Deromanization of Code-mixed Texts

by

Rashed Rubby Riyadh

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

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. Considering these two problems together is necessary to utilize the NLP resources and tools that are developed and trained on text corpora written in the standard form of the language. In this thesis, we propose a novel approach for handling these two problems together in a single system. Due to the unavailability of sufficiently large annotated resources for training an end-to-end approach, the proposed approach combines several supervised models for the three components: word-level language identification, back-transliteration, and sequence prediction. The results of the experiments on Bengali and Hindi datasets show that the proposed approach is substantially more accurate than Google Translate, and establish the state of the art for the task of deromanization of code-mixed texts.

Preface

The work presented in this thesis is a elaboration of the research article advised by Professor Greg Kondrak (Riyadh and Kondrak, 2019). The author was the main contributor who implemented different methods, and performed all the experiments.

Acknowledgements

I am deeply grateful to Professor Greg Kondrak for his supervision and guidance throughout the thesis. It would not be possible to overcome the steep challenges of performing graduate-level research in NLP without his help.

I would also like to thank Bradley Hauer and Saeed Najafi for the suggestions and feedback.

Finally, I am eternally indebted to my parents for their love and support. I cannot think of coming this far without them.

Contents

List of Tables

List of Figures

Chapter 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 1.1). The phenomenon is often observed in informal settings, such as social media, blog posts, news forums etc., and is due to several reasons, such as unavailability of a native-script keyboard, difficulties in writing in the native keyboard, minimizing the switching between English and native keyboard etc. 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 also lost in the romanization process due to the difficulty of representing native phonological distinctions in the Roman script. This makes *deromanizing* or converting such messages back to their native scripts 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 1.1). It can happen both within (inter-sentential) and between (intra-sentential) sentences. People may use code-mixing consciously or sub-consciously in their discourse for several reasons, such as comfort in expressing some concepts (such as technical terms) in a particular language, excluding people from a conversation, exerting influence on others etc. Code-mixing is particularly common in multi-lingual areas such as South Asia (Bali et al., 2014), Eastern Africa (Myers-Scotton, 1995), United States (Martínez, 2010) etc. Several

(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 1.1: 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.

studies have also been conducted to understand the extent of code-mixing in social media contents of specific locations. Rijhwani et al. (2017) show that non-English speaking cities produce a large number of code-mixed messages in the social media, such as 12% tweets in Istanbul and 8% tweets in Quebec City contain code-mixing. Bali et al. (2014) study the code-mixing phenomena in Indian social media and report 17% Facebook posts generated in India contain code-mixing.

Most prevalent form of code-mixing observed on the internet is in the same scripts (such as English-Spanish). Also, the languages involved in the phenomena are generally linguistically similar. Thus, analyzing these code-mixed texts are simpler. However, it is more difficult to process the texts when multiple scripts are employed or words from multiple languages are written in a single script. The trivial way to handle the romanized code-mixed texts is to convert every words to the native script. However, most of the English words have no equivalent conversions in the native script in many cases.

Transliterations have been commonly employed in the deromanization task. *Transliteration* is conversion of named entities and words between distinct scripts. *Back-transliteration*, on the other hand, is the conversion of transliterated named entities or words back to their original scripts. Though there can be several ways to transliterate a native word to other scripts, there is generally only one correct way to render the transliterated word to the native form (Knight and Graehl, 1998). Transliteration and back-transliteration are generally employed in machine translation and contextual cues are used to predict the correct transliterations. However, most of the works in deromanization perform the transliteration without considering the contextual information. Also, these deromanization systems generally consider only the top predictions, which might be incorrect in many cases due to spelling variations.

Transliterations and romanizations are closely related. Romanization can be considered a special form of transliteration, where words from any non-Latin script are converted to Latin script. As romanization involved only two distinct scripts, there are generally one or more standard romanization conventions. However, it is difficult and sometimes infeasible to come up with hand-crafted rules or standard mappings for transliterations due to a large number of possible scripts. So, transliteration systems are generally trained on a set of word pairs, which are transliterated by professional annotators. Transliteration mining can also be used to extract a large set of possible transliteration pairs from the parallel translation corpus.

In this thesis, 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, opinion mining, product recommendation in the social media. In addition, web-search queries are often expressed in a romanized form by speakers of languages that use native scripts, such as Arabic, Greek, and Hindi (Gupta et al., 2014b). Thus, deromanization of the search queries and the vast amount of web documents is required for providing the relevant search results which should match both native-script and code-mixed documents.

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. Several unsupervised and supervised approaches are proposed for language identification in code-mixed texts. However, only a few of these researches consider language identification in the code-mixed romanized scenario, which is a more difficult task. Workshops and shared tasks have also been devoted to code-mixing, including the problem of language identification

(Choudhury et al., 2014; Solorio et al., 2014; Molina et al., 2016).

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). The focus of these shared tasks is on the transliteration of names rather than dictionary words, which is performed without considering the word in context. Thus, directly transliterating the romanized words may not be ideal approach, as deromanization should take the context into consideration. 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. We show that this limitation leads to sub-optimal performance on deromanization.

In this thesis, 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 instead 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- and word-level language models, discriminative transduction, joint n-grams, and Hidden Markov Models (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 available tool that can be applied to this task.

The main statement of the thesis is:

> *Romanized messages that involve code-mixing can be more accurately recovered by jointly considering the phenomena of romanization and code-mixing through the combination of language identification, back-transliteration, and word sequence prediction.*

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

This thesis is organized as follows. In Chapter 2, we provide the literary works related to the thesis, which include the previous methods for identifying languages in code-mixed texts, and different approaches for addressing deromanization. Chapter 3 provides the necessary background for the proposed approach. It also introduces our approach for handling deromanization of code-mixed texts. In Chapter 4, we provide detailed experimental analysis of the proposed approach and show it's effectiveness on the task. Finally, Chapter 5 concludes the thesis and provides the discussion for future work.

