Joint Approach to Deromanization of Code-mixed Texts

Rashed Rubby Riyadh and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta, Edmonton, Canada
{riyadh,gkondrak}@ualberta.ca

Abstract

The conversion of romanized texts back to the native scripts is a challenging task because of the inconsistent romanization conventions and non-standard language use. This problem is compounded by code-mixing, i.e., using words from more than one language within the same discourse. In this paper, we propose a novel approach for handling these two problems together in a single system. Our approach combines three components: language identification, back-transliteration, and sequence prediction. The results of our experiments on Bengali and Hindi datasets establish the state of the art for the task of deromanization of code-mixed texts.

1 Introduction

Ad-hoc romanization is the practice of using the Roman script to express messages in languages that have their own native scripts (Figure 1). The phenomenon is observed in informal settings, such as social media, and is due to either unavailability of a native-script keyboard, or the writer's preference for using a Roman keyboard. Rather than following any predefined inter-script mappings, romanized texts typically constitute an idiosyncratic mixture of phonetic spelling, ad-hoc transliterations, and abbreviations. A great deal of information is lost in the romanization process due to the difficulty of representing native phonological distinctions in the Roman script. This makes *deromanization* of such messages a challenging task (Irvine et al., 2012).

Another phenomenon that further complicates the task of deromanization is *code-mixing*, which occurs when words from another language (typically English) are introduced in the messages (e.g., the word *decent* in Figure 1). Code-mixing is particularly common in multi-lingual areas such as South Asia (Bali et al., 2014). In many cases, the English words have no transliterated equivalents in the native language and script.

(a)	tomake	to	decent	mone	hoyechilo
(b)	B	B	E	B	B
(c)	তোমাকে	তো	decent	মনে	হয়েছিল
(d)	"you"	"like"	"decent"	"in mind"	"was"
(e)	"you seemed a decent person"				

Figure 1: An example Bengali sentence that involves both romanization and code-mixing: (a) original message; (b) implied language tags; (c) target deromanization; (d) word-level translation; (e) sentence-level translation.

In this paper, we address the task of deromanization of code-mixed texts. This normalization process is necessary in order to take advantage of NLP resources and tools that are developed and trained on text corpora written in the standard form of the language, which in turn can facilitate tasks such as sentiment analysis and opinion mining in the social media. In addition, web-search queries are often expressed in a romanized form by speakers of languages that use non-Latin scripts, such as Arabic, Greek, and Hindi (Gupta et al., 2014b).

The task of deromanization of code-mixed texts is related to the study of language variation. Ad-hoc romanization represents a language variety, which resembles the usage of multiple scripts in some languages (e.g., Tajik). Code-mixing can also be considered a language variety, which exhibits similarities to dialects whose lexicons are strongly influenced by a different language (e.g., Upper Silesian).

The individual sub-tasks of deromanization of code-mixed texts have been investigated in prior work, but we are the first to incorporate them in a single system. Workshops and shared tasks have been devoted to code-mixing, including the prob-

Proceedings of VarDial, pages 26–34
Minneapolis, MN, June 7, 2019 ©2019 Association for Computational Linguistics

lem of word-level language identification (Chittaranjan et al., 2014; Choudhury et al., 2014). Transliteration and back-transliteration is a well-understood problem, which also has been the topic of several shared tasks (Duan et al., 2016; Chen et al., 2018). However, unlike romanization, transliteration is focused on names rather than dictionary words, and usually performed without considering the context of the word in a sentence. Finally, a number of papers address the deromanization of social media contents and informal texts, but propose no effective way of handling the code-mixing issue (Irvine et al., 2012; May et al., 2014). We show that this limitation leads to sub-optimal performance on deromanization.

