-ensemble (SE) model, one having the same size of a single baseline model, the other scaled to have twice the number of parameters of a single BASE model.

Data Preprocessing We use `espeak-ng`[4] to convert the original input to phonetic input. For Indic languages, we use `indic-trans`[5] (Bhat et al., 2015) to obtain the romanized as well as the in-family transliterated input. On the Turkic dataset, we manually align the Cyrillic and Latin alphabet and substitute the letter(s) in one script with the corresponding one in another.[6] The Indic languages are tokenized with `indic_nlp_library` and the rest are tokenized with `mosesdecoder`[7]. We use `sentencepiece`[8] to create 32K BPE (Sennrich et al., 2016) subword vocabularies for each type of input signal. Examples longer than 250 tokens are discarded. We merge the source dictionaries of different signals by dropping duplicated tokens, while keeping the decoder dictionaries all the same in order to compute the average log probabilities in ensemble settings.

Training & Evaluation We train many-to-En language directions during training (10 and 5 directions for Indic and Turkic dataset respectively). The architecture is a standard 6-layer encoder 6-layer decoder Transformer model, with 512 embedding dimension and 2048 hidden dimension in the default setting. For the scaled self-ensemble model, we increase the encoder hidden dimension such that the number of parameters in this model approximately matches that of n baseline models ($n = 2$ for results in Table 1). We use 4000 warmup steps and learning rate 0.0003. Both the dropout and attention dropout rate are set to 0.2. Label smoothing is set to 0.1. Data from different language pairs are sampled with 1.5 temperature sampling. We

[4] https://github.com/espeak-ng/espeak-ng

[5] https://github.com/libindic/indic-trans

[6] The substitution process starts from the letter in the target script that corresponds to the most number of letters in the source script.

[7] https://github.com/moses-smt/mosesdecoder

[8] https://github.com/google/sentencepiece

	BASE	IPA	ROMANI	INSCRIP
Uni-gram	0.03	0.15	0.13	0.16
Sent. len	34.7	39.3	25.9	51.3

Table 2: Uni-gram token overlap and sentence length of various types of input on MultiIndic dev set.

train all models for 18 epochs and 40 epochs for Indic and Turkic dataset respectively and evaluate the best checkpoint selected by dev loss. We use `spBLEU`[9] (Goyal et al., 2021; Guzmán et al., 2019) to compute the BLEU scores.[10]

4 Results

In this section, we compare the performance of our proposed multi-source self-ensemble model to various alternative ways of input combinations on two low-resource language families: Indic and Turkic languages. Furthermore, we show multi-source self-ensemble learns faster and generates more consistent and accurate translations.

4.1 Performance of Multi-Source Self-Ensemble

Our method is based on the hypothesis that incorporating alternative inputs increases the token overlap of source languages, which benefits the transfer during training. To verify this, we compute average sentence-level uni-gram overlap of all source language pairs (Table 2) and find that alternative signals do have higher token overlap compared to the original input. For instance, the IPA signal, having similar average sentence length as BASE , has much higher token overlap (0.15 vs. 0.03).

Do increased token overlaps result in better translation performance? We train models on each of the alternative inputs alone and report the results in the left column of Table 1. We find that using only one alternative input in the source has either worse or similar performance as the original baseline, indicating higher token overlap among source languages does not guarantee better BLEU scores. The degraded performance is likely due to unfavorable interference introduced by shared tokens in the alternative signals. The interference may create information loss[11] or increased ambiguity[12], which reinforces the importance of combining alternative inputs with the original input.

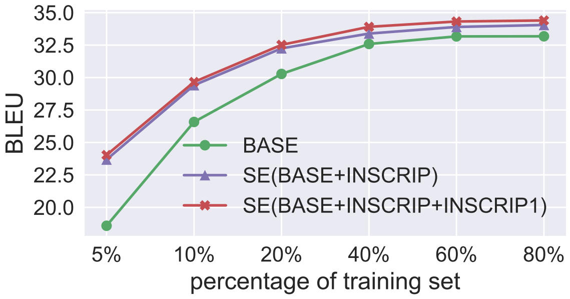

Figure 2: Learning curve of the baseline model BASE and the same-sized self-ensemble model trained on the original input as well as transliterated input. INSCRIP denotes the transliteration where the target script for Indo-Aryan and Dravidian languages are Hindi and Tamil respectively. The target scripts of INSCRIP1 are Oriya and Kannada respectively.