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

Chapter 2

Literature Review

This chapter provides an overview of the two important tasks, namely language identification and deromanization. We will mainly focus on the recent progress of these respective tasks in the code-mixed setting. Additional prior work will also be presented as required in the next chapters.

2.1 Language Identification

Though the identification of language in a monolingual document is a well studied problem, identifying languages in multilingual texts has garnered lots of attention recently. In this section, the recent advancements of language identification in code-mixed texts will be covered.

King and Abney (2013) propose several weakly supervised systems for identifying languages in mixed-language documents. The methods are based on conditional random fields (CRF), Hidden Markov Model (HMM) and logistic regression that use several hand-crafted features. The authors show that the systems which consider the contextual information perform better than the independent language identification. The proposed approaches are evaluated on manually-annotated bilingual datasets that contain 30 languages. Among the evaluated approaches, the CRF-based approach is found to be most accurate. However, the use of hand-crafted features has been proven to be sub-optimal and replaced with word representations learned using neural approaches, which surpass the performance of the traditional feature-based approaches.

Nguyen and Doğruöz (2013) present several supervised approaches, which are guided by the language labels produced by an unsupervised approach, for language identification in multilingual conversational texts. The unsupervised approach uses dictionary and character n-gram features to predict the language labels. The labels are then used as features by the supervised approaches, namely logistic regression and CRFs. Additionally, the authors also perform experiments with the contextual features to evaluate how these features affect the identification accuracy. The approaches are evaluated on a manually created Dutch-Turkish dataset. Though both the CRF (97.6%) and LR (96.7%) with contextual features outperforms the methods without the features, the margin of improvement is low (.2% on average).

Yu et al. (2013) employ statistical language modeling for identifying code-switched sentences and the language labels in code-mixed document. Their supervised approach determines the probability of the next word being a code-switched word based on the previous n-words. The authors train two language models-one from code-switched sentences and another from non-code-switched sentences and compare the probability of a test sentence on these language models to determine whether a sentence is code-switched or not. The proposed approach is evaluated on *Sinica* corpus, which contains Mandarin-Taiwanese sentences and achieved 53.08% accuracy on the word-level language identification task. Though approach is able to successfully apply language models for language identification, the presence of ambiguous and informal words significantly deteriorates the performance as seen from the low-accuracy of the approach.

Chittaranjan et al. (2014) describe a CRF-based supervised approach for word-level language identification in code-mixed. Their approach uses various token-level and contextual features such as capitalization information, character n-gram, lexicon features etc. for training the CRF system. The approach is tested on the Computational Approaches to Code-switching shared task (Diab et al., 2014) in four language pairs, namely, English-Spanish (En-Es), English-Nepali (En-Ne), English-Mandarin (En-Cn), and Standard Arabic-Arabic (Ar-Ar), and achieves accuracy ranging from 80%-95%, except for Ar-Ar test set.

Barman et al. (2014) propose several unsupervised and supervised approaches for language identification of social media. The unsupervised dictionary-based approach makes use of several word lists to determine the origin of a word. The supervised approaches employ contextual features to train support vector machines (SVM) and CRFs. The experiments are performed on a manually collected and annotated corpus which contains words from Hindi, English and Bengali. Among the tested approaches, the CRF approach achieves the best result with 95.8% accuracy, with the SVM approach close behind.

Das and Gambäck (2014) perform a comparison among several unsupervised and supervised approaches for word-level language identification in code-mixed texts. Their unsupervised approaches are based on character n-gram and dictionary look-up. The authors use SVM that employs several hand-crafted features as the supervised approach. A post-processing technique is also proposed that performs error correction on the predicted label by considering the contextual information. The approaches are evaluated on a dataset manually created and annotated from several Facebook pages and groups. The experimental analysis shows that SVM-based supervised approach with post-processing achieves 94.4% and 91.9% accuracy on Hindi and Bengali, respectively, and outperforms both unsupervised approaches.

Sadat et al. (2014) employ character-level language modeling for Arabic dialect identification in social media. They propose two supervised approaches, namely Markov model and Naive Bayes classifier, which utilize character n-gram models (Uni-gram, Bi-gram and Tri-gram) built from training data in social media context. The approaches are tested using a data set collected from Arabic blogs and forums. Though the approaches achieve around 97% in dialect identification, it does not consider the mixing of dialects in the documents, and performs document-level classification.

Gella et al. (2014) introduce an interesting post-processing step on top of a supervised approach for word-level language identification in code-mixed texts. Their approach seeks to build a general language identification system which makes use a binary classifier such as Naive Bayes, Logistic Regression etc. which can employ character n-grams as features, and performs a post-processing step known as *CoverSet*. As the approach does not assume the languages of the text, the CoverSet chooses the minimum number of languages for labeling all the words in a text. The authors evaluate their approach on two datasets - FIRE dataset, and a synthetic dataset, and achieve 42% and 82% average accuracy, respectively.

Jaech et al. (2016) describe a supervised approach for language identification in social media. The approach is based on hierarchical character-word models, which employs character and contextualized word-level representations for the task. The character-level representation is built using a convolutional neural network (CNN) applied over the whitespace-limited words. A bi-directional LSTM recurrent neural network is used over the character representation to produce the word labels. The approach is evaluated on TweetLID and Tweet70 datasets, and achieves 76.1% and 91.2% F1-score, respectively.

An unsupervised approach based on Hidden Markov Models (HMMs) is proposed by Rijhwani et al. (2017) for language identification in code-mixed Twitter texts. The approach considers k-HMMs for k languages. The transition probabilities are generated by considering both the monolingual and code-switched word as probable states, and emission probabilities are produced from monolingual word language models. It then uses the HMM parameters to decode the most probable language label sequence using a Viterbi decoder. The authors train the model on corpora of seven languages collected from Twitter, and evaluate it on a manually curated test set. The proposed approach achieves on average 96.3% accuracy in a mixture of two languages.

