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

Anonymous ACL submission

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 is such a region, with all the Indic languages currently being underrepresented in the NLP ecosystem. It is essential to advance the state-of-the-art of such low-resource languages atleast by using whatever data is available in open-source, which itself is something not very explored in the Indic ecosystem. In our work, we focus on improving the performance of very-low-resource Indic 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 Indic languages despite the script barrier. We also study how augmentation techniques like back-translation can be made use-of to build unified models that achieve state-of-the-art result among open source models, especially just using openly available raw data.

1 Introduction

The Indian subcontinent is home to around a quarter of the world's population, with a total which is about to hit 2 billion. Despite this, the progress in natural language processing is significantly lacking. Especially, machine translation is of core importance since South Asia is largely a multilingual society, with more than 20 Indic languages recognized officially[1] 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. Especially, we propose strategies 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 Indic languages together, which provides better performance than existing open-source models (using only freely available data), and present a brief survey of open-source monolingual corpora across low-resource languages. 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 low-resource languages is achieved. Recent works (Ramesh et al., 2021) have explored strategies to train multilingual NMT for Indic languages, both with and without shared vocabulary across languages, demonstrating that vocabulary sharing by

[1] https://en.wikipedia.org/wiki/Languages_of_Indian_subcontinent

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 Indic languages, it is more convenient to map all scripts to an Indic script like Devanagari which represents 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 show how state-of-the-art can be advanced for low-resource languages just using openly available data. The next subsection mentions the list of all aligned datasets used in this work and further, the appendix mentions 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 table.

3.2 Script Unification

As explained earlier, script unification is essential for sub-word vocabulary sharing between related languages. It is common to use Devanagari as the common script unify to all the Indic scripts of India, although any script (like IPA) can be used as the pivot. Devanagari is predominantly chosen since it is used for many languages like Hindi, Marathi, Nepali, 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 essential a dot mark placed below the main consonants to represent non-native phonemes. 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 mostly has only consonants, and the same consonants is used to indicate 2 long-vowels. So the reader generally fills in 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. The table below shows an example of raw mapping.

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

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.

However, 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[9].

[9]Redacted link to our Indic-PersoArabic-Script-Converter

Dataset	as	bn	gu	hi	kn	ml	mr	ne	or	pa	sd	si	ta	te	ur
Samanantar	142	8522	3054	8568	4076	5851	3322		1006	2422			5167	4842	
CVIT-PIB[2]	38														203
Anuvaad[3]	9								19						21
PMI[4]															11
OPUS[5]	29							2255	117		1892	8530			8689
U.Kathmandu[6]								21							
Charles Univ[7]															13
MTurks 2012[8]															34
Total	218	8522	3054	8568	4076	5851	3322	2276	1142	2422	1892	8530	5167	4842	8971

Table 1: Open-source parallel Indic corpora

3.2.2 Mapping Sinhala and Devanagari

Sinhala alphabet 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. In addition, Sinhala also has short and long forms of the vowel /æ/ which we also map to Devanagari uniquely.

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[10] encoding scheme in which equivalent phonemes are mapped at same offsets in the unicode blocks. We use the Akshara-Mukha[11] tool to perform transliteration between Indic scripts.

3.3 Romanization of Indic languages

We also experiment with romanized models for all Indic languages in our work. Generally, there is no standard way to perform romanization for Indic languages, since the way one types it colloquially is very personal. Hence we perform romanization using multiple ways. This includes machine learning-based romanization (using Lib-IndicTrans[12]), and rule-based romanization techniques which covers different possible ways of romanizing, which will be open-sourced.

[10] https://en.wikipedia.org/wiki/Indian_Script_Code_for_Information_Interchange

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

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

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.

4.1 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 later trained for both the Indic language families (in2en). As explained in section 3.2, we use Malayalam as the common script for Dravidian languages and Devanagari for Indo-Aryan and Indic models.

Table 2 presents the performance across languages. We see that the Indic model trained on both the families outperform the scores 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).

4.2 Script-agnostic model

We generate a romanized version of the parallel dataset available, 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 2 shows the performance of this romanized model. We see that the model is slightly better than the *in2en* model. This can be attributed to the drastic reduction in alphabet size of then model: Devanagari usually requires more than 80 characters (on average) to represent all Indic languages; whereas in the roman model, only 26 characters and a bit lossy.