In this paper, we propose a novel approach for tackling the problem of romanization and code-mixing together in a single system. Since sufficiently large annotated data sets for training an end-to-end approach are not available, we combine supervised models for the three main components of the complete task: (a) word-level language identification, (b) back-transliteration, and (c) word sequence prediction. These modules involve several diverse techniques, including neural networks, character-level and word-level language models, discriminative transduction, joint n-grams, and HMMs. We perform experiments on three datasets that represent two languages, including a new dataset that we have collected and annotated ourselves. The results show that our system is substantially more accurate than Google Translate, which is the only publicly available tool that can be applied to this task.

Our main contributions are: (1) a novel approach to deromanization of code-mixed texts through the combination of word-level language identification, back-transliteration, and sequence prediction; (2) a system that establishes the state of the art on the task; and (3) an annotated dataset of romanized Bengali messages. We make our code and data publicly available.[1]

2 Related Work

The tasks of deromanization and word-level language identification have been considered separately in the majority of the previous work.

[1] https://github.com/x3r/deromanization

2.1 Language Identification

While the identification of language of a monolingual document is a well studied problem, the task of word-level language identification has also garnered a fair amount of attention recently. A number of different approaches have been proposed for the task. Among the unsupervised approaches, dictionary-based and statistical language modeling approaches are the most common. Conditional Random Fields (CRF) and Support Vector Machines (SVMs) are among the most used supervised approaches.

The unsupervised approaches require no word-level annotation of mixed-code texts, but generally achieve low accuracy. Dictionary-based approaches make use of words and their frequencies in wordlists to determine the origin of a token (Barman et al., 2014; Das and Gambäck, 2014; Verulkar et al., 2015). However, those approaches cannot handle spelling variations and non-standard romanizations in code-mixed data. Statistical language modeling approaches employ n-gram probabilities which are derived from monolingual corpora. Both word and character n-grams have been used in the literature. The approach of Yu et al. (2013), which determines the probability of the next word being a code-switched word based on the previous n-words, achieves only 53% accuracy on the $Sinica$ (Mandarin-Taiwanese) corpus. A character n-gram based approach of Das and Gambäck (2014) achieves approximately 70% accuracy when tested on Bengali and Hindi.

The supervised approaches for language identification generally employ hand-crafted features such as capitalization information, character n-gram, and lexicon presence etc. CRFs make use of a set of features to determine the most probable language labels for a token sequence (King and Abney, 2013; Chittaranjan et al., 2014; Barman et al., 2014), and achieve accuracy in the low 90% on the evaluated languages. SVMs are also commonly employed for language classification (Barman et al., 2014) and achieve consistent performance (low 90%) on Bengali and Hindi. King and Abney (2013) employ Hidden Markov Models (HMM) trained using Expectation Maximization (EM) algorithm for the task, which can perform on par with the CRFs. Finally, supervised approaches that use contextual features generally outperform approaches that cannot utilize them.

2.2 Deromanization

Though a number of papers address the deromanization of social media contents and informal texts, they propose no effective way of handling the code-mixing issue.

Short Message Service (SMS) is a potential source of romanized texts due to the difficulty of typing in the native-script keyboard. A supervised deromanization approach of Irvine et al. (2012) uses an HMM to combine the candidates derived from a character-level transliteration model and a dictionary derived from automatically aligned words. The approach achieves 51% word-level accuracy on a self-annotated corpus of informal Urdu text messages.

Chakma and Das (2014) employ several supervised approaches for the automatic transliteration of code-mixed social media texts, which are based on joint source channel (JSC) and International Phonetic Alphabet (IPA). The experiments on Bengali-English and Hindi-English social media datasets show that the IPA-based approach outperforms the JSC-based approaches, achieving approximately 80% accuracy.

A supervised approach for converting Dialectical Arabic written in Latin script (Arabizi) to Arabic script of Al-Badrashiny et al. (2014) employs a character-level finite state transducer that generates transliteration candidates. A morphological analyzer is then used to filter the candidates and a language model to choose the output transliteration. The approach achieves 74% word-level transliteration accuracy on a Egyptian Arabizi SMS corpus.

