Unified NMT models for the Indian subcontinent, transcending script-barriers

Gokul NC
Devnagri AI
gokulnc@devnagri.com

Abstract

Highly accurate machine translation systems are very important in societies and countries where multilinguality is very common, and where English often does not suffice. The Indian subcontinent (or South Asia) is such a region, with all the Indic languages currently being under-represented in the NLP ecosystem. It is essential to thoroughly explore various techniques to improve the performance of such low-resource languages at least using the data available in open-source, which itself is something not very explored in the Indic ecosystem. In our work, we perform a study with a focus on improving the performance of very-low-resource South Asian languages, especially of countries in addition to India. Specifically, we propose how unified models can be built that can exploit the data from comparatively resource-rich languages of the same region. We propose strategies to unify different types of unexplored scripts, especially Perso–Arabic scripts and Indic scripts to build multilingual models for all the South Asian languages despite the script barrier. We also study how augmentation techniques like back-translation can be made use-of to build unified models just using openly available raw data, to understand what levels of improvements can be expected for these Indic languages.

1 Introduction

The Indian subcontinent is a well-studied linguistic area (Emeneau, 1956), known as South Asian sprachbund. The region is home to around a quarter of the world's population, with a total which is projected to reach more than 2 billion in a decade. Despite this, the progress in natural language processing is significantly lacking for South Asian languages (or Indic languages). Especially, machine translation is of core importance since South Asia is largely a multilingual society, with more than 25 languages recognized officially across the subcontinent and more than 100s attested and spoken. Although there are quite a few number of works which have released datasets for languages of India (Siripragada et al., 2020) and studied multilingual models for the same (Philip et al., 2019), they are not exhaustively studied. In particular, the Indic languages of other South Asian countries like Pakistan, Nepal and Sri Lanka are almost never studied together with the languages of India and Bangladesh.

In this work, we aim to study all the available Indic languages (of Indo-Aryan and Dravidian families) of all the above countries together, precisely 15 South Asian languages (listed in appendix A). Especially, we propose a simple strategy to unify digraphic languages like Hindi–Urdu, Sindhi and Punjabi which are written in Indic scripts in India and Perso–Arabic scripts in Pakistan. We propose how one can build a script-agnostic encoder which can generalize well across different types of translation models, like code-mixed, roman (social media) and formal texts. We study for the first time in literature backtranslation-based NMT for all script-unified Indic languages together, which provides significantly better performance than models trained only on parallel data, by using only freely available monolingual data. We finally provide brief recommendations for researchers working in this Indic-NMT domain, and finally mention how this work can be extended and its future scope.

2 Related works

Training multilingual models for neural machine translation currently the go-to approach for significantly improving the performance of low-resource languages (Ngo et al., 2020). Especially sharing of sub-word vocabulary among related languages (of the same or similar families) is of more importance to exploit the inter-relationships between the languages (Khemchandani et al., 2021), so that resource sharing from high-resource languages to

Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing, pages 227 - 236

Dataset	as	bn	gu	hi	kn	ml	mr	ne	or	pa	sd	si	ta	te	ur
Samanantar	0.14	8.52	3.05	8.57	4.08	5.85	3.32		1.00	2.42			5.17	4.84	
CVIT-PIB	0.04														0.20
Anuvaad*	.003								0.02						0.02
PMI*															0.01
OPUS	0.03							2.25	0.12		1.89	8.53			8.69
U.Kathmandu								0.02							
Charles Univ															0.01
MTurks 2012															0.03
Total	0.21	8.52	3.05	8.57	4.07	5.85	3.32	2.28	1.14	2.42	1.89	8.53	5.17	4.84	8.97

Table 1: Open-source parallel Indic corpora (in millions), totalling around 69M sentence-pairs

low-resource languages is achieved. Recent works (Ramesh et al., 2021) have explored strategies to train multilingual NMT for 11 languages of India, both with and without shared vocabulary across languages, demonstrating that vocabulary sharing by script unification is significantly beneficial. It is also common to convert all the text across all languages to IPA (International Phonetic Alphabet) or any common script, especially in speech-to-text (Javed et al., 2021) and text-to-speech (Zhang et al., 2021) to obtain a universal representation of text across any language/script. In the case of South Asian languages, it is more convenient to map all scripts to a common Indic script (like Devanagari) which is capable of representing all phonemes used in the Indic families (Khare et al., 2021).

