Normalization and Back-Transliteration for Code-Switched Data

Dwija Parikh and **Thamar Solorio**
Department of Computer Science
University of Houston
Houston, TX 77204-3010
{dkparikh, tsolorio}@uh.edu

Abstract

Code-switching is an omnipresent phenomenon in multilingual communities all around the world but remains a challenge for NLP systems due to the lack of proper data and processing techniques. Hindi-English code-switched text on social media is often transliterated to the Roman script which prevents from utilizing monolingual resources available in the native Devanagari script. In this paper, we propose a method to normalize and back-transliterate code-switched Hindi-English text. In addition, we present a grapheme-to-phoneme (G2P) conversion technique for romanized Hindi data. We also release a dataset of script-corrected Hindi-English code-switched sentences labeled for the named entity recognition and part-of-speech tagging tasks to facilitate further research in this area.

1 Introduction

Linguistic code-switching (CS) is the phenomenon of mixing two or more languages in the context of a single utterance. Multilingual speakers around the world engage in code-switching on a regular basis. Code-switched data differs from monolingual data to a great extent which discourages use of existing NLP technologies on code-switched text. Code-switching also combines the syntax and lexicon of the languages used, making it difficult for monolingual models to adapt to code-switched data (Çetinoğlu and Çöltekin, 2019).

In textual code-switching, text is frequently romanized[1] due to various technical constraints. This is especially true in the case of Hindi-English since the Devanagari script for Hindi is not widely available or efficient on modern technology. Figure 1 shows an example of a code-switched Hindi-English tweet. As we can see in the example, keyboard layouts force users to choose a single script during time of purchase or adapt to using the standard QWERTY layout for transliterating multiple scripts. Since most users need to use English in their daily life, it is impractical to choose a different keyboard layout. The ease of convenience due to Latin script keyboard layouts and the lack of a standardized transliteration process leads users to employ ad-hoc phonetic transcription rules when transcribing Hindi in the Roman script (Aguilar and Solorio, 2020). Variations in transliteration and the informality of social media adds noise which makes Hindi-English code-switched data increasingly different from standard script text and harder to process. Further, transliteration also prevents from leveraging the resources available for standard Devanagari text like Wikipedia entries and monolingual models for Hindi.

Original: bhai...*why r u crying, film* me to boht maza a aajyega...!!😊😊😊
Translation: brother why are you crying, the film will be fun!

Figure 1: An example of a code-switched Hindi-English tweet. English text appears in italics and Hindi text is underlined.

Recent trends in NLP research on code-switching have explored the performance of large pre-trained models on code-switching tasks. State-of-the-art multilingual models are typically trained on standard script text like Wikipedia and struggle at adapting to transliterated noisy code-switched input. Transfer learning has emerged as a promising method to adapt monolingual models trained on high resource languages like English to code-switched data. Large pre-trained models like multilingual BERT (henceforth, mBERT) (Devlin et al., 2019) have shown robust cross-lingual zero-shot performance with code-switching data. Aguilar and Solorio (2020) demonstrated the cross-lingual transfer ability of ELMo (Peters et al., 2018) ,

[1]Throughout this paper, we use *romanized* to mean transliterated to the Roman script

Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, pages 119–124

https://doi.org/10.26615/978-954-452-056-4_015

which was trained on English, to Spanish-English, Hindi-English, and Nepali-English code-switched data. They observe that mBERT is outperformed by their model (CS-ELMo) for Hindi-English, possibly due to the fact that mBERT is trained on Hindi in Devanagari and their code-switched input is Romanized. In another study, Pires et al. (2019) tested mBERT's zero-shot performance on code-switched data in two formats: transliterated, where Hindi words are written in the Roman script, and corrected, where Hindi words have been converted back to the Devanagari script by human annotators. Their results show a substantial increase in zero-shot performance with script-corrected data. Other studies have also shown improvement in performance after normalization and back-transliteration on various tasks like named entity recognition and part-of-speech tagging (Ball and Garrette, 2018; Bhat et al., 2018). Thus, there is often a need for computationally inexpensive systems to prepocess data by normalization and/or back-transliteration.