Due to undesired interference exhibited in the alternative input spaces, we therefore adopt the input combination using our proposed *Multi-Source Self-Ensemble* to combine the original input and alternative signals. Results in left lower part of Table 1 demonstrate improvements over the single-input baseline. Our best performing alternative input configuration improves +1.0 BLEU on Turkic languages and +0.6 BLEU on Indic languages for 93M parameter models.

In production, model ensembles are often employed to achieve the best possible performance. This is usually done by training multiple models each initialized with a different random seed (Bawden et al., 2020; Tran et al., 2021b), and averaging the predicted next token probabilities at inference time. We also provide results against these strong ensemble baselines and observe +1.3 BLEU improvements on both Indic and Turkic languages. Note that, to enforce a fair comparison, we compare a scaled version of the multi-source self-ensemble model which has the same number of parameters as multiple ensemble baseline components.

[9] https://github.com/facebookresearch/flores#spm-bleu

[10] While prior work (Kocmi et al., 2021) has shown better correlation between neural metrics and human ratings, there have not been extensive evaluations for low-resource languages, especially for systems dealing with various writing scripts. Therefore we use spBLEU which is consistent with previous works (Goyal et al., 2021).

[11] For example, the punctuation marks are lost during phonemization process.

[12] For instance, multiple words may have the same pronunciation and thus have the same input in IPA form, which makes the learning harder.

Configuration	BLEU
Single-input Baseline	
BASE	33.6
Straight Concatenation	
BASE+<SEP>+IPA	33.7
BASE+<SEP>+ROMANI	33.7
BASE+<SEP>+INSCRIP	33.6
Multi-Encoder Architectures	
Bi-Encoder	
BASE+BASE	34.2
BASE+IPA	33.9
BASE+ROMANI	33.9
BASE+INSCRIP	34.0
Quad-Encoder	
BASE+BASE+BASE+BASE	34.3
BASE+IPA+ROMANI+INSCRIP	34.1
Multi-source Self-ensemble	
BASE+INSCRIP	34.2

Table 3: Indic test set BLEU of models trained on straight concatenation of input as well as multi-encoder architectures. Training on the concatenated input does not impact the BLEU much. Multi-encoder architectures, although having a lot more number of parameters, for instance, quad-encoder, achieve similar performance of a much smaller multi-source self-ensemble.

	C-BLEU	NE-F1
Single-input Baseline		
BASE	34.7	55.9
Single-input Alternative Input		
IPA	33.8	54.7
ROMANI	33.0	54.5
INSCRIP	35.3	55.4
Multi-Source Self-Ensemble		
BASE+IPA	**36.2**	56.1
BASE+ROMANI	35.5	56.3
BASE+INSCRIP	**36.2**	**56.4**

Table 4: The consistency BLEU (**C-BLEU**) and exact named entity match F1 (**NE-F1**) of MultiIndic test set. Higher **C-BLEU** scores imply more consistent output in many-to-En setting. Higher **NE-F1** scores indicate better translation of named entities.

4.2 Advantages of Multi-Source Self-Ensemble

Architectural Simplicity. As introduced in §2.2, there are various ways to incorporate multiple inputs, such as concatenation to form a longer input or using multiple encoders networks. In Table 3, we show that using multiple encoders has no improvements over the comparable baseline with raw text input, and straight concatenation only brings marginal gains (+0.1 BLEU). Further, our simple but effective Multi-Source Self-Ensemble technique reaches the same performance as that of a much larger quad-encoder model, which requires non-trivial architectural changes and takes more compute to train. Thus, our technique is suitable to be used out of the box in any seq-to-seq library.