3 Background

This section sets provides the background required for the subsequent sections.

3.1 Datasets

As mentioned earlier, our work only focuses on open-source datasets inorder to explore how performance can be improved for low-resource languages just using openly available data. Overall, the datasets used in this work are mostly from the general domain, and hyperlinks are provided to access all the datasets. The next sub-section mentions the list of all aligned datasets used in this work and further, the we mention the list of all available monolingual data sources which we exploit in this work for improving performance.

3.1.1 Parallel datasets

Table 1 shows the list of all parallel datasets used for training our models. It is to be noted that the Samanantar (Ramesh et al., 2021) is the major source of data, for languages of India. To explore more languages as well as to study how the above data is useful for other similar Indic languages, especially focusing on other related South Asian countries, we gather more data from different sources shown in the same table. Specifically, we aim at increasing the amount of data obtainable for Indo-Aryan languages not covered in Samanantar, viz. Nepali, Sinhala, Sindhi and Urdu which are predominantly spoken in Nepal, Sri Lanka and Pakistan respectively. In addition, we also manually add new sources of data (marked *) from Anuvaad corpus and PM India corpus which were not covered in the latest Samanantar v0.2 for Assamese and Odia, although relatively very small in size.

3.1.2 Benchmark dataset

For test set, we use the FLoRes101 benchmark (Goyal et al., 2021) which has data for 14 Indic languages, manually translated from various domains of English Wikipedia. Since this new benchmark does not have data for Sinhala, we evaluate it on the initial FLoRes benchmark (Guzmán et al., 2019). Note that we do not use the WAT 2021 MultiIndicMT testset (Nakazawa et al., 2021) for benchmarking, since we find the data quite very close to the distribution of the corresponding training data, as also observed by IndicBART (Dabre et al., 2021). All BLEU scores reported in this paper are computed using sacreBLEU (Post, 2018) after generating translations with a beam decoding size of 4. Note that we compare our scores only against IndicBART (and experiment only with same architecture), as they already demonstrate superior scores over fine-tuned models like mBART and the chosen model is lighter than pretrained

models like mT5 or mBART50.

3.1.3 Monolingual data

Table 2 shows list of all monolingual corpora used in this work. It is to be noted again that the AI4Bharat IndicCorp is the major source of data (row 1), for languages of India. For Indo-Aryan languages of other South Asian countries, we consolidate most of the available open-source corpora from different sources as shown in other rows of the table. We also try to consolidate more data for very-low-resource languages of India like Assamese and Odia.

3.2 Script Unification

As explained earlier, script unification is essential for sub-word vocabulary sharing between related languages. It is essential for the unification to be lossless so that the resultant dataset quality is not affected. In literature, it is common to use Devanagari as the common script to unify all the Brahmic scripts of India, although any script (like IPA) can be used as the pivot. For example, for models trained only for Dravidian languages, we use the Malayalam script as the common representation for the 4 languages: Kannada, Malayalam, Tamil and Telugu. But Devanagari is predominantly chosen since it is used for many languages like Hindi, Nepali, Marathi, etc. as well as due to the fact that it is one of the few Indic scripts which supports almost all phonemes required for both the Indic language families, not just Indo-Aryan for which the script is predominantly used. One important aspect of Devanagari is a diacritic called nuqta, which is essentially a dot mark placed below the main consonants to represent non-native phonemes. Its primary use is to represent consonants of other languages, including from different families like Dravidian, Iranic (for Persian), Semitic (for Arabic). Hence, using Devanagari for all Indic languages as a common script is preferable, including languages like Urdu, Sindhi and Kashmiri which are written in Perso–Arabic scripts. In the subsequent section, we explain how the latter is achieved, which is an unexplored track in research.