A supervised approach of van der Wees et al. (2016) uses a character-based transliteration model which is incorporated as a component in the pipeline for an Arabizi-to-English phrase-based machine translation system. A contextual disambiguation with a character-level language model is then used for selecting between the transliteration candidates. When evaluated in the context of the NIST OpenMT evaluation campaign, the approach achieves 50% word-level accuracy on the Arabizi to Arabic deromanization.

Hellsten et al. (2017) propose a supervised approach for transliteration of romanized keyboard input to native scripts based on Weighted Finite State Transducers (WFST). The approach employs target word lists and pair language models constructed from the source and target alignments, and incorporates several weight pushing approaches for fast and memory-efficient decoding. The experiments are conducted on manually annotated Hindi and Tamil datasets and achieve 84% and 78% word-level accuracy, respectively. The approach was launched in the Google Gboard keyboard for 22 South Asian languages.

3 Methods

In this section, we present our approach for converting romanized code-mixed texts to their native scripts. It consists of three main components: language identification (Section 3.1), back-transliteration (Section 3.2), and sequence prediction (Section 3.3).

3.1 Language Identification

We approach language identification as a sequence labeling task, in which a sequence of word tokens in a code-mixed text is transformed into a sequence of the binary language tags (c.f., Figure 1b). Without loss of generality, we assume that one of the languages is English. Depending on the language label generated by this module, each input word is either fed into our back-transliteration module (Section 3.2) or copied unchanged to the final output.

Our supervised language identification module is based on the encoder-decoder model of Najafi et al. (2018).[2] On the language identification task, the encoder-decoder model achieves higher accuracy than a bi-directional Long Short Term Memory (LSTM) network with a CRF layer which is designed for sequence labeling tasks such as part-of-speech tagging (POS) and named entity recognition (NER) (Huang et al., 2015). This may be due to the nature of the language identification task, or the use of rich features in the encoder RNN.

The encoder takes character-level and word-level embedding of the input tokens as features in a bi-directional LSTM network over the input sequence. The outputs of bi-directional LSTM applied to characters of each word are concatenated and passed through a dropout layer to construct the character-level embedding. The capitalization pattern indicators (e.g. first letter is capital or all letters are capital) are then concatenated to these feature vectors. Pre-trained English word- and

[2] https://github.com/SaeedNajafi/ac-tagger

character-embeddings help the model identify English words in romanized texts. A fully-connected layer produces the final hidden vectors of the input sequence. The decoder's forward-LSTM generates output tokens incrementally from left-to-right; the output tokens are conditioned on the hidden vectors and the generated tokens from the previous steps. During the test phase, beam search is used to generate the outputs.

3.2 Back-Transliteration

Back-transliteration from romanized texts to the native scripts is difficult because there is generally only one correct way to render the romanized word to the native form (Knight and Graehl, 1998). We propose to overcome this problem by pooling the top-n predictions from three diverse transliteration systems: (a) Sequitur, a generative joint n-gram transducer; (b) DTLM, a discriminative string transducer; and (c) OpenNMT, a neural machine translation tool. In addition, we bolster the transliteration accuracy by leveraging target word lists, character language models, as well as synthetic training data, whenever possible. All of the generated candidate transliterations are then provided to the sequence prediction module, which is described in Section 3.3.

Sequitur (Bisani and Ney, 2008) is a data-driven transduction tool which derives a joint n-gram model from unaligned source-target data.[3] The model reflects the edit operations used in the conversion from source to target, and allows for the inclusion of source context in the generative model. Higher n-gram order models are trained iteratively from the lower order models. Sequitur was adopted as a baseline in the most recent NEWS shared task on transliteration (Chen et al., 2018).