Faster Learning in Low-Resource Settings. To understand how self-ensemble performs with different amounts of data, we plot the learning curve of both the baseline and the self-ensemble model on 5%[13] to 80% of the total Indic training set.[14] As shown in Figure 2, the self-ensemble model outperforms the baseline model by a large margin when the amount of training data is small (**+5 BLEU** when only 5% of the total set is used for training). This is the scenario for most low-resource languages, as the gap gradually closes when more data is available. Overall, the multi-source self-ensemble model is consistently better than the baseline model irrespective of training data scale. This suggests that transliteration can be a cheap and effective data augmentation approach when used in conjunction with multi-source self-ensemble.

Improved Output Consistency. We conduct a deeper analysis to understand the performance improvement of Multi-Source Self-Ensembles beyond BLEU scores alone. We find that our proposed technique generates much more consistent output, which could be a benefit of alternative signals transferring information more easily amongst source languages. We propose consistency BLEU (**C-BLEU**) to quantify the consistency of multi-way evaluation output of a many-to-En translation model. We treat the output of L_1-En direction as reference and output of all other L_i-En directions as hypothesis. We compute this for all N source languages in the dataset, accounting for total $N(N-1)$ C-BLEU scores, then take the average of all (Table 4). While training on IPA or ROMANI alone does not outperform the baseline in terms of C-BLEU, model trained on INSCRIP input improves the score by +1.3. Self-ensemble over BASE and IPA increases the C-BLEU to **36.2** (and from **36.3** to **38.1** with scaled model), indicating the alternative signals are best trained together with the original input.

[13] When the training set is very small (5% and 10%), we train for 60 epochs and select the model by dev loss.

[14] The transliterated input are those of the same subset of training data, thus no sentences having new semantic meaning are added in the multi-source self-ensemble setup.

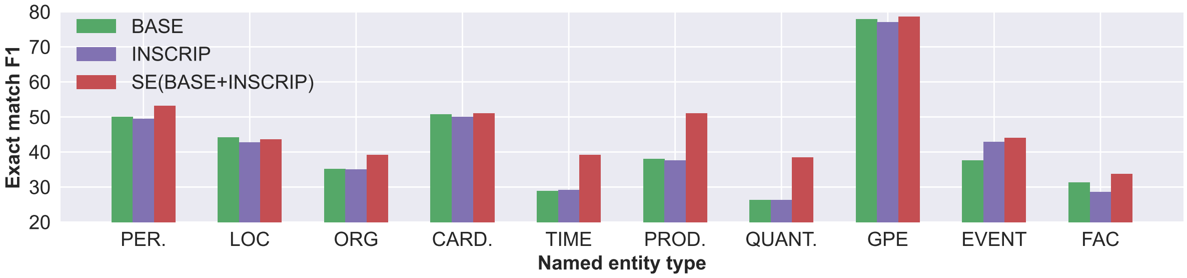

Figure 3: The exact named entity match F1 score of BASE INSCRIP and same-sized self-ensemble model trained on the previous two inputs (SE(BASE+INSCRIP)). Although the results of the self-ensemble model only slightly outperforms the baseline (55.9 vs. 56.4), the gains are more obvious when breaking the results by entity type.

Improved Named Entity Accuracy. The previous analysis implies the self-ensemble model outputs more consistent translation, yet this does not mean the consistent translations are accurate. In this section, we conduct an analysis targeted at named entities. We use spaCy (Honnibal et al., 2020) NER tagger to extract all named entities, and then compute the exact match of the extracted entities. According to the results in Table 4, self-ensemble introduces small gains (+0.5) in terms of named entity F1 (NE-F1), whereas the scaled self-ensemble boosts NE-F1 score by **+1.1**. Although the improvement is small in aggregate, we find significant improvement when breaking down by entity type. As shown in Figure 3, the multi-source self-ensemble model (without scaling) outperforms the baseline model on certain entity types, e.g., person, organization, time and event by a large margin.