An interesting idea of employing the encoding of characters for the language identification of Bengali-English code-mixed data is proposed by Mandal et al. (2018). Their supervised approaches are based on LSTM networks which employ two different encoding, namely character-based and root phone-based encoding for training. The character-based encoding is developed by replacing each character with an integer index, and the phone-based encoding is built by creating a mapping between n-gram language model tokens and Unicode pronunciation of Bengali words. The phones present in a word are then replaced by the corresponding index of these phones in the mapping. The encoding schemes are combined to create two ensemble models using stacking and threshold technique. The approach is evaluated on the data from two shared tasks, namely- ICON 16[1], ICON 17[2]. The stacking and threshold models achieve 91.8% and 92.3% accuracy on the test set, respectively. As the proposed approaches are based on character-level information and do not consider the context, they are not able to handle the spelling ambiguities in code-mixed data and predict the same label for similarly spelled words that are from different languages.

Several workshops and shared tasks are dedicated to the task of language identification in code-mixed texts, including FIRE shared task on Transliterated Search (Choudhury et al., 2014), Computational Approaches to Code Switching (Solorio et al., 2014; Molina et al., 2016) etc. FIRE shared task (Choudhury et al., 2014) holds the subtask of language identification in several code-mixed romanized South-Asian languages. There are 6 language pairs which contain a South-Asian language (Bengali, Hindi, Malayalam, Kannada, Tamil, Gujrati) mixed with English. A total of 18 teams participated in the subtask, who have mostly utilized character n-grams and token level features for supervised classifiers, such as SVMs and Naive Bayes etc. The best reported accuracy for the languages are - Bengali (90.5%), 96.3% (Gujrati), Hindi (87.9%), Kannada (68.1%), Malayalam (86.0%) and Tamil (98.6%).

Computational Approaches to Code Switching (Molina et al., 2016) conducts the shared task on language identification in code-switched data from two language pairs - Modern Standard Arabic - Dialectal Arabic (MSA-DA), and Spanish - English (SPA-ENG). A total of nine teams participate in the shared task. The most used approach for the task are based on CRFs, which utilize language models, case and contextual features. A few teams also employ deep learning techniques using Convolutional Neural Networks (CNNs), Long Short Term Memory (LSTMs) etc. The best reported F1-score for the language pairs are - SPA-ENG (91.3%) and MSA-DA (83.0%).

[1]http://ltrc.iiit.ac.in/icon2016/
[2]http://ltrc.iiit.ac.in/icon2017/

2.2 Deromanization

The problem of converting romanized texts to their native scripts has been studied on its own or as a sub-module in pipeline approaches for machine translation, mixed-script information retrieval etc. In this section, the recent works in deromanization will be presented.

Short Message Service (SMS) is a potential source of romanized texts due to the difficulty of typing in the native-script keyboard. A supervised approach is proposed for deromanizing the informal Urdu messages by Irvine et al. (2012). The supervised approach makes use of a Hidden Markov Model (HMM) to combine the candidates derived from a character-level transliteration model and a dictionary of automatically aligned words. The experiments are performed on a Urdu SMS corpus which is manually annotated using Mechanical Turk. The authors show that the approach surpasses a baseline deterministic approach (Buckwalter Arabic deterministic map) and achieves 51% word-level transliteration accuracy. However, deterministic mapping algorithms are not able to consider the variations in spelling and the rich contextual information. Due to the unavailability of the standard train-test splits, we are unable to compare our approach with the proposed method.

Chakma and Das (2014) employ several supervised approaches for the automatic transliteration of code-mixed social media texts. Their supervised approaches are based on joint source channel (JSC) which breaks down the source and target words into phoneme units to learn the mappings between the source and target phonemes using the contextual information. They also make use of International Phonetic Alphabet (IPA) to generate the mappings between source and target characters. They perform the experiments on the manually collected and annotated FIRE 2014 shared task corpus that contains words from English, Hindi and Bengali (Choudhury et al., 2014). The experimental analysis shows that IPA-based system (on average 80%) outperforms the JSC-based approaches (on average 70%) on both Bengali and Hindi. Due to the unavailability of the test corpus, we cannot directly compare our system with the proposed approach.

A supervised approach for converting Dialectical Arabic written in Latin script (Arabizi) to Arabic script is presented by Al-Badrashiny et al. (2014). The approach utilizes a character-level finite state transducer to generate transliteration candidates. A morphological analyzer is then used to filter the candidates and a language model to choose the output transliteration. The authors test their approach on two in-house datasets that contain Egyptian Arabic SMS written in Latin script. The approach is evaluated on a Egyptian Arabizi SMS corpus and achieves 74% word-level transliteration accuracy.

Similar to the above task, van der Wees et al. (2016) introduce a supervised Arabizi-to-Arabic transliteration approach to deromanize the Arabizi texts. The approach uses a character-based transliteration model, which is incorporated as a component in the pipeline for a Arabizi to English phrase-based machine translation system. The authors test their approach on NIST OpenMT evaluation campaign and achieve a comparable performance to human Arabic-Arbizi transliteration and translation. Though the approach does not consider the language identification of the texts and transliterates every word to Arabic, the authors mention that the language identification would improve the overall translation accuracy. As the datasets are LDC-licensed and only available to the researchers who participated in the NIST OpenMT evaluation campaign, we are unable to compare our system.

Verulkar et al. (2015) describe a supervised approach for searching the Hindi song lyrics using romanized queries. Their supervised approach employs a dictionary-based method for identifying the language origin and a mapping between the English and Hindi words for the transliteration. The approach is tested on the FIRE 2014 shared task on transliterated search (Choudhury et al., 2014), and achieves 83% language identification accuracy, and 82% transliteration F-score. Though the proposed approach achieves competitive performance on the respective tasks, the dictionary-based method is not able to identify and transliterate unknown words.