DTLM is a new system that combines discriminative transduction with character and word language models (LM) derived from large unannotated corpora (Nicolai et al., 2018).[4] DTLM is an extension of DirecTL+ (Jiampojamarn et al., 2010). For target language modeling, which is particularly important in low-data scenarios, DirecTL+ uses binary n-gram features based exclusively on the forms in the parallel training data. This limitation often results in many ill-formed output candidates. DTLM avoids the error propagation problem that is inherent in pipeline approaches by incorporating the LM feature sets directly into the transducer. The weights of the new features are learned jointly with the other features of DirecTL+.

In addition, the quality of transduction is bolstered by employing a novel alignment method, which is referred to as *precision alignment*.[5] The idea is to allow null substrings on the source side during the alignment of the training data, and then apply a separate aggregation algorithm to merge them with adjoining non-empty substrings. This method yields precise many-to-many alignment links that result in substantially higher transduction accuracy. DTLM was among the best-performing systems at the recent NEWS shared task on transliteration (Chen et al., 2018).

As our neural transliteration system, we adopt the PyTorch variant of the OpenNMT tool (Klein et al., 2017).[6] The system employs an encoder-decoder architecture with an attention mechanism on top of the decoder RNN. We insert word boundaries between all characters in the input and output, resulting in translation models which view characters as words and words as sentences. We apply the default translation architecture provided by OpenNMT with the exception of using a bidirectional-LSTM in the encoder model. We optionally generate additional synthetic training data for the neural system, using a simple romanization table that maps each native script character to a set of English letters.

3.3 Sequence Prediction

The transliteration systems process individual words in isolation, and thus fail to take into account the context of a word in a sentence. However, multiple native words may have the same romanized form, so the top-scoring prediction is often incorrect in the given context. To solve this problem, we propose a sequence prediction system that attempts to select the best prediction from the pooled candidate list using both the transliteration score and the word trigram language model score.

We frame the task as a Hidden Markov Model, where the romanized words are the observed states, and the words in their original scripts are the hidden states. The emission probabilities are

[3] https://github.com/sequitur-g2p/sequitur-g2p
[4] https://github.com/GarrettNicolai/DTLM
[5] https://github.com/GarrettNicolai/M2MP
[6] https://github.com/OpenNMT/OpenNMT-py

Task	Language	Train	Tune	Test
LI	Bengali	18660 †	2000 †	539 †
	Hindi	15980 †	1780 †	3287 †
TL	Bengali	12623 ‡	1000 ‡	363 †
	Hindi	11937 ‡	1000 ‡	2465 †

Table 1: The size of the datasets used in our experiments. The sets from the FIRE 2014 and NEWS 2018 shared tasks are marked with † and ‡, respectively.

based on the prediction scores from the transliteration systems, which are normalized to represent valid probability distributions. The transition probabilities are based on the trigram probabilities from a word language model created with the *KenLM* language modeling tool.[7] The candidates and scores of all the systems are then concatenated together. If a transliteration candidate is generated by more than one system, the best prediction score among the systems is considered. Finally, the combined scores are normalized again.

We use a modified *Viterbi* decoder to determine the most likely transliteration sequence from the generated candidates. The scores are linearly combined to produce the score of a hidden sequence, $score(s)$. Due to the large number of n-grams and small number of transliteration candidates, the $score(s)$ is heavily skewed towards the emission scores. To mitigate this imbalance, we use exponent parameters p_t and p_e for the transition scores $T(s)$ and emission scores $E(s)$, respectively. These parameters are tuned on a development set. The scoring function is computed using the following formula:

$$score(s) = \max_k([\log T(s)]^{p_t} + [\log E(s)]^{p_e}) =$$

$$\max_k(\sum_{k=1}^{n}[\hat{T}(b_k|b_{k-1}b_{k-2})]^{p_t} + \sum_{k=1}^{n}[\hat{E}(e_k|b_k)]^{p_e})$$

where $\hat{T}$ represents the probability of transitioning from state b_{k-2} to state b_k, and $\hat{E}$ is the probability of observing e_k from state b_k.