5 Related work

5.1 Alternative Input for Multilingual MT

Our work can be viewed as multilingual MT (Firat et al., 2016) combined with multi-source MT (Zoph and Knight, 2016), where the sources are not other languages but rather alternative transliterated signals. The transliterated input has been explored in the past for translation system. Nakov and Ng (2009) use transliteration as a preprocessing step for their phrase-based SMT model to tackle systematic spelling variation. Both Chakravarthi et al. (2019) and Koneru et al. (2021) convert Dravidian languages to Latin script and train multilingual models with *both source and target* in Latin script; the latter identify code-switching to be a challenge during back-transliteration. Besides converting to Latin script, Dabre et al. (2018) use another common script, Devanagari, for Indic languages. In addition to the natural written scripts, previous works also explored artificial script, such as IPA. Liu et al. (2019) incorporate phonetic representations, specifically for Chinese Pinyin, to cope with homophone noise. Unlike our work, Chakravarthi et al. (2019) adopt transliteration to IPA for both the source and target. Apart from transliterated input, other potential alternative signals we did not fully explored include orthographic syllable units (Kunchukuttan and Bhattacharyya, 2016, 2020), morpheme-based units (Ataman et al., 2017; Dhar et al., 2020), and character (Lee et al., 2017) or byte (Wang et al., 2019a) level input in addition to the subword-level units (Sennrich et al., 2016).

5.2 Input signal combination

Multi-encoder architecture is the most common way to combine input from different sources. While previous works mainly use additional encoders to encode syntactic information (Li et al., 2017; Currey and Heafield, 2018) or input in another language (Nishimura et al., 2018), we feed in each encoder with different signals of the same sentence. Prior works also investigated approaches to combining input at different granularity (Ling et al., 2015; Chen et al., 2018; Casas et al., 2020). Wang et al. (2019b) combine the decoupled lexical and semantic representations through an attention mechanism. Another common method of utilizing additional input signal is multi-task learning, force the model to output extra labels (Luong et al., 2016; Grönroos et al., 2017). Apart from combining the sources during training, inference-time ensemble (Garmash and Monz, 2016) is often adopted by recent submissions to shared MT tasks (Ng et al., 2019; Tran et al., 2021a). The ensemble components are usually separate systems trained with different random initialization or language pairs. Fan et al. (2020) ensemble the same model by feeding in source sentences in different languages. The self-ensemble approach was also found to make networks more robust after adding random noises (Liu et al., 2018). Prior work also uses the term "self-ensemble" to refer to an ensemble of models using weights from different time steps during training (Xu et al., 2020).

6 Conclusion

To overcome the low token-overlap issue exhibited in multilingual MT systems due to distinct writing system, we examined three alternative signals (phonetic, romanized and in-family transliterated input) and investigated four approaches (input concatenation, multi-encoder, multi-source ensemble, self-ensemble) to combining them with the original input. Our results show that training a single model with a mixture of diverse signals and performing self-ensemble during inference time can improve BLEU by 1.3 points on Indic and Turkic dataset. The improvements can reach +5 BLEU when training data size is small. Further, we show this approach generate more accurate and consistent translation of named entities which greatly impacts the factuality accuracy of news translation.

Acknowledgement

We thank Shiyue Zhang, Xiang Zhou, Jean Maillard, Yixiao Song, and Marzena Karpinska for the helpful discussions during the course of this work.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation.

Duygu Ataman, Matteo Negri, Marco Turchi, and Marcello Federico. 2017. Linguistically motivated vocabulary reduction for neural machine translation from turkish to english.

Rachel Bawden, Alexandra Birch, Radina Dobreva, Arturo Oncevay, Antonio Valerio Miceli Barone, and Philip Williams. 2020. The University of Edinburgh's English-Tamil and English-Inuktitut submissions to the WMT20 news translation task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 92–99, Online. Association for Computational Linguistics.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

Noe Casas, Marta R. Costa-jussà, and José A. R. Fonollosa. 2020. Combining subword representations into word-level representations in the transformer architecture. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 66–71, Online. Association for Computational Linguistics.

Bharathi Raja Chakravarthi, Mihael Arcan, and John P. McCrae. 2019. Comparison of Different Orthographies for Machine Translation of Under-Resourced Dravidian Languages. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 6:1–6:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Huadong Chen, Shujian Huang, David Chiang, Xinyu Dai, and Jiajun Chen. 2018. Combining character and word information in neural machine translation using a multi-level attention. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1284–1293, New Orleans, Louisiana. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data.