3.2.1 Mapping Devanagari and Perso–Arabic

The Perso–Arabic script is an abjad, meaning that it is based on a writing system which mostly has only consonants (in its purest form). In addition, in Perso–Arabic, two of the same consonants (w & y) are used to indicate few long-vowels (respectively: /u/, /o/ and /i/, /e/). So the reader of the script mentally fills-in / interprets most of the vowels as they read, based on their knowledge of the language and context. Devanagari is an abugida, meaning that it is an alphasyllabary system where the script is generally expected to be almost phonetic with all consonants and vowels represented. This makes a direct mapping of Perso–Arabic consonants to Devanagari slightly illegible for readers of usual Devanagari due to lack of any vowels. Figure 1 below shows an example of raw mapping for the Hindostani language.

Urdu	پلس نے چور کو پکڑ کے جیل مے ڈال دیا
Raw-Devanagari	पलस ने चवर कव पकड़ के जयल मे डाल दया
Hindi	पुलिस ने चोर को पकड के जेल मे डाल दिया
English	The police caught the thief and put him in jail

Figure 1: Row-1: Perso–Arabic, Row-2: Devanagari-transliteration, Row-3: Actual Hindi spelling, Row-4: Translation

Despite this, we propose that NMT models are capable of learning both abjad and abugida forms, with a deeper understanding of the underlying language. That is, we directly use the raw mapping of Perso–Arabic consonants to Devanagari (without any phonetic transcription) to train an unified model. It is to be noted that there are some consonants in Perso–Arabic for which, although the phonemes are different, they represent the same phone. Those consonants usually are mapped to a single Devanagari phoneme. In our work, especially to generate Perso–Arabic texts, we require lossless mapping of each character from Perso–Arabic. Hence we propose to map them uniquely by creating new Devanagari consonants using nuqta.We also open-source our transliterator implementation as a python library[1].

Upon training using the above unification, we see that our model is capable of understanding that the standard registers of Hindi & Urdu have the same underlying language, with only differences being in writing form and formal vocabulary. This was verified by swapping the scripts used for Hindi & Urdu to see if still produces legitimate outputs. As later described in Section 5.1, while training, we explicitly specify what is the expected output script-type and language that is to be produced by the model. Upon specifying Arabic as script for Hindi and Devanagari as script for Urdu to the

[1] Indic-PersoArabic Script Converter

Dataset	as	bn	gu	hi	kn	ml	mr	ne	or	pa	pnb	sd	si	ta	te	ur
IndicCorp	2.38	77.7	46.6	77.3	56.5	67.9	41.6		10.1	35.3				47.8	60.5	
University[a]				45				3.2								5.5
CC100	0.5							12.7	2.2		0.02	1.4	12.6			28
Wikipedia	0.3							0.4	0.3		1.2	0.4	0.6			1.3
Leipzig	0.06							4.2	0.04		0.06	0.007	0.4			1.1
Crawled[b]								2					4.8			
Total	3.24	77.7	46.6	122.3	56.5	67.9	41.6	22.5	12.64	35.3	1.28	1.807	18.4	47.8	60.5	35.9

Table 2: Open-source monolingual Indic corpora (in millions), totalling 650M sentences

[a]hi: IIT-B Corpus, ne: JNU Corpus, ur: Charles University
[b]ne: GitHub sources, si: FacebookDecade, News sources, SinMin

trained model, we found that the model still produced Urdu and Hindi sentences respectively. Now we generate augmented data for Devanagari-Urdu and Arabic-Hindi by transliterating 1M Hindi parallel data to PersoArabic (later unified again to abjadi-Devanagari) and by transcribing 1M Urdu parallel data to Devanagari using Sangam transliterator (Lehal and Saini, 2012). We fine-tune the model for few epochs using this synthetic data. We observe that even using such small fraction of data, the model was able to easily generate translations for Urdu in proper Devanagari and for Hindi in proper-Arabic for unseen data, hence qualitatively proving the hypothesis that the script-unified model can also learn writing-system-agnostic features.