We begin by providing background for the normalization and back-transliteration tasks. Then, we describe our system for normalization, grapheme-to-phoneme, and back-transliteration. Finally, we provide results and statistics of our system against human annotated data. Our contributions include: (1) a model to normalize phonetic typing variations, (2) a simplified back-transliteration technique, (3) a grapheme-to-phoneme conversion technique for romanized Hindi, and (4) publicly available data sets of script corrected Hindi-English text.

2 Related Work

Normalization. Research in phonetic typing variations when transliterating Hindi has gained increasing attention recently due to the presence of code-switched data on social media. Singh et al. (2018c) proposed a normalization model using skip-gram and clustering techniques for Hindi-English data. Mandal and Nanmaran (2018) presented the first sequence-to-sequence model for normalizing Bengali-English code-switched data.

Transliteration. Previous work in Hindi transliteration has fallen in two classes: rule based systems and machine translation based approaches. Multiple libraries like indic-transliteration[2] exist for simple transliteration tasks using rule based systems; however, they require input to be normalized and fail at adapting to non-standard data that is typical on social media. Before the advent of neural machine translation, statistical machine translation tools such as Moses (Koehn et al., 2007) were deployed for transliteration. Neural machine translation based approaches have continued to treat transliteration as a translation problem and applied methods such as sequence-to-sequence learning successfully. For instance, Bhat et al. (2018) proposed a three step encoder-decoder model for normalization and transliteration of Hindi-English code-switched text.

Grapheme-to-Phoneme. Grapheme-to-phomene (G2P) is an important task for speech recognition. Mortensen et al. (2018) presented a multilingual G2P system for transcribing a multitude of languages using simple mappings. G2P for standard Hindi is a straightforward task using simple phonetic mappings. However, for non-standard transliterated Hindi, it can be tricky to generate accurate phonemic representations.

3 Background

User generated code-switched data is noisy and riddled with word variations, spelling mistakes, and grammatical errors. Since the Latin script does not possess all the consonants and vowels required to transliterate Hindi, users come up with the most convenient ways to transcribe Hindi. Common variations in transliterated Hindi are:

- Ambiguous consonant transliteration: For consonants not covered by the Roman script, users rely on the most appropriate transliteration available which leads to multiple sounds being transliterated to the same grapheme in the roman script. For example, both दिल <*heart*> and डब्बा <*box*> are transliterated as *dil* and *dabba* respectively but the character <d> corresponds to different consonants in Hindi.

- Vowel dropping: Since native speakers of Hindi do not require explicit notation for vowels that can be easily inferred, they tend to skip their transcription in text. For instance, the Hindi word यार is generally transliterated as *yaar*. However, vowel dropping changes it to *yr*.

- Long vowel transliteration: Users transliterate long vowels in various ways. For ex-

[2] https://pypi.org/project/indic-transliteration/

ample, the most standard way to transliterate the word काम would be *kaam* but it is often transliterated as *kam*. During back-transliteration, this can be confused as कम instead of काम.

- Double consonant transliteration: Singh et al. (2018c) describe informal variations in double consonant transliteration, similar to long vowel transliteration, where users use variants with or without repeating the respective consonant. For example, इज़्ज़त can be transliterated as *izzat* or *izat*.

- Slang and abbreviations: We define some commonly used slang and abbreviations for both Hindi and English. Some examples include:

  btw -> by the way

  wassup -> what's up?

Besides the above, there are other non standard variations observed in transliterated Hindi as well. These variations make it difficult to properly transliterate text using simple phonetic mappings due to the lack of a standard transliteration scheme. Numerous schemes like WX notation (Chaitanya et al., 1996), BrahmiNet-ITRANS (Kunchukuttan et al., 2015), and others have been introduced. However, none of these have been widely employed by the general public.

4 Methodology

We follow a two step system to transliterate Romanized Hindi to the Devanagari orthography. First, we normalize the input using a sequence-to-sequence model. Then, for the back-transliteration task, we syllabify the token and transcribe to Devanagari. For the grapheme-to-phoneme task, we directly map the normalized tokens into the international phonetic alphabet (IPA).

5 Data

We use the hinglishNorm dataset by Makhija et al. (2020) to train the normalization model. The dataset comprises of romanized code-switched sentences and their normalized forms annotated by humans. The data contains both Hindi and English tokens along with their normalized forms. We create pairs of tokens and their normalized forms to train our model. We further augment the dataset with some frequently encountered Hindi words on social media and their variations.