Anna Currey and Kenneth Heafield. 2018. Multi-source syntactic neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2961–2966, Brussels, Belgium. 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.

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.

Prajit Dhar, Arianna Bisazza, and Gertjan van Noord. 2020. Linguistically motivated subwords for English-Tamil translation: University of Groningen's submission to WMT-2020. In *Proceedings of the Fifth Conference on Machine Translation*, pages 126–133, Online. Association for Computational Linguistics.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzmán, and Philipp Koehn. 2020. CCAligned: A massive collection of cross-lingual web-document pairs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 5960–5969, Online. Association for Computational Linguistics.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Ekaterina Garmash and Christof Monz. 2016. Ensemble learning for multi-source neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1409–1418, Osaka, Japan. The COLING 2016 Organizing Committee.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2021. The flores-101 evaluation benchmark for low-resource and multilingual machine translation.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2017. Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis. In *Proceedings of the Second Conference on Machine Translation*, pages 296–302, Copenhagen, Denmark. Association for Computational Linguistics.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. Two new evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Tom Kocmi, Christian Federmann, Roman Grundkiewicz, Marcin Junczys-Dowmunt, Hitokazu Matsushita, and Arul Menezes. 2021. To ship or not to ship: An extensive evaluation of automatic metrics for machine translation. In *Proceedings of the Sixth Conference on Machine Translation*, pages 478–494, Online. Association for Computational Linguistics.

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.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2016. Orthographic syllable as basic unit for SMT between related languages. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1912–1917, Austin, Texas. Association for Computational Linguistics.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2020. Utilizing language relatedness to improve machine translation: A case study on languages of the indian subcontinent.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697, Vancouver, Canada. 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.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation.

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.

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the*

European Conference on Computer Vision (ECCV), pages 369–385.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning.

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.

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.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission.

Yuta Nishimura, Katsuhito Sudoh, Graham Neubig, and Satoshi Nakamura. 2018. Multi-source neural machine translation with missing data. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 92–99, Melbourne, Australia. Association for Computational Linguistics.

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

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *LREC*.

Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. 2021a. Facebook ai wmt21 news translation task submission.

Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. 2021b. Facebook AI's WMT21 news translation task submission. In *Proceedings of the Sixth Conference on Machine Translation*, pages 205–215, Online. Association for Computational Linguistics.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2019a. Neural machine translation with byte-level subwords. *arXiv preprint arXiv:1909.03341*.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019b. Multilingual neural machine translation with soft decoupled encoding. In *International Conference on Learning Representations*.

Yige Xu, Xipeng Qiu, L. Zhou, and Xuanjing Huang. 2020. Improving bert fine-tuning via self-ensemble and self-distillation. *ArXiv*, abs/2002.10345.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California. Association for Computational Linguistics.

A Multi-encoder architecture

As has been systematically explored by Libovický et al. (2018), there are four kinds of multi-encoder cross-attention that can be applied on the decoder side: (1) Serial: cross-attention to each encoder is performed layer by layer. (2) Parallel: cross-attention to each encoder is performed in parallel and then the outputs are added together before feeding to the feed-forward layer. (3) Flat: outputs of all encoders are concatenated along the length dimension as the input to a single cross-attention. (4) Hierarchical: a second attention block is added to attend to the representations output by the parallel cross-attention. While models in Table 3 all use the parallel cross-attention described in § 2.2, Table 5 ablates different multi-source cross-attention mechanisms. Three out of four cross-attention achieve similar performance, whereas the 'flat' attention is considerably worse. This echos the findings by Libovický et al. (2018).

Config.	BLEU	Config.	BLEU
Serial	34.1	Flat	24.9
Parallel	34.0	Hierarchical	34.1

Table 5: Indic test set BLEU scores of multi-encoder architecture trained on BASE+INSCRIP using different multi-source cross-attention. All mechanisms perform similarly except *flat* cross-attention.