Furthermore, we perform something similar for Sindhi language – Sindhi is majorly spoken in Pakistan by 30M people & written in Perso–Arabic script; in India, it is spoken by around 2M people & officially mandated to be written in Parivardhit-Devanagari, an extended version of Devanagari. Since all the Sindhi datasets available are in Perso–Arabic, we use the same Sangam transliteration API as mentioned above to generate Sindhi datasets in Devanagari. We use data this as well to train the models in Section 5, and find that the model now was also able to produce (almost) same Sindhi outputs for both the scripts. Note that we implement a similar but separate converter for Sindhi script-unification, as the Perso–Arabic script for Sindhi has significant difference from that of Urdu. Also, since the amount of Sindhi corpus is very low, we augment the dataset while training with the following synthetic data – since Gujarati is a closely-related language to Sindhi, we sample 2M random Gujarati translation-pairs and create Arabic-Gujarati dataset and train for this artifical language–script combination as well in the training described in Section 5.1.

We would also like to point out that we do not perform this for the Punjabi language, which is written in an Indic script called Gurmukhi in India, and using a Perso–Arabic alphabet called Shahmukhi in Pakistan. This is because all available Punjabi datasets are in Gurmukhi, an almost phonetic script (similar to Devanagari). Hence we directly use our transliterator to convert from Gurmukhi to Shahmukhi and return the translation if required. But it was observed that due to the formal nature of the Punjabi datasets, the generated translations were of Eastern-Punjabi literary standard, hence the outputs may not always be mutually-intelligible to speakers who are used to Western-Punjabi literary standard. We do not find this issue significant in the case of Sindhi, as the formal Sindhi standards of both the countries do not differ much.

3.2.2 Mapping Sinhala and Devanagari

Sinhala alphabet (of Sri Lanka) is mostly similar in phonetics to most other alphabets of India, except a couple of minor differences. Sinhala has separate unicode points for representing 6 prenasal consonants, whereas in Devanagari, they are represented as ligature of a nasal consonant with another consonant, as shown in Figure 2. In addition, Sinhala also has short and long forms of the vowel /æ/ which we also map to Devanagari uniquely, for both dependent & independent vowels. The publicly available transliterators (like the transliterate sub-package in Indic-NLP-Library) are lossy, and do not handle all these cases.

Sinhala	ඟ ඬ ඳ ඹ ඳ ඹ
Devanagari	ङ्ग ञ्ज ञ्ज ण्ड न्द म्ब

Figure 2: Example mapping of pre-nasal consonants between Sinhala and Devanagari

3.2.3 Mapping between Indic scripts

For all the remaining scripts in this work, the mapping is mostly straightforward due to the fact that they follow the ISCII encoding scheme in which equivalent phonemes are mapped at same offsets in the unicode blocks. We use the AksharaMukha[2] tool to perform lossless transliteration between these Indic scripts.

3.3 Romanization of Indic languages

We also experiment with romanized models for all Indic languages in our work to translate to English. In this sub-section, we briefly explain the different ways using which we perform the romanization. Generally, there is no standard way to perform romanization for Indic languages, since the way one types it colloquially is quite personal in style. Hence we perform romanization using multiple ways. This includes machine learning-based romanization as well as rule-based romanization techniques which covers different possible ways of romanizing, which will be open-sourced[3].

In brief, for each language, we first generate 4 variants of romanization:

1. Raw & case-insensitive ASCII transliteration of the script (a readable lossy variant of the Velthuis scheme). For example, vowel diacritics are dropped (like ī→i, ū→u, etc.).
2. Approximate colloquial transcription of the script (taking into consideration phonological mapping to English, schwa deletion, etc.), and also involving language-specific random substitutions of related roman representations of consonants (like ph→f, v→w, etc.) and vowels (like ī→ee, ū→oo, etc.)
3. Consonant-only romanization (including initial vowels), to simulate (extreme) social media short-hand typing (not done for Urdu & Sindhi, as the roman variant-1 already does the same for languages that use Perso–Arabic scripts).
4. ML-based romanization using the python-library: LibIndicTrans[4].