Both generated transliteration candidates and foreign words in the code-mixed texts are sources of out-of-vocabulary (OOV) tokens. Prior to building the language model, we add a single UNK token to the corpus. During decoding, the identified English words and OOV transliterations are replaced with the UNK token. This results with OOV words being assigned very low probabilities, biasing the sequence prediction module towards in-vocabulary words.

[7] https://github.com/kpu/kenlm

Dataset	Bengali	English	Hindi	English
Train	48.0	52.0	46.2	53.8
Dev	87.1	12.9	-	-
Test	68.3	31.7	75.0	25.0

Table 2: The language balance in the code-mixed datasets (% of word tokens).

4 Data

In order to demonstrate the generality of our approach, we perform experiments on two languages: Bengali and Hindi. In addition to the datasets from the FIRE 2014 and NEWS 2018 shared tasks, we create our own annotated development set, and generate synthetic romanization data.

4.1 Code-mixed Data

The data for our language identification module is from the track on transliterated search of the FIRE 2014 shared task (Choudhury et al., 2014). The data consist of transliterated search queries, which include a substantial number of English words, as shown in Table 2. Search queries constitute a very different domain from social media messages that our system is designed for. In particular, they are rarely composed of complete sentences, which limits the ability of our sequence prediction module to take advantage of the word context.

We hold out approximately 10% of the original training data for tuning the hyper-parameters of our language identification module (Table 1). Since we have no access to the test sets of the shared task, we use the development sets as our test sets. These sets contain 100 Bengali and 500 Hindi search queries, respectively.

In order to mitigate the sparsity of annotated resources, we create our own Bengali development set. This allows us to develop and tune our system independently from the test sets. We collect romanized posts from several Facebook groups and pages, and manually deromanize them. Our development set contains 247 sentences and 1990 word tokens. The percentage of English word tokens in the development set is much smaller than in the FIRE 2014 datasets, as shown in Table 2.

System	Bengali		Hindi
	Dev	Test	Test
Majority baseline	87.1	68.3	75.0
Word-level ID	87.0	87.5	87.4
Bi-LSTM + CRF	92.4	90.9	93.2
Encoder-decoder	**95.2**	**92.2**	**95.3**

Table 3: Language identification accuracy (in %).

System	NEWS 2018 data		+ Annotated data	
	top-1	top-10	top-1	top-10
Sequitur	22.1	**58.7**	34.6	**69.3**
DTLM	29.1	43.9	40.5	61.5
NMT	**35.8**	52.1	**45.7**	63.4

Table 4: Impact of manually created transliteration data on the Bengali development set (in % word accuracy).

4.2 Transliteration Data

All back-transliteration systems are trained on the Bengali and Hindi datasets from the NEWS 2018 shared task (Chen et al., 2018). Their parameters are tuned on the corresponding development sets from the shared task. We create our back-transliteration test sets from the FIRE 2014 development sets by extracting romanized Bengali and Hindi word tokens together with their corresponding forms in the native scripts. As with the code-mixed data, the transliteration training and test sets are heterogeneous: the NEWS 2018 datasets contain mostly proper names, while the FIRE 2014 datasets contain mostly dictionary words.

We derive the character language models and target word lists for DTLM from publicly available unannotated monolingual corpora: Bengali Wikipedia[8], a Bengali news corpus[9], and a Hindi news corpus[10].

While no additional data is used for the Hindi models, we experiment with leveraging language-specific expertise to improve Bengali back-transliteration. First, since the training data contains mostly named entities, we augment it with manually-created transliterations of the most frequent 700 Bengali words from the news corpus. Second, since the performance of the neural system depends strongly on the amount of training data, we automatically generate romanizations for 50,000 Bengali words from Wikipedia.