B Experiments

B.1 Data statistics

The number of training examples for each language in both Turkic and Indic dataset is shown in Table 7. We evaluate the Turkic dataset on multi-way FloRes101 devtest set, each having 1012 examples. To evaluate the Indic models, we use the provided multi-way test set of WAT21 MultiIndic task, each having 2390 examples.

B.2 Input concatenation analysis

In § 4, results show that models trained on the concatenated input does not bring any discernible improvement, but rather the performance is almost the same. To understand if the model has indeed utilized the concatenated alternative signals, we take the trained model and evaluate BLEU scores on the corrupted input. Specifically, one part of the concatenated input is corrupted while the other is left intact. The corruption is done by shuffling the tokens within the selected part of the input. Overall, we find that the model indeed pays attention to both parts of the input, as corrupting any part of them leads to large regression in BLEU scores (Table 6). Moreover, no matter which type of signal is put in the front of the sentence, the model always pays more attention to the original input rather than the phonetic input, since corrupting the original input causes larger performance degradation than corrupting the phonetic input.

Config.	BLEU	Config.	BLEU
BASE + IPA	23.3	IPA + BASE	23.0
BASE' + IPA	3.3	IPA' + BASE	13.9
BASE + IPA'	20.2	IPA + BASE'	9.5

Table 6: Models trained on concatenated original and phonetic input while evaluated on partially corrupted input. We use IPA' to denote the phonetic part of the input is in corruption. Results are reported on the FloRes101 Indic languages instead of MultiIndic test set.

C Example alternative input signal

We present example alternative signals in Figure 4 and Figure 5. When the input are transformed to scripts other than their native script, there are more shared tokens in the source languages (as highlighted in Figure 5).

D Analysis

D.1 Token overlap details

In § 4 we show the token overlap of various signals aggregated over all source language pairs, in this section we show the token overlap of *each* source language pair in Table 8 for the original input and in Table 9 for the in-family transliterated input[15]. Before performing transliteration, all source languages share only a small amount of token overlap except between Marathi and Hindi. The shared tokens between native scripts are mostly punctuation marks, digits and English tokens. After transliteration, the token overlap becomes more obvious and a clear division between language families can be found.

D.2 Similarity in latent space

Besides examining the consistency of system output as in § 4.2, we also measure the distance of

[15] Target script for Indo-Aryan languages is Oriya and Dravidian languages Kannada.

BASE	_ପ୍ରଧାନମନ୍ତ୍ରୀ _କହିଥିଲେ _, _କୋଟି _କୋଟି _ଲୋକ ଙ୍କ _ମନ _ଏବଂ _ମ ସ୍ତ ି ଷ ୍କ ରେ _“ ଅ ଭି ଳା ଷ ା ” _ସ ୁଷ୍ଟି ରେ _କରିବ ାରେ _ବା ବା ସା ହେ ବ _ଆ ମ୍ ବେ ଦ କ ର _ଗୁରୁ□□□ ପୂର୍ଣ୍ଣ _ଭୂ ମିକ ା _ନିର୍ ବା ହ _କରିଥିଲେ _।
IPA	_prɔdhanɔmɔntɾi _kɔhithile _koʈi _koʈi _lokɔŋkɔ _mɔnɔ _ebɔŋ _mɔ sti s kɔɾe _ɔbhi l ̩ a sa _sɾusʈi ɾe _kɔɾibaɾe _baba sa he bɔ _am bedɔ kɔɾɔ _gɔ _hɾɔʃʃoukaɾ _ɾɔ _hɾɔʃʃoukaɾ _ʈɔ _halʌnt _letəbi : satɔekɔ _pɔ _di : ɾghukaɾ _ɾɔ _halʌnt _muɾddheɳɳɔ _halʌnt _muɾddheɳɳɔ _bhu : mika _niɾba h _kɔɾithile
ROMANI	_pradhanamanthri _kahithile _, _koti _koti _lok ank _man _eban _m asth ishk are _“ _ab hil asha _” _srist ire _kariv are _bab asa hib _ambed akar _guruthpurna _bhoomika _nirvah _karithile _.
TRANSL	_प्रधान म न्त ्री _कह ि थ िले _, _को ट ि _को ट ि _लोक ङ ्क _मन _ए ब ं _म स्त ि ष् कर े _“ अ भ ि ळा षा ” _स ु ष्ट ि रे _कर ि ब ारे _बाब ासा हे ब _आम ्ब ेद कर _गुर ुत ्प ूर्ण ्ण _भूमिका _निर् ब ाह _कर ि थ िले _.