6 Experiments

6.1 Normalization

Rule based systems are not the most efficient solution to normalization since they are not capable of capturing all possible variations. Instead, we treat normalization as a general machine translation problem. We train a character level sequence-to-sequence model for normalization following the architecture of Sutskever et al. (2014). The model is comprised of a Long Short-Term Memory(LSTM) encoder and LSTM decoder. We use the Keras library (Chollet, 2015) for training the model. Table 1 compares our model's performance with the baselines provided by Makhija et al. (2020). We evaluate our system using Word Error Rate (Nießen et al., 2000), BLEU score (Papineni et al., 2002), and METEOR score (Banerjee and Lavie, 2005).

Model	WER	BLEU	METEOR
(Makhija et al., 2020)	15.55	71.21	0.50
Ours	18.5	80.48	0.56

Table 1: Results showing the effectiveness of the normalization model using the WER, BLEU, and METEOR metrics.

It is likely that some of the errors are due to inconsistencies in the transcription scheme in the hinglishNorm dataset since it is annotated by humans. One such instance is the long vowel आ which is normalized to "aa" through most of the data. However, in some instances, the annotators normalize it to "a". For example, "bt control to krna pdega" from the training data is normalized to "but control to karana padega". A sample normalized output is shown in Table 2. Here we see that the Hindi token "bhai" has been normalized to "bhaai" while the English tokens "wher", "r", "u", and "frmm" have all been corrected to their correct spellings.

Original	**bhai**$_{\text{HIN}}$ wher r u frmm
Translation	*brother, where are you from?*
Normalized	**bhaai** where are you from

Table 2: An example of normalized output

6.2 Back-transliteration

Contemporary approaches treat transliteration using computationally intensive deep learning approaches. However, once data is normalized in effort to mitigate these variations, transliterating data does not require any sophisticated approaches.

Roman	IPA	Dev	Roman	IPA	Dev
k,q	kə	क	kh	k^hə	ख
g	gə	ग	gh	g^hə	घ
h	ɦə	ह	ch	t͡ʃə	च
chh	t͡ʃʰə	छ	j	d͡ʒə	ज
jh	d͡ʒʱə	झ	y	jə	य
sh	ʃə	श	t	tə	त
th	ʈʰ	थ	d	ɖ	द
dh	ɖʱ	ध	r	ɾə	र
n	nə	न	l	lə	ल
s	sə	स	p	pə	प
f,ph	pʰ	फ़	b	bə	ब
bh	bʱə	भ	m	mə	म
v	ʋə	व	z	zə	ज़

Table 3: Mappings for consonants

Table 3 shows mappings between orthographic forms and phonemic forms for consonants. Table 4 describes the corresponding mappings for vowels.

Roman	IPA	Dev	Roman	IPA	Dev
a	ə	अ	aa	ɑː	आ
i	i	इ	ee	iː	ई
u	u	उ	oo	uː	ऊ
ri, ru	r̩	ऋ	e	eː	ए
ai,ei	ɑːi	ऐ	o	oː	ओ
ou	ɑːu, ɔː	औ	am	əm	अं
ah	əh	अः			

Table 4: Mappings for vowels

A sample process for transliteration is outlined in Table 5.

Original	let's go **bhaai**$_{HIN}$ **abhi**$_{HIN}$ **kitnaa**$_{HIN}$ wait **karoge**$_{HIN}$
Translation	*let's go brother how long will you wait*
Transliterated	let's go भाई अभी कितना wait करोगे

Table 5: An example of back-transliteration

We test our system against human annotated data from the Xlit-Crowd[3] corpus for Hindi-English transliteration (Khapra et al., 2014). The corpus provides crowd-sourced data for romanized Hindi back-transliterated by human annotators. Results show that our system is **78.6%** accurate. Most of the errors are due to inconsistencies in transcription schemes and the rest are due to mistakes in normalizing by our model. For comparison, the popular indic-trans[4] library achieves 63.56% on the same data set (Bhat et al., 2015).

6.3 Grapheme-to-Phoneme

For the grapheme-to-phoneme task, we describe many-to-one mappings from romanized Hindi to IPA and Devanagari as shown in Tables 4 and 3. We use the Epitran[5] library by Mortensen et al. (2018) for transcribing English tokens to IPA. We extend Epitran with customized mappings for the Hindi tokens. Since the Roman script doesn't cover all the consonants required for transcribing Hindi, there are multiple ways of transcribing the same phoneme. However, prepocessing by normalization reduces the variation to a large extent. An example of grapheme-to-phoneme is provided in Table 6.