2.3 Summary

We have provided a detailed literature study on language identification and deromanization. Though the task of deromanization has been studied for a while, few studies consider it in the code-mixed setting, and most of these proposed approaches achieve sub-optimal performance. So, the task requires close attention and we aim at overcoming the issues in these approaches.

Chapter 3

Methodology

In this chapter, we describe our approach for converting romanized code-mixed texts to their native scripts. First, we provide some preliminary concepts related to our proposed approach. We then present our approach for handling the code-mixing and deromanization together in a single system.

3.1 Background

3.1.1 Statistical Language Modeling

Languages are ever-changing and evolve over time. There can be written thousand of rules to model the characteristics of a language, yet there will be some examples of language usage that do not conform to these rules. Statistical language modeling aims at estimating probability distribution of various linguistic units, such as - characters, words, sentences etc. Specifically, given a sentence *X*, the language model assigns a probability, *P(X)*, which represents how likely *X* is from a specific language. Statistical language modeling has been successfully used in various tasks, such as - language identification, speech recognition, machine translation, spell correction, decipherment etc.

The n-gram language model (LM) is based on the Markov assumption, where probability of a word, w_{n+1}, with its preceding context, w_n can be calculated by dividing the count of $w_n w_{n+1}$, by the count of w_n.

$$P(w_{n+1}) = \frac{count(w_n w_{n+1})}{\sum_w count(w_n w)}$$

In case of a uni-gram model, where the probability of the current word does not consider context, is the relative frequency of the word in a corpus. The bi-gram and tri-gram models consider a context of preceding one- and two-words, respectively. The probability of a sentence is calculated using chain rule, that is, by multiplying the likelihood of individual words in the sentence. We can get the probability of a n-word sentence,

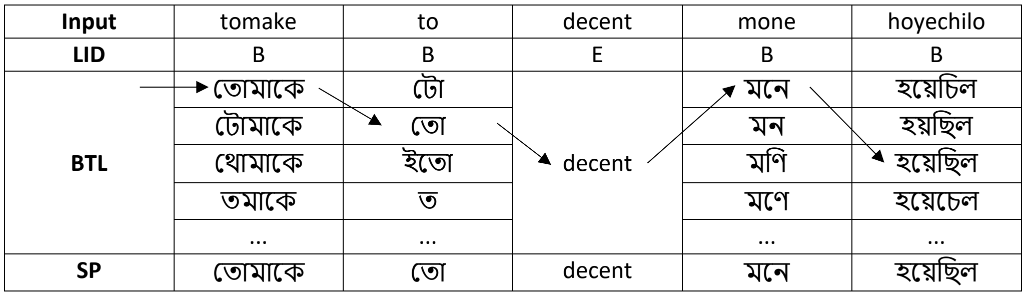
$$P({w^n}_1) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_n|w_1^{n-1})$$

Generally, n-gram order of 2 to 5 is used that can take a longer context into consider. However, some of the n-grams of a new sentence might not be present in the training corpus, which will result in the probability of the whole sentence being zero. To solve this problem, several smoothing and back-off techniques have been proposed. We have used the *deleted interpolation* smoothing technique in our approach, which will be discussed later.

3.1.2 Hidden Markov Model

Hidden Markov Model (HMM) is a statistical model where the system under consideration is assumed to be Markov process with unobserved states. In HMM, the states are not directly visible, however, the outputs or observations are visible. Each state has a probability distribution (*emission probabilities*) over the possible states it can emit. Also, there is a probability distribution (*transition probabilities*) over all the possible transitions of the states. HMM has been widely used in different application areas, particularly in reinforcement learning, speech recognition, parts-of-speech tagging etc.

For example, consider there is a chef's special dessert in a restaurant, which depends on the weather that day. Also, it is known that the chef prepares from a small set of desserts. Now, we can model the behavior of the chef using a HMM and predict, for example, what special dessert we can expect today. If we assume that the weather of each day solely depends on the previous day's weather, it will be a Markov chain. Then the probability of today's weather, provided the weather of yesterday will form the transition probability. Now, if



Input	tomake	to	decent	mone	hoyechilo
LID	B	B	E	B	B
BTL	তোমাকে	টো		মনে	হয়েচিল
	টোমাকে	তো		মন	হয়ছিল
	থোমাকে	ইতো	decent	মণি	হয়েছিল
	তমাকে	ত		মণে	হয়েচেল
	...	...		...	...
SP	তোমাকে	তো	decent	মনে	হয়েছিল

Figure 3.1: Overview of the proposed deromanization system.

we construct the probabilities for all the desserts for every weather condition, we will get the probability of preparing a dessert on a specific weather day. This probability distribution will form the emission probabilities. Now, we can employ the HMM to predict today's dessert using the Forward algorithm, or to determine the sequence of weather for last week from what was on the menu that week using the Viterbi algorithm.

3.2 Proposed Approach

Our approach consists of three main components: language identification (Section 3.2.1), back-transliteration (Section 3.2.2), and sequence prediction (Section 3.2.3). An overview of the proposed approach is presented in Figure 3.1.

The proposed approach takes a romanized code-mixed sentence as input and converts it to the native scripts. It first attempts to identify the languages present in the sentence using the language identification component, which assigns a language label to each of the tokens in the sentence and provides to the back-transliteration component. Based on the language label information, the transliteration systems generate transliteration candidates for the non-English words and keep the English words unchanged. Finally, the sequence prediction component chooses the best transliteration from the generated candidates using the prediction scores and a word-level language model.