We further generate generate 2 batches of the full dataset by mixing different variants of the above 4 romanizations at the word-level.

4 Indic to English MT

In this section, we explore different models for Indic to English translation using datasets mentioned in section 3.1.1. Note that before training, we perform text normalization of all datasets using the Indic-NLP-Library.

4.1 Experimental settings

The input sentence to the models is prepended with the language-tag token, "__*langcode*__ ", inorder to explicitly provides cues to the model about what the source language is. All the models experimented above are transformer-based, with the same network and hyperparameter configurations as in *transformer-big* (Vaswani et al., 2017), which has 6 encoder layers and 6 decoder layers inorder to be consistent with the scores comparison against the previous work (Dabre et al., 2021). For all experiments, we use the sentence-piece tokenizer (Kudo and Richardson, 2018) to build our sub-word vocabulary, with vocabulary sizes for input and output sides respectively 32000 (Indic side) and 16000 (English side). We use Marian-NMT toolkit (Junczys-Dowmunt et al., 2018) to train all our models, with mean cross-entropy as the loss function. Note that all models are trained from scratch.

4.2 Unified models

First, we build models from English specific to Indo-Aryan (ia2en) and Dravidian (dr2en) languages to compare how these models perform with respect to a model which is trained for both the Indic language families (in2en). As explained in section 3.2, we use Malayalam as the common script for Dravidian model and Devanagari for Indo-Aryan and Indic models.

Table 3 presents the performance across languages (ia2en and dr2en models are shown in same row for simplicity). We see that the Indic model trained on both the families outperform the scores

[2] https://github.com/virtualvinodh/aksharamukha

[3] https://github.com/GokulNC/Indic-Romanizer

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

Model	as	bn	gu	hi	kn	ml	mr	ne	or	pa	sd	si	ta	te	ur
	Indic-En														
IndicBART	-	30.7	33.6	36.0	27.4	30.4	30.0	-	28.6	34.2	-	8.5	27.7	32.7	-
ia2en, dr2en	21.4	30.2	32.8	36.1	25.3	27.7	28.9	35.1	28.4	34.2	24.1	12.8	22.5	29.6	24.9
in2en	23.9	31.8	33.9	36.8	28.1	30.7	30.7	36.2	31.3	35.3	24.1	15.1	27.7	33.0	25.1
rom_in2en	**24.1**	**31.9**	**34.0**	**37.3**	**28.4**	**30.9**	**30.7**	**36.3**	**31.5**	**35.3**	**24.7**	**15.3**	**28.3**	**33.0**	**25.8**
	En-Indic														
IndicBART	-	17.3	22.6	31.3	16.7	14.2	14.7	-	10.1	21.9	-	-	14.9	20.4	-
en2ia, en2dr	6.3	17.4	22.6	31.4	16.1	14.1	14.8	10.5	10.1	21.7	18.9	8.8	14.4	20.5	20.2
en2in	6.3	17.2	21.9	31.0	16.2	13.7	14.7	10.4	9.9	21.5	18.1	8.9	14.5	20.5	19.8
bt_en2in	9.9	18.9	23.1	34.2	18.7	16.2	16.1	17.1	14.3	23.9	23.7	14.1	17.2	22.3	22.3
bt_en2ia, bt_en2dr	**10.8**	**19.8**	**23.7**	**36.1**	20.0	17.3	**16.8**	**17.6**	**16.7**	**24.3**	**24.2**	**14.1**	17.2	**22.9**	**23.6**
t_bt_en2dr	-	-	-	-	**20.1**	**17.5**	-	-	-	-	-	-	**18.1**	22.8	-

Table 3: Comparison of BLEU scores of different trained models of same network architecture on FLoRes101 benchmark (Goyal et al., 2021) along with the scores of the existing best open-source model trained on Samanantar, taken from IndicBART paper (Dabre et al., 2021)

of family-specific models. This observation is consistent with the results for many other languages, where we see significant gains in accuracy with a shared encoder, in-cases like many-to-one NMT (Arivazhagan et al., 2019).