Figure 4: Example alternative signals. **BASE** is the original input in Oriya script, **IPA** is the phonetic input, **ROMANI** the romanized input, and **TRANSL** (INSCRIP in the main text) the input transliterated into Devanagari script.

Turkic languages		Indic Languages			
Language	**#bi-text**	**Language**	**#bi-text**	**Language**	**#bi-text**
Kazakh	919,877	Bengali	1,756,197	Marathi	781,872
Kyrgyz	243,179	Gujarati	518,015	Oriya	252,160
Turkish	4,000,000	Hindi	3,534,387	Punjabi	518,508
Uzbek	156,615	Kannada	396,865	Tamil	1,499,441
Azerbaijani	1,847,723	Malayalam	1,204,503	Telugu	686,626

Table 7: Training data statistics for Turkic and Indic dataset.

	bn	hi	pa	or	gu	mr	kn	ml	ta	te
bn		0.05	0.04	0.04	0.01	0.01	0.01	0.01	0.01	0.01
hi	0.05		0.06	0.05	0.02	0.18	0.01	0.01	0.02	0.01
pa	0.04	0.06		0.04	0.02	0.02	0.01	0.01	0.02	0.01
or	0.04	0.05	0.04		0.01	0.01	0.01	0.01	0.01	0.01
gu	0.01	0.02	0.02	0.01		0.05	0.04	0.04	0.05	0.04
mr	0.01	0.18	0.02	0.01	0.05		0.04	0.05	0.05	0.04
kn	0.01	0.01	0.01	0.01	0.04	0.04		0.04	0.05	0.04
ml	0.01	0.01	0.01	0.01	0.04	0.05	0.04		0.05	0.04
ta	0.01	0.02	0.02	0.01	0.05	0.05	0.05	0.05		0.05
te	0.01	0.01	0.01	0.01	0.04	0.04	0.04	0.04	0.05	

Table 8: Token overlap BASE

	bn	hi	pa	or	gu	mr	kn	ml	ta	te
bn		0.33	0.26	0.29	0.32	0.29	0.03	0.03	0.03	0.03
hi	0.33		0.49	0.26	0.49	0.4	0.03	0.03	0.03	0.03
pa	0.26	0.49		0.19	0.37	0.34	0.03	0.03	0.03	0.03
or	0.29	0.26	0.19		0.23	0.22	0.01	0.01	0.01	0.01
gu	0.32	0.49	0.37	0.23		0.41	0.03	0.03	0.03	0.03
mr	0.29	0.4	0.34	0.22	0.41		0.03	0.03	0.03	0.03
kn	0.03	0.03	0.03	0.01	0.03	0.03		0.25	0.17	0.31
ml	0.03	0.03	0.03	0.01	0.03	0.03	0.25		0.35	0.33
ta	0.03	0.03	0.03	0.01	0.03	0.03	0.17	0.35		0.26
te	0.03	0.03	0.03	0.01	0.03	0.03	0.31	0.33	0.26	

Table 9: Token overlap INSCRIP

source representations in the latent space. Concretely, we compute the average of normalized Euclidean distance over all source language pairs:

$$\frac{1}{\binom{N}{2}} \sum_{m,n} dist(l_m, l_n)$$

, where N is the total number of source languages, $dist(l_m, l_n)$ compute the distance between a sentence in language m and language n:

$$dist(l_m, l_n) = \frac{1}{2}\Big(\frac{1}{|l_m|}\sum_i \min_j \big(dist(w_{mi}, w_{nj})\big) + \frac{1}{|l_n|}\sum_j \min_i \big(dist(w_{mi}, w_{nj})\big)\Big)$$

, where $|l_m|$ and $|l_n|$ are the number of tokens within sentence in language m and language n respectively. w_{mi} represents the i^{th} encoder output of sentence l_m. $dist$ represents the Euclidean distance between the two vectors. We additionally normalize the distance with $\sqrt{d}$ where d is the dimension of the dense vector. Adding the scaling factor is to make the scaled self-ensemble model comparable with the rest variants.