The romanizations are generated with a context-free mapping from Bengali characters into Latin letters. As there are many ways to represent Bengali characters to Latin letters, and some Bengali characters have different representations based on their position in a word, we allow multiple mappings. For example, the Bengali character 'BO' is mapped to three English substrings: 'b', 'ba', and 'bo'. Each Bengali character is represented using on average 1.8, and maximum 3 Latin letters. (We make our mapping publicly available.[11]) In order to estimate the quality of the mapping script, we applied it to the Bengali-English training dataset from the NEWS 2018 shared task, which yielded 21.7% word-level and 78.9% character-level accuracy.

5 Experiments

In this section, we present the results for each of the three tasks.

5.1 Setup

We tune all parameters, including the exponents p_t and p_s, of our sequence prediction module, on the Bengali development set, and apply them unchanged to both test sets. For the transliteration, we set the n-gram order of Sequitur to 6. We apply a grid-search to establish the parameters for the DTLM transducer and aligner. We set parameters of the OpenNMT system to the default settings.

For the sequence prediction, we use pre-trained Glove word-embeddings of 100 dimensions (Pennington et al., 2014), and derive the character-embedding of 32 dimensions from the training data. The training is accomplished with Adam optimizer (Kingma and Ba, 2014), dropout regularization, and batch size of 64.

5.2 Language Identification

We evaluate our encoder-decoder model against a strong general sequence tagging system (Huang et al., 2015). For this purpose, we adapt an implementation[12] of a CRF-based sequence tagging model on top of RNNs to the language identification task (Bi-LSTM + CRF). In addition, we compare to a word-level language identification system[13] based on word frequencies and character n-grams (Word-level ID). We also report the result

[8] https://bn.wikipedia.org/wiki/
[9] https://scdnlab.com/corpus/
[10] http://wortschatz.uni-leipzig.de/
[11] https://github.com/x3r/deromanization
[12] https://github.com/guillaumegenthial/tf_ner
[13] https://github.com/eginhard/word-level-language-id

System	Bengali		Hindi	
	top-1	top-10	top-1	top-10
Sequitur	42.0	**82.7**	**43.6**	**89.9**
DTLM	48.8	70.2	42.7	82.7
OpenNMT	**61.0**	81.7	41.1	80.4

Table 5: Back-transliteration accuracy on the test sets.

System	Bengali		Hindi
	Dev	Test	Test
Sequitur	47.6	51.0	49.2
Our system	**78.2**	**79.8**	**84.3**
w/o language ID	69.6	50.5	61.6
Google Translate	77.1	60.4	64.4

Table 6: The results on deromanization of code-mixed texts (in % word accuracy).

of a majority baseline which classifies every token as non-English.

The results are shown in Table 3. Our encoder-decoder achieves the highest accuracy on all sets, with the CRF system close behind. The lower results of the n-gram based approach highlight the issue of the ambiguity introduced through romanization. Because the gold tags for the FIRE 2014 test sets are not publicly available, we are unable to directly compare to the systems that participated in that shared task; the best reported results were 90.5% on Bengali (Banerjee et al., 2014) and 87.9% on Hindi (Gupta et al., 2014a).

5.3 Back-Transliteration

Table 4 shows the impact of enhancing the NEWS 2018 training data with manually-annotated dataset described in Section 4.2. The result is a substantial improvement in accuracy of all three back-transliteration systems. The synthetic data generated with our mapping script further boosts the top-1 accuracy of the OpenNMT system by 4.7%, but the impact on Sequitur and DTLM is negligible.

The results on the test sets are shown in Table 5. All three systems obtain similar top-1 results on Hindi. The neural system has the best top-1 accuracy on Bengali, which we attribute to the use of the synthetic data for training. However, Sequitur achieves the best accuracy among the top-10 predictions on both languages. The substantial discrepancy between the results on the Bengali development and test sets (Tables 4 and 5) are due to their different domains. The development set is curated from social media messages which contain a higher degree of spelling variation that reflects non-standard language use, while the test set consists of queries from search engine logs.