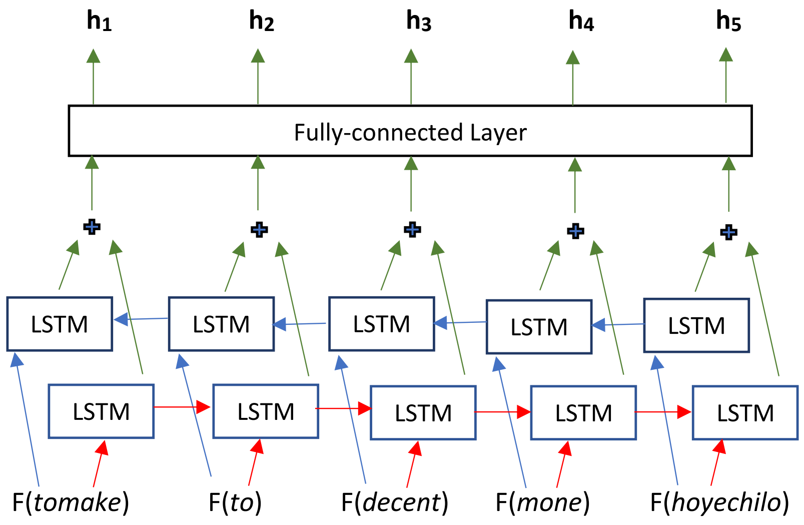


Figure 3.2: The encoder for labeling the sentence "tomake to decent mone hoyechilo". F retrieves the computed feature vectors. The symbol '+' denotes the concatenation of the forward (left-to-right) and backward (right-toleft) outputs.

3.2.1 Language Identification (LID)

Languages can be considered as a sequence of tokens, where there is dependency among the tokens. Recurrent neural networks (RNNs) have been successfully applied in many NLP tasks, such as language modeling (Mikolov et al., 2010), speech recognition (Graves et al., 2013), sentiment classification (Tang et al., 2015), image caption generation (Mao et al., 2014) etc., to capture the long-distance dependency in a sequence. Recently, neural sequence-to-sequence (seq2seq) models are introduced for tasks that take a sequence as an input and produce a sequence as output (Kalchbrenner and Blunsom, 2013; Cho et al., 2014). The encoder-decoder models are special variants of seq2seq models where the encoder employs an RNN to encode the input sequence into real valued vectors (hidden layers), and the decoder employs another RNN to decode the vectors into the target sequence.

We approach language identification as a sequence-to-sequence task, in which a sequence of word tokens is transformed into a sequence of language tags (Figure 3.1). Without a loss of generality, we assume that one of the two languages is English. Depending on the language label generated by this module, each input word is either fed into our back-transliteration module (Section 3.2.2) or copied unchanged to the final output.

Our supervised language identification module is based on the encoder-decoder model of Najafi et al. (2018a).[1] Though Huang et al. (2015) show that bi-directional Long Short Term Memory (LSTM) network with CRF layer can achieve state-of-the-art results on sequence labeling tasks such as part-of-speech tagging (POS), named entity recognition (NER) etc., we get superior performance from the encoder-decoder model on the language identification task. We attribute the superiority of the approach to either the nature of the task itself or the use of rich features in the encoder RNN.

The encoder architecture of the language identification module is presented in Figure 3.2. 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 each word's characters 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 character-embeddings help the model identify English words in the 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.

3.2.2 Back-Transliteration (BTL)

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 discrimina-

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

tive string transducer; and (c) Open-NMT, 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 3.2.3.

Sequitur

Sequitur (Bisani and Ney, 2008) is a data-driven transduction tool.[2] It trains a joint n-gram based model from the unlabeled data. The joint n-gram model is built on a language model over the operations used in the conversion from source to target, which allows the inclusion of source context in the generative model. Higher n-gram order models are trained iteratively from the lower order models. The alignment parameters between the source and target sequence are learned using Expectation Maximization (EM) algorithm. Sequitur was adopted as a baseline in the NEWS shared task on transliteration (Chen et al., 2018).

DTLM

DTLM is a new system that combines discriminative transduction with character and word language models derived from large unannotated corpora (Nicolai et al., 2018). DTLM is an extension of DirecTL+ (Jiampojamarn et al., 2010), whose target language modeling is limited to a set of binary n-gram features. Target language modeling is particularly important in low-data scenarios, where the limited transduction models often produce many ill-formed output candidates. We avoid the error propagation problem that is inherent in pipeline approaches by incorporating the LM feature sets directly into the transducer, which are based exclusively on the forms in the parallel training data. 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 as precision alignment. The idea is to

[2]https://github.com/sequitur-g2p/sequitur-g2p

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.

OpenNMT

As our neural transliteration system, we adopt the PyTorch variant of the OpenNMT tool (Klein et al., 2017).[3] The system employs an encoder-decoder architecture with an attention mechanism on top of the decoder RNN. The encoder encodes the source sentence into a fixed length representation. It can use character- and word-embedding as input, which is fed into the encoder RNN. The decoder then outputs a translation using the encoded hidden vector and the generated outputs in the previous steps. The attention mechanism helps the model to learn where to place attention on the input sequence as each word of the output sequence is decoded.

We insert word boundaries (whitespace) 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 bi-directional 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.

3.2.3 Sequence Prediction (SP)

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.

[3]https://github.com/OpenNMT/OpenNMT-py

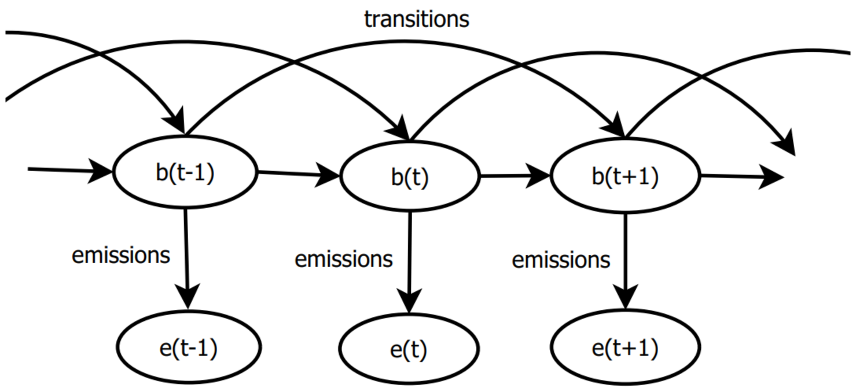


Figure 3.3: Modeling the sequence prediction system as HMM, where b(t) = hidden states and e(t) = observations.