As shown in Table 10, none of the alternative signals alone can lead to more similar source representations. While training on the original input and one alternative input, only the combination of BASE and INSCRIP lowers the distance of original input representations from 0.60 to 0.58. The distances become even more smaller while training the scaled self-ensemble model. The distances

Config.	BASE	IPA	ROMANI	INSCRIP
Trained separately	0.60	0.62	0.60	0.61
SE(BASE+IPA)	0.60	0.62	-	-
SE(BASE+ROMANI)	0.61	-	0.60	-
SE(BASE+INSCRIP)	0.58	-	-	0.60
SE(ALL)	0.60	0.62	0.60	0.61
S-SE(ALL)	0.54	0.53	0.52	0.52

Table 10: Normalized Euclidean distances of single-input model (Trained separately), self-ensemble model (SE) and scaled self-ensemble model (S-SE).

among BASE representations decrease to 0.54 and the rest three input signals all yield more similar representations than the original input. Overall, we didn't find significant differences in latent space, which we would like to keep investigating in the future.

	BASE	IPA	ROMANI	TRANSL
bn	_প্রধানমন্ত্রী _বলেন ...	_prɔɖhanmɔntri _bɔlen...	_pradhanmantri _balen ...	_ପ୍ରଧାନମନ୍ତ୍ରୀ ...
gu	_પ્રધાનમંત્રીએ _કહ્યું _હતું _કે ...	_prədha : nəmntri : e : ...	_pradhanmantri _kahyu ...	_ପ୍ରଧାନ ମ ଂ ତ ୍ରୀ ...
hi	_प्रधानमंत्री _ने _कहा _कि ...	_prədha : nəmʌntri _ne : ...	_pradhanmantri _ne ...	_ପ୍ରଧାନ ମ ଂ ତ ୍ରୀ ...
kn	_ಬ ಾ ಬ ಾ _ಸಾ ಹ ೇ ಬ್ ...	_ba : ba : _sa : he : b _ɐm be...	_b aab a _sahe b _ambedkar ...	_ବ ା ବ ା _ସା ...
ml	_കോട ിക്ക ണ ക്ക ിന് ...	_ko : ɖik : ɐ ɳək : inɨ ...	_kot ic nak in _janath ute ...	_ಕೋ ಟಿ ಕ್ಕ ಣ ಕ್ಕ ...
mr	_करो ड ो _लोक ांच्या ...	_kəro : ɖo : _lo : kãc : ja : ...	_karod o _lok anchya _mana ...	_କର ୋ ଡ ୋ ...
or	_ପ୍ରଧାନମନ୍ତ୍ରୀ _କହିଥିଲେ...	_prɔdhanɔmɔntri _kɔhithile ...	_pradhanamanthri _kahithile ...	_ପ୍ରଧାନମନ୍ତ୍ରୀ ।...
pa	_ਪ੍ਰਧਾਨ _ਮੰਤਰੀ _ਨੇ _ਕਿਹਾ _ਕਿ ...	_prədhan _mʌntəri _ne _kɪha ...	_pradhan _mantri _ne ...	_ପ୍ରଧାନ _ମ ଁ ...
ta	_கோ டிக்க ண க்கான _மக்கள	_ko : ɖi kkʌɳʌkka : nʌ _mʌkkʌɭin ...	_kot ik n akk ana _makalin ...	_ಕೋ ಟಿ ಕ್ಕ ಣ ಕ್ಕ ...
te	_కో ట్లా ది _ప్రజల _హ ృ ద ...	_ko : ʈla : di _prajala ...	_kot l adi _prajal _hrid ayal ...	_ಕೋ ಟ್ ಲ ಾದ ಿ ...

Figure 5: Example alternative signals of the same sentence in ten Indic languages. The token overlap across multiple languages are highlighted in blue. Compared to the original input, transliteration significantly increases token overlap.