4.3 Script-agnostic model

We generate a romanized version of the parallel dataset available as explained in Section 3.3, which is typically 6x large in size due to different ways of romanization the **same** data, and train a Roman-Indic-to-English model (rom_in2en). Table 3 shows the performance of this romanized model. We see that the model is slightly better (on the romanized benchmark) than the *in2en* model. This can be attributed to the significant reduction in alphabet size of the model: Devanagari usually requires more than 80 characters (on average) to represent all Indic languages; whereas in the roman model, only 26 characters (though a bit lossy). Based on manual analysis, we infer that romanized models are slightly more robust to noise in inputs, owing to the varied nature of the romanized data. We also note that owing to increased amount of data in abjad form (due to romanization variant-3, shown in section 3.3), the performance of Sindhi and Urdu (which use Arabic scripts) have significantly improved.

In addition, to study how our model performs with real-world code-switched (roman) data, we attempt the Microsoft GLUECoS (Khanuja et al., 2020) Machine Translation task[5]. We fine-tune our model on the training set of the above dataset, and measure a validation BLEU score of 27.36. Unfortuanately, the leaderboard of the task is not yet out. Upon manually checking the validation results, we see that our model has performed reasonably good despite the fact that the dataset is code-mixed and romanization styles were somewhat different. Although this is not a comparable result, we hope that this is helpful in advancing further Indic-NMT research on this benchmark.

5 English to Indic MT

In this section, we explore one-to-many NMT models for training English to Indic translator. We initially train models using the parallel data, then train few more models using synthetic data from monolingual corpora to understand the level of improvement achievable using raw data.

5.1 Experimental settings

The input sentences to all the models is prepended with a novel type of language-tag token, "*__lang-code__ __script-type__* ", inorder to explicitly provide cues to the model about what script-type is to be produced (in-addition for the given language). The possible script types are: 1. '*a*' to denote Perso–Arabic writing system; 2. '*i*' to denote Indic writing system; 3. '*t*' to denote Tamil alphabet, which is a small subset of the Indic set.[6]

[5] https://github.com/microsoft/GLUECoS#code-mixed-machine-translation-task

[6] Tamil script is a lossy Indic alphabet, which has same phonemes for unvoiced and voiced consonants (like 'k' and 'g'), in-addition to a few other features (like aspirated consonants) that are not explicitly supported in the script. In

All the trained models follow the same network configuration (transformer-big) as in the previous experiments; see section 4.1. The sub-word vocabulary sizes for input and output sides respectively 16000 (English side) and 32000 (Indic side).

5.2 Models trained only on parallel data

We initially train 3 different models (from English) just using the parallel data: Dravidian (en2dr), Indo-Aryan (en2ia) and Indic (en2in). The results are shown in Table 3. We see that the performance does not vary much between the family-specific models and the common model. This observation is consistent with the results for many other languages, where we see trivial to almost-no gains in accuracy with a shared decoder, in-cases like one-to-many NMT (Arivazhagan et al., 2019).

We experiment in the next subsection to understand if a common model could be more beneficial than family-specific models when a huge backtranslated data is augmented with the (upsampled) original data.

5.3 Models trained on parallel and back-translated data

Using all the Indic monolingual data listed in Section 3.1.3, we generate English sentences using the *rom_in2en* model with a beam-search width of 6. We then train 4 models (from English) after upsampling the parallel data and concatenating with the backtranslated dataset: 1. *bt_en2in*: To all Indic languages after 5× upsampling; 2. *bt_en2ia*: To Indo-Aryan languages after 6× upsampling; 3. *bt_en2dr*: To Dravidian languages after 10× upsampling; 4. *t_bt_en2dr*: To Dravidian languages after 7× normal upsampling, and 3× Tamilized-augmented upsampling (by converting other Dravidian alphabets to Tamil subset and marking their script_type as '*t*' when prepending language token). The upsampling scale is decided such that the amount of original parallel data and backtranslated data are in ratio 1:2.