5.4 Sequence Prediction

We evaluate two variants of our system: the complete system that incorporates all three modules, and a restricted variant without language identification, which indiscriminately deromanizes every input word. For the baseline, we concatenate the top-1 predictions of the Sequitur system. As we are unaware of another publicly-available system for deromanization of code-mixed texts, we compare to the output of Google Translate (Figure 2).[14]

The end results are presented in Table 6. Our complete system substantially outperforms Google Translate (GT) on both Bengali and Hindi test sets. Both GT and our restricted variant unconditionally deromanize all tokens regardless of their language origin. On average, 27% of errors made by GT on the test sets are due to deromanization of English words. GT outperforms our restricted variant, which we attribute to their vastly superior resources. These results highlight the importance of handling the code-mixing issue in the deromanization task.

For the reasons explained in Section 5.2, we are unable to directly compare to the systems that participated in the FIRE 2014 shared task. The best reported F-score results on the deromanization of transliterated search subtask were 7.3% for Bengali (Gupta et al., 2014a) and 30.4% for Hindi (Mukherjee et al., 2014). We attribute the superior results of our system to its ability to handle spelling variations found in romanized code-mixed texts.

5.5 Error Analysis

An example output of the proposed deromanization system is presented in Figure 3. The errors are marked in bold and red. Our system incorrectly classifies the words '1' and 'bar' as English. These errors can be attributed to the existence of words with the same spelling in English, as well as the presence of another English word 'only' in the context. Google Translate avoids making these er-

[14] Although Google Gboard performs word-level deromanization of a mobile keyboard input, it has no interface for automatically deromanizing large number of sentences.

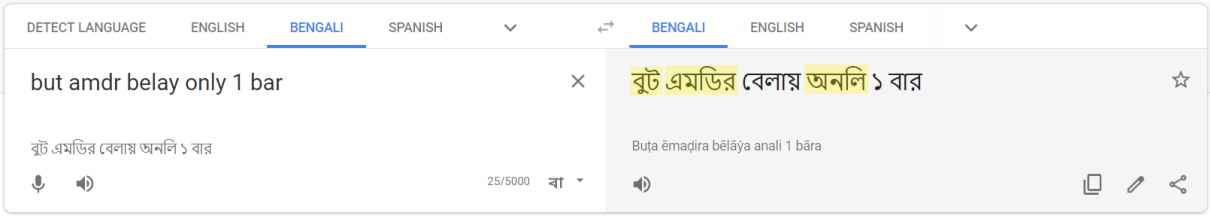

Figure 2: Google Translate interface. The errors are highlighted in yellow.[15]

(a)	but	amdr	belay	only	1	bar
(b)	E	B	B	E	B	B
(c)	but	আমাদের	বেলায়	only	১	বার
(d)	“but”	“our”	“case”	“only”	“one”	“time”
(e)			“but only once in our case”			
(f)	but	আমার	বেলায়	only	1	bar

Figure 3: An example system error: (a) original message; (b) implied language tags; (c) gold output; (d) word-level translation; (e) complete translation; (f) our system output.

rors by always deromanizing all tokens; however, this indiscriminate approach also produces bewildering Bengali forms that reflect the pronunciation of English function words ‘but’ and ‘only’ (Figure 2). In addition, both systems make deromanization errors on ‘amdr’, which is a contracted form of the Bengali word ‘amader’. This example illustrates the inherent difficulty of the task, and the importance of handling the code-mixing and deromanization together.

6 Conclusion

We have presented a joint approach for deromanization of code-mixed texts. The experiments on two languages show that our system achieves state-of-the-art results. In the future, we plan to apply our approach to other languages and scripts that involve both code-mixing and romanization.

References

Mohamed Al-Badrashiny, Ramy Eskander, Nizar Habash, and Owen Rambow. 2014. Automatic transliteration of romanized dialectal Arabic. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 30–38.

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. “I am borrowing ya mixing?” an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126.