Original	let's go **bhaai**$_{HIN}$ **abhi**$_{HIN}$ **kitnaa**$_{HIN}$ wait **karoge**$_{HIN}$
Translation	*let's go brother how long will you wait*
IPA	lɛts gəʊ baɪ kɪtnɑ weɪt kəroːgeː

Table 6: An example of Grapheme to Phoneme

7 Released Datasets

We use our system to back-transliterate the Hindi-English corpora from the LinCE[6] benchmark (Aguilar et al., 2020). The NER corpus is from Singh et al. (2018a) and has 2,079 tweets while the POS tagging corpus is from Singh et al. (2018b) and has 1,489 tweets. Some statistics about the datasets are presented in Table 7.

8 Conclusion and Future Work

Our method can easily be extended to other languages that employ variations of the Devanagari

[3] https://github.com/anoopkunchukuttan/crowd-indic-transliteration-data

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

[5] https://github.com/dmort27/epitran

[6] https://ritual.uh.edu/lince/home

Task	Corpus	Hindi	English
NER	Singh et al. (2018a)	13,860	11,391
POS	Singh et al. (2018b)	12,589	9,882

Table 7: Statistics on the datasets

script, for instance Gujarati and Nepali. For other Romanized languages, simple phonetic mappings can be generated by domain experts. Using back-transliteration can help pre-process code-switched data to improve performance on a variety of tasks. We also plan to augment the normalization process with a dictionary of common word variations to make the normalization task more efficient. Our ongoing work includes testing performance of cross-lingual transfer on romanized and scrip-corrected text using multilingual models like mBERT.

9 Acknowledgements

This work was supported by the National Science Foundation (NSF) on the grant #1910192. We also thank Gustavo Aguilar for insightful discussions during preliminary investigations.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Gustavo Aguilar and Thamar Solorio. 2020. From English to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8033–8044, Online. Association for Computational Linguistics.

Kelsey Ball and Dan Garrette. 2018. Part-of-speech tagging for code-switched, transliterated texts without explicit language identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3084–3089, Brussels, Belgium. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. 2018. Universal Dependency parsing for Hindi-English code-switching. 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 987–998, New Orleans, Louisiana. 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.

Özlem Çetinoğlu and Çağrı Çöltekin. 2019. Challenges of annotating a code-switching treebank. In *Proceedings of the 18th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2019)*, pages 82–90, Paris, France. Association for Computational Linguistics.

Vineet Chaitanya, Rajeev Sangal, and Akshar Bharati (Group), editors. 1996. *Natural language processing: a Paninian perspective*, eastern economy ed edition. Prentice-Hall of India, New Delhi.

François Chollet. 2015. keras. `https://github.com/fchollet/keras`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mitesh M. Khapra, Ananthakrishnan Ramanathan, Anoop Kunchukuttan, Karthik Visweswariah, and Pushpak Bhattacharyya. 2014. When transliteration met crowdsourcing : An empirical study of transliteration via crowdsourcing using efficient, non-redundant and fair quality control. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*. The Association for Computational Linguistics.

Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-Net: A transliteration and script conversion system for languages of the Indian subcontinent. In *NAACL: System Demonstrations*.

Piyush Makhija, Ankit Kumar, and Anuj Gupta. 2020. hinglishNorm - a corpus of Hindi-English code mixed sentences for text normalization. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 136–145, Online. International Committee on Computational Linguistics.

Soumil Mandal and Karthick Nanmaran. 2018. Normalization of transliterated words in code-mixed data using Seq2Seq model & Levenshtein distance. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 49–53, Brussels, Belgium. Association for Computational Linguistics.

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

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece. European Language Resources Association (ELRA).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

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

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018a. Language identification and named entity recognition in Hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58, Melbourne, Australia. Association for Computational Linguistics.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018b. A Twitter corpus for Hindi-English code mixed POS tagging. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 12–17, Melbourne, Australia. Association for Computational Linguistics.

Rajat Singh, Nurendra Choudhary, and Manish Shrivastava. 2018c. Automatic normalization of word variations in code-mixed social media text.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.