Table 3 shows the performance of all the 4 models. We see that, family-specific models perform notably better than a common model (given a fixed model size). Moreover, for the *t_bt_en2dr* model, we observe a significant boost in accuracy for Tamil after the Tamilized-data is augmented, and a trivial improvement for Malayalam and Kannada compared to *bt_en2dr*.

Section 5.3, we further clarify on how treating Tamil as a special case could be helpful to improve its performance.

It is also seen that, our model easily outperforms models which are fine-tuned from language models like IndicBART (Dabre et al., 2021). This is because we use the same entire monolingual data (Kakwani et al., 2020) which was used to pretrain IndicBART, but along with supervised translation signals in the form of backtranslated data.

For very-low resource languages (like Sindhi and Sinhala), we notice very significant improvements with back-translation, even with relatively lesser amount of monolingul data.

6 Discussions and Conclusion

We demonstrate in this paper various methods to achieve improvement in performance, especially across South Asian languages which were not previously explored along with the languages of India. We believe our presented contributions are more of exploratory nature, and make fundamental proposals (like always building romanized models when the source side is Indic). Although the fact that a unified model results in better performance in low-resource scenarios has been discovered by many prior work and hence not surprising, our work merely focuses on quantitatively studying the improvement in the case of Indic languages. In this section, we provide general suggestions for research groups working on NMT for Indic languages.

In general, to train model for any low-resource Indic language to English, we recommend that data from all the languages is used to train a multilingual model.[7] Especially, training a romanized model would be more beneficial, since it would be a script-agnostic model, and hence easily generalize for code-mixed and social media texts.

For training English to any low-resource Indic language, it maybe be preferable to train family-specific models when working under resource-contrained settings. Especially for languages of the countries Pakistan, Bangladesh, Nepal and Sri Lanka, we highly recommend and encourage them to exploit the datasets made available by researchers of India. If possible, it is highly recommended to exploit the abundant monolingual data and train models using backtranslated data.

[7] Works like (Dabre et al., 2021) have already shown why multilingual models are more preferable for Indic languages, so we do not redemonstrate it in our work.

6.1 Limitations

As generally known, bigger models could push the improvements even further than what we have seen in our results. In fact, the recent work by (Ramesh et al., 2022) show better results on the FLoRes101 benchmark by using a transformer-4x model even without using back-translated data. We only benchmark on transformer-2x in this work for consistent comparison, and to be more practical during training and inference (as well as due to our unaffordability of large infrastructure for such experimentations). Also, we only perform one round of back-translation to study English to Indic models in Section 5.3. We encourage researchers to study multiple rounds of back-translations (which is out of scope for this paper).

Thorough anlaysis of the performance on code-mixed (not code-switched) data using benchmarks like PHINC (Srivastava and Singh, 2020) is required for the *rom_in2en* model in Section 4.3, which is one of the on-going works in our research.

References

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges.

Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh M. Khapra, and Pratyush Kumar. 2021. Indicbart: A pre-trained model for natural language generation of indic languages.

M. B. Emeneau. 1956. India as a lingustic area. *Language*, 32(1):3–16.

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

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.

Tahir Javed, Sumanth Doddapaneni, Abhigyan Raman, Kaushal Santosh Bhogale, Gowtham Ramesh, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M. Khapra. 2021. Towards building asr systems for the next billion users.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in c++.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Shreya Khare, Ashish Mittal, Anuj Diwan, Sunita Sarawagi, Preethi Jyothi, and Samarth Bharadwaj. 2021. Low Resource ASR: The Surprising Effectiveness of High Resource Transliteration. In *Proc. Interspeech 2021*, pages 1529–1533.