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 (Figure 3.2.3). The emission probabilities are 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.[4]

As we employ three diverse transliteration systems, these systems predict n-best transliterations and prediction scores. We concatenate the individual system's output together to produce the final transliteration candidates. If any candidate is produced by more than one system, we chose the highest of the assigned prediction scores during concatenation. Finally, we normalize the scores of the final candidate list so that they represent a probability distribution.

The word-level language model helps the sequence prediction component to choose the words which are present in the vocabulary and maximizes the n-gram probability of the sequence. We train the unigram, bigram and trigram language models on a large monolingual corpus and combine the models using deleted interpolation. The smoothed word-trigram probability $\hat{P}$ is-

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

$$\hat{P}(b_k|b_{k-1}b_{k-2}) = \lambda_1 P(b_k) + \lambda_2 P(b_k|b_{k-1}) + \lambda_3 P(b_k|b_{k-1}b_{k-2})$$

,

such that the linear coefficients, λs sum to 1. The linear coefficients are calculated by successively deleting each trigram from the training corpus and maximizing the likelihood on the rest of the corpus. The probability of a sequence $s = b_1, b_2, b_3, ..., b_n$ according to the smoothed trigram language model is-

$$P(s) = \prod_{k=1}^{n} \hat{P}(b_k|b_{k-1}b_{k-2})$$

We use a modified *Viterbi* decoder to determine the most likely transliteration sequence from the generated candidates using both the prediction scores assigned by transliteration systems and transition scores derived from word-level language model (Algorithm 1). The scores are linearly combined to produce the score of a hidden sequence, s. If the transition score of s is $T(s)$ and the emission score is $E(s)$, then the score of s is derived from the following equation-

$$score(s) = \log T(s) + \log E(s) = \sum_{k=1}^{n} \hat{T}(b_k|b_{k-1}b_{k-2}) + \sum_{k=1}^{n} \hat{E}(e_k|b_k)$$

,

where $\hat{E}$ represents the probability of observing e_k from state b_k.

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 learned from the development set. So, the scoring function transforms to-

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

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

Algorithm 1: Modified Viterbi Algorithm

```
Input: Romanized sentence, $R$
       Emissions, $E$
       Transitions, $T$
Result: The most likely hidden state sequence, $S$
Initialize score matrix, $V$;
Initialize path back-pointer, $BP$;
begin
    /* Add sentence delimiters.                                  */
    $R =$ “$<s><s>$” $+ R +$ “$</s></s>$”;
    $l = length(R)$;
    for $i \in range(1, l-1)$ do
        $r_1 = R[i-1]$;
        $r_2 = R[i]$;
        $r_3 = R[i+1]$;
        for $w_1 \in E[r_1]$ do
            for $w_2 \in E[r_2]$ do
                for $w_3 \in E[r_3]$ do
                    $score = V(i, w_2) + [E(r_3, w_3)]^{p_e} + [T(w1, w2, w3)]^{p_t}$;
                    if $V(i+1, w_3) < score$ then
                        $V(i+1, w_3) = score$;
                        $BP(w_3) = w_2$
                    end
                end
            end
        end
    end
    /* Backtrace the most probable sequence.                     */
    Initialize decoded sequence, $S$;
    Initialize temporary array, $A$;
    $previous =$ “$</s>$”;
    while $BP(previous) \neq$ “$<s>$” do
        $A = A \cup BP(previous)$;
        $previous = BP(previous)$;
    end
    $S = reverse(A)$;
end
```

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.

3.3 Summary

We have presented a novel approach for deromanizing code-mixed texts in this chapter. Our proposed approach identifies the language labels of the provided romanized text and generates the transliteration candidates based on the language label information. Finally, the sequence prediction component combines the transliteration scores and the word-level language model scores to predict the most probable deromanization sequence from the transliteration candidates.

Chapter 4

Experiments

In this chapter, we present the results and analysis of the conducted experiments. In order to demonstrate the generality of our approach, we perform experiments on two languages: Bengali and Hindi. We also provide ablation study to show the effectiveness of the proposed approach.

4.1 Setup

We provide the experimental setup for all the individual components of the proposed approach in this section. The datasets for training different components are presented in Table 4.1.

4.1.1 Language Identification

The training data for the language identification shared task is collected from the track on transliterated search of the FIRE shared task (Choudhury et al., 2014). The Bengali and Hindi sets contain around 20.6k and 17.7k tokens, respectively. We hold out 10% of the data as a validation set. The language balance of the sets are presented in Table 4.2.

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 encoder and decoder model both employs a single layer of Long Short Term Memory (LSTM) network with 256 hidden units. The training is accomplished with Adam optimizer (Kingma and Ba, 2014), dropout regularization, and batch size of 64. The detailed hyper-parameter setting is

System	Dataset	Tokens	
		Bengali	Hindi
LI	FIRE 2014	20,660	17,756
TL	NEWS 2018	12,623	11,937
	Annotated	700	-
	Romanization	50,000	-
SP	Dev	1,990	-
	Test	539	3,287

Table 4.1: Datasets for the experiments.

Dataset	Bengali	English	Hindi	English
Dev	87.1	12.9	-	-
Test	68.3	31.7	75.0	25.0

Table 4.2: The language balance (in % of word tokens) in data sets.

provided in Table 4.3.