Model	as	bn	gu	hi	kn	ml	mr	ne	or	pa	sd	si[13]	ta	te	ur
	Indic-En														
Samanantar	-	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														
Samanantar	-	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	18.7	8.9	14.5	20.5	20.1
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 2: Comparison of BLEU scores of different trained models along with the scores of the existing best open-source model trained on Samanantar, taken from IndicBART paper (Dabre et al., 2021)

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 a model using synthetic data from monolingual corpora to understand the level of improvement achievable using raw data.

The input sentences to all the models is prepended with a novel type of language-tag token, "*__langcode__ __script-type__* ", inorder to explicitly provides cues to the model about what the target language and script-type is. 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.

5.1 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 2. 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). But due to the fact that we are using more data than the previous work, our scores improves slightly over the AI4Bharat-IndicTrans model.

We still continue to experiment in the next section 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.

5.2 Models trained on parallel and back-translated data

Using all the Indic monolingual data listed in Appendix A.1, we generate English sentences using the *rom_in2en* model with a beam-size of 6. We then train 4 models from English after upsampling the parallel data and concating with backtranslated data: 1. *bt_en2in*: To all Indic languages after 5× upsampling; 2. *bt_en2ia*: To Indo-Aryan languages after 6× upsamling; 3. *bt_en2dr*: To Dravidian languages after 10× upsampling; 4. *t_bt_en2dr*: To Dravidian languages after 6× normal upsampling, and 4× Tamilized-augmented[14] upsampling.

Table 2 shows the performance of all the 4 models. We surprisingly see that, family-specific models perform notably better than a common model. Morever, for the *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 Kannda.

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.

[14] converting other Dravidian alphabet to Tamil subset

6 Conclusion

We demonstrate in this paper various strategies to improve the state-of-the-art open-source performance of Indic-NMT models. We believe our presented contributions are more of exploratory nature, and make fundamental proposal like always building romanized models when the source side is Indic. We further discuss more about the same in appendix Appendix D.

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.

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.

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.

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.

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.

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.

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.

A Datasets

A.1 Monolingual data

Table 3 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, for languages of India. For Indic languages of other South Asian countries, we consolidate most of the available open-source corpora.

Dataset	as	bn	gu	hi	kn	ml	mr	ne	or	pa	pnb	sd	si	ta	te	ur
AI4B IndicCorp[15]	2.38	77.7	46.6	77.3	56.5	67.9	41.6		10.1	35.3				47.8	60.5	
University[16]				45				3.2								5.5
CC100[17]	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[18]	0.06							4.2	0.04		0.06	0.007	0.4			1.1
Crawled[19]								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 3: Open-source monolingual Indic corpora

B Model architectures

B.1 Indic to English models

The input sentences 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-base* (Vaswani et al., 2017), which has 6 encoder layers and 6 decoder layers inorder to be consistent with the scores comparison against the previous open-source state-of-the-art (Dabre et al., 2021). For all experiments, we use the sentencepiece 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).

B.2 English to Indic models

All the models trained on original parallel data follows the same networ configuration as in previous sub-section (transformer-base). For models trained with both backtranslated and original data, we found that the transformer-base model does not significantly improve the scores, hence we use the *transformer-big* standard with the same hyperparameters as in (Vaswani et al., 2017) to train those models. The sub-word vocabulary sizes for input and output sides respectively 16000 (English side) and 32000 (Indic side).

C Additional evaluations

C.1 Script-agnostic (romanized) model

In addition, to study how our model performs with real-world code-mixed data, we attempt the Microsoft GLUECoS (Khanuja et al., 2020) Machine Translation task[20]. 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; hence upon manually checking the validation results, we see that our model has performed significantly better despite the fact that the dataset is code-mixed and romanization styles were somewhat different. We believe that although this is not a concretely comparable result, this is definitely helpful in advancing the Indic NMT research.

D Discussions

We demonstrate in this paper various methods to achieve improvement in performance, especially across Indic languages which were not previously explored along with the languages of India. In this section, we provide suggestions for other research groups working on NMT for Indic languages. In general, to train model for any Indic language to English, we recommend that data from all the languages is used to train a multilingual model. 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 (obviously any Indic text present should be romanized).

For training English to any low-resource Indic language, it can be preferable to train family-specific models. Especially for languages of the countries Pakistan, Bangladesh, Nepal and Sri Lanka, we highly recommend and encourage them to exploit the abundantly available from datasets by researchers of India. For (near-)high-resource languages (like Hindi), it maybe beneficial to focus only on unilingual models, or multilingual model with only very similar languages in the train-set

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

(like Punjabi, Gujarati).

If possible, it is highly recommended to exploit the abundant monolingual data and train models using backtranslated data. We also encourage researchers to try multiple rounds of backtransliteration as in.