Yash Khemchandani, Sarvesh Mehtani, Vaidehi Patil, Abhijeet Awasthi, Partha Talukdar, and Sunita Sarawagi. 2021. Exploiting language relatedness for low web-resource language model adaptation: An Indic languages study. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1312–1323, Online. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing.

Gurpreet Singh Lehal and Tejinder Singh Saini. 2012. Development of a complete Urdu-Hindi transliteration system. In *Proceedings of COLING 2012: Posters*, pages 643–652, Mumbai, India. The COLING 2012 Organizing Committee.

Toshiaki Nakazawa, Hideki Nakayama, Chenchen Ding, Raj Dabre, Shohei Higashiyama, Hideya Mino, Isao Goto, Win Pa Pa, Anoop Kunchukuttan, Shantipriya Parida, Ondřej Bojar, Chenhui Chu, Akiko Eriguchi, Kaori Abe, Yusuke Oda, and Sadao Kurohashi. 2021. Overview of the 8th workshop on Asian translation. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*, pages 1–45, Online. Association for Computational Linguistics.

Thi-Vinh Ngo, Phuong-Thai Nguyen, Thanh-Le Ha, Khac-Quy Dinh, and Le-Minh Nguyen. 2020. Improving multilingual neural machine translation for low-resource languages: French, English - Vietnamese. In *Proceedings of the 3rd Workshop on Technologies for MT of Low Resource Languages*,

pages 55–61, Suzhou, China. Association for Computational Linguistics.

Jerin Philip, Vinay P. Namboodiri, and C. V. Jawahar. 2019. A baseline neural machine translation system for indian languages.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Kumar Deepak, Vivek Raghavan, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Shantadevi Khapra. 2021. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages.

Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Srihari Nagaraj, Kumar Deepak, Vivek Raghavan, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Shantadevi Khapra. 2022. Samanantar: The Largest Publicly Available Parallel Corpora Collection for 11 Indic Languages. *Transactions of the Association for Computational Linguistics*, 10:145–162.

Shashank Siripragada, Jerin Philip, Vinay P. Namboodiri, and C V Jawahar. 2020. A multilingual parallel corpora collection effort for Indian languages. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3743–3751, Marseille, France. European Language Resources Association.

Vivek Srivastava and Mayank Kumar Singh. 2020. Phinc: A parallel hinglish social media code-mixed corpus for machine translation. In *WNUT*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Haitong Zhang, Haoyue Zhan, Yang Zhang, Xinyuan Yu, and Yue Lin. 2021. Revisiting ipa-based cross-lingual text-to-speech.

APPENDIX

A Indic languages

Language (& family)	ISO code	Script(s)	Countries
Assamese (Indo-Aryan)	as	Eastern Nagari	India
Bengali (Indo-Aryan)	bn	Eastern Nagari	Bangladesh, India
Gujrati (Indo-Aryan)	gu	Gujarati	India
Hindostani (Indo-Aryan)			
→ Hindi	hi	Devanagari	India
→ Urdu	ur	Perso–Arabic	Pakistan, India
Kannada (Dravidian)	kn	Kannada–Telugu	India
Malayalam (Dravidian)	ml	Malayalam	India
Marathi (Indo-Aryan)	mr	Marathi	India
Nepali (Indo-Aryan)	ne	Devanagari	Nepal
Oriya (Indo-Aryan)	or	Odia	India
Panjabi (Indo-Aryan)	pa	Gurmukhi	India
		Shahmukhi	Pakistan
Sindhi (Indo-Aryan)	sd	Perso–Arabic	Pakistan
		Parivardhita Devanagari	India
Sinhala (Indo-Aryan)	si	Sinhala	Sri Lanka
Tamil (Dravidian)	ta	Tamil	India, Sri Lanka
Telugu (Dravidian)	te	Telugu	India

Figure 3: List of all 15 South Asian languages studied in this work