4.1.2 Back-Transliteration

The Bengali and Hindi training datasets for all three transliteration systems are from the NEWS 2018 shared task (Chen et al., 2018). The character language model and target word list for DTLM are built from publicly available unannotated corpora: Bengali Wikipedia[1], a Bengali news corpus[2], and a Hindi news corpus[3]. No additional data is used for the Hindi models.

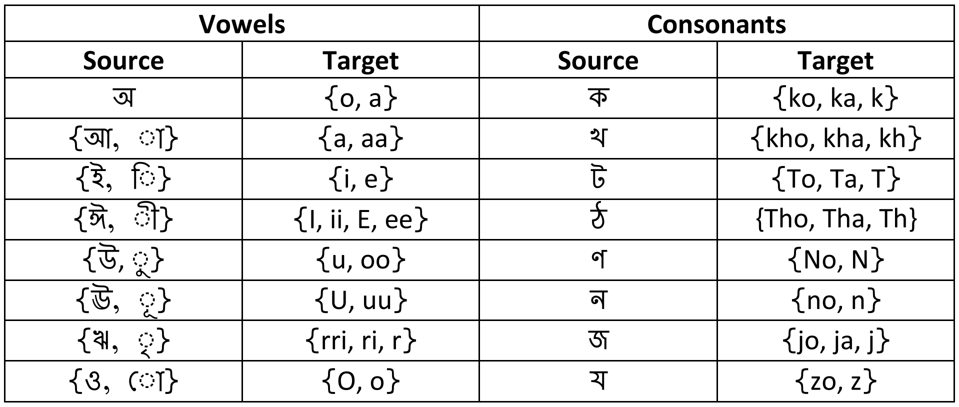
For Bengali back-transliteration, we experiment with leveraging language-specific expertise. First, since the training data contains mostly named entities, we augment it with manually-created transliterations of the most frequent 700 Bengali words from the news corpus. Second, since the performance of the neural system depends strongly on the amount of training data, we add romanizations of 50,000 Bengali words from Wikipedia.

The romanizations are generated with a context-free mapping from Bengali characters into Latin letters. A sample of the mapping is presented in Figure 4.1. As there are many ways to represent Bengali characters to Latin letters and some Bengali characters have different representations based on

[1]https://bn.wikipedia.org/wiki/
[2]https://scdnlab.com/corpus/
[3]http://wortschatz.uni-leipzig.de/

Hyper-parameter	Value
character embedding size	32
word embedding size	100
encoder hidden units	256
decoder hidden units	256
dropout	.5
batch size	32
learning rate	.0005

Table 4.3: The hyper-parameter setting for the language identification system.



Vowels		Consonants	
Source	**Target**	**Source**	**Target**
অ	{o, a}	ক	{ko, ka, k}
{আ, া}	{a, aa}	খ	{kho, kha, kh}
{ই, ি}	{i, e}	ট	{To, Ta, T}
{ঈ, ী}	{I, ii, E, ee}	ঠ	{Tho, Tha, Th}
{উ, ু}	{u, oo}	ণ	{No, N}
{ঊ, ূ}	{U, uu}	ন	{no, n}
{ঋ, ৃ}	{rri, ri, r}	জ	{jo, ja, j}
{ও, ো}	{O, o}	য	{zo, z}

Figure 4.1: Sample mapping from Bengali to English characters.

their position in a word, we allow multiple mappings for some of them. Each Bengali character is represented using on average 1.8, and maximum 3 Latin letters. We make both the mapping and romanization codes publicly available.[4] To evaluate the accuracy of the romanization program, we test it on the English-Bengali training dataset from the NEWS 2018 shared task. The program achieves 21.7% word-level and 78.9% character-level accuracy.

We perform a limited parameter tuning on our Bengali development set. 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. The architecture and hyper-parameters of OpenNMT system are also selected using grid-search. We found that using the LSTM as RNN gate performs better than Gated Recurrent Unit (GRU). The detailed hyper-parameter setting for each of the systems is provided in Table 4.4.

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

System	Hyper-parameter	Value
Sequitur	n-gram order	6
DTLM-aligner	maximum source sub-string	2
	maximum target sub-string	2
	maximization function	joint
DTLM-transducer	joint M-gram features	{1-5}
	n-gram size	{5-17}
	context-size	{2-8}
OpenNMT	encoder RNN gate	bi-LSTM
	decoder RNN gate	LSTM
	encoder/decoder hidden units	500
	encoder/decoder word-vector size	500
	encode/decoder hidden layers	2
	dropout	.5
	attention type	general

Table 4.4: The hyper-parameter settings for the back-transliteration systems.

4.1.3 Sequence Prediction

Due to the sparsity of annotated resources, we created our own Bengali development set. We collected romanized posts from several Facebook groups and pages, and manually deromanized them. It contains 247 sentences and 1990 words. As our test sets, we use the official development sets from transliterated search track of the FIRE 2014 shared task. The test sets of Bengali and Hindi contain 100 and 500 sentences, respectively. The test sets consist of transliterated search queries, which have a greater number of English words as shown in Table 4.2. Also, these data sets are not from the social media and do not contain complete sentences, which limit the sequence prediction system's ability to utilize the contexts.

4.2 Evaluation Metrics

We use standard evaluation metrics for measuring the performance of the components. We report the accuracy of the language identification component. For transliteration, we calculate the word-level Top-1 and Top-10 accuracy of the transliteration component. Finally, the performance of the sequence prediction component is measured in word-level accuracy.

System	Bengali		Hindi
	Dev	Test	Test
Naïve baseline	87.1	68.3	75.0
Dictionary-based	77.8	82.0	83.6
Bi-LSTM + CRF	92.4	90.9	93.2
Encoder-decoder	**95.2**	**92.2**	**95.3**

Table 4.5: Language identification accuracy (in %).