Somnath Banerjee, Alapan Kuila, Aniruddha Roy, Sudip Kumar Naskar, Paolo Rosso, and Sivaji Bandyopadhyay. 2014. A hybrid approach for transliterated word-level language identification: Crf with post-processing heuristics. In *Proceedings of the Forum for Information Retrieval Evaluation*, pages 54–59. ACM.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the first workshop on computational approaches to code switching*, pages 13–23.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Kunal Chakma and Amitava Das. 2014. Revisiting automatic transliteration problem for code-mixed romanized Indian social media text. *Social-India 2014*, 2014:42.

Nancy Chen, Xiangyu Duan, Min Zhang, Rafael E. Banchs, and Haizhou Li. 2018. News 2018 whitepaper. In *Proceedings of the Seventh Named Entities Workshop*, pages 47–54. Association for Computational Linguistics.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identification using CRF: Code-switching shared task report of MSR India system. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 73–79.

Monojit Choudhury, Gokul Chittaranjan, Parth Gupta, and Amitava Das. 2014. Overview of FIRE 2014 track on transliterated search. *Proceedings of FIRE*, pages 68–89.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed Indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387. NLP Association of India.

[15] https://translate.google.com (accessed on Mar. 3, 2019)

Xiangyu Duan, Min Zhang, Haizhou Li, Rafael Banchs, and A Kumaran. 2016. Whitepaper of NEWS 2016 Shared Task on Machine Transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 49–57.

Deepak Kumar Gupta, Shubham Kumar, and Asif Ekbal. 2014a. Machine learning approach for language identification & transliteration. In *Proceedings of the Forum for Information Retrieval Evaluation*, pages 60–64. ACM.

Parth Gupta, Kalika Bali, Rafael E Banchs, Monojit Choudhury, and Paolo Rosso. 2014b. Query expansion for mixed-script information retrieval. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 677–686. ACM.

Lars Hellsten, Brian Roark, Prasoon Goyal, Cyril Allauzen, Françoise Beaufays, Tom Ouyang, Michael Riley, and David Rybach. 2017. Transliterated mobile keyboard input via weighted finite-state transducers. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2017)*, pages 10–19.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Ann Irvine, Jonathan Weese, and Chris Callison-Burch. 2012. Processing informal, romanized Pakistani text messages. In *Proceedings of the Second Workshop on Language in Social Media*, pages 75–78. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating joint n-gram features into a discriminative training framework. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 697–700. Association for Computational Linguistics.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1110–1119.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. *Proceedings of ACL 2017, System Demonstrations*, pages 67–72.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational linguistics*, 24(4):599–612.

Jonathan May, Yassine Benjira, and Abdessamad Echihabi. 2014. An Arabizi-English social media statistical machine translation system. In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas*, pages 329–341.

Abhinav Mukherjee, Anirudh Ravi, and Kaustav Datta. 2014. Mixed-script query labelling using supervised learning and ad hoc retrieval using sub word indexing. In *Proceedings of the Forum for Information Retrieval Evaluation*, pages 86–90. ACM.

Saeed Najafi, Colin Cherry, and Grzegorz Kondrak. 2018. Efficient sequence labeling with actor-critic training. *arXiv preprint arXiv:1810.00428*.

Garrett Nicolai, Saeed Najafi, and Grzegorz Kondrak. 2018. String transduction with target language models and insertion handling. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 43–53.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Abdul Rafae, Abdul Qayyum, Muhammad Moeenuddin, Asim Karim, Hassan Sajjad, and Faisal Kamiran. 2015. An unsupervised method for discovering lexical variations in Roman Urdu informal text. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 823–828.

Pallavi Verulkar, Rakesh Chandra Balabantray, and Rohit Arvind Chakrapani. 2015. Transliterated search of Hindi lyrics. *International Journal of Computer Applications*, 121(1).

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

Liang-Chih Yu, Wei-Cheng He, Wei-Nan Chien, and Yuen-Hsien Tseng. 2013. Identification of code-switched sentences and words using language modeling approaches. *Mathematical Problems in Engineering*, 2013.