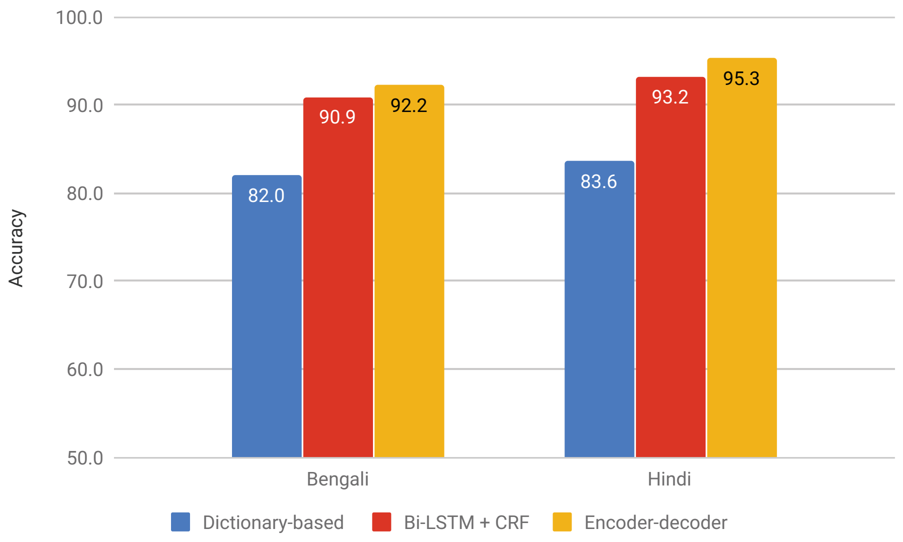


Figure 4.2: Language identification accuracy on test sets (in %).

4.3 Results

In this section, we present the results for each of the three components in turn.

4.3.1 Language Identification

We evaluate our encoder-decoder model based language identification system against a CRF-based sequence tagging model on top of recurrent neural networks (Huang et al., 2015). We adapt a general sequence tagging implementation[5] to the language identification task. A dictionary-based approach of Barman et al. (2014) serves as a baseline.

The results are shown in Table 4.5. Our encoder-decoder achieves the

[5]https://github.com/guillaumegenthial/tf_ner

System	Training 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 4.6: Impact of annotated data on the back-transliteration word accuracy of Bengali dev set (in %).

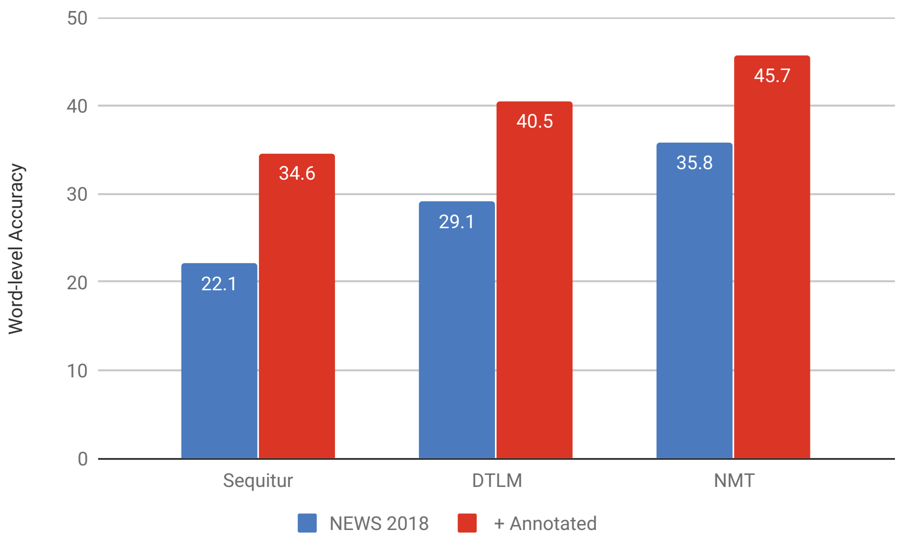


Figure 4.3: Impact of annotated data on the back-transliteration word accuracy (Top-1) of Bengali dev set (in %).

highest accuracy on all sets, with the CRF system close behind. The lower results of the dictionary-based approach highlight the issue of the ambiguity introduced through romanization. We also report the result of a Naïve baseline which classifies every token as non-English. Because the gold tags for the FIRE 2014 tests set are not publicly available, we are unable to directly compare to the systems that participated in the FIRE shared task; however, the best reported result on those Bengali and Hindi test sets were 90.5% (Banerjee et al., 2014) and 87.9% (Gupta et al., 2014a), respectively.

System	DirecTL+	Sequitur	OpenNMT	Linear Combination
Bengali	35.8	37.8	32.7	**40.7**
Hindi	**32.3**	30.3	29.4	32.2

Table 4.7: NEWS shared task results on NEWS test sets (in %).

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 4.8: Back-transliteration word accuracy on test sets (in %).

4.3.2 Back-Transliteration

First, we present the results of using the manually-annotated data for training the systems in Table 4.6. It is clearly evident that this additional data improves the accuracy of all the three systems, and accounts for 12.5% Top-1 and 14.3% Top-10 average accuracy improvement over the NEWS 2018 training data. Though the romanization data does not provide much improvements for Sequitur and DTLM, it further improves the OpenNMT system's Top-1 and Top-10 accuracy by 4.7% and 3.4%, respectively.

We also report our results of NEWS 2018 Shared Task on Transliteration in Table 4.7 (Najafi et al., 2018b). The main goal of the shared task is transliterating named entities between various scripts. Though the task is different from the problem we tackle in this thesis, the results will give an overview of the state of the art in the transliteration, which is a main component of our proposed approach. We employed 2 non-neural approaches, namely - DirecTL+ and Sequitur; 3 neural approaches, namely - OpenNMT, BaseNMT and RL-NMT; and, a linear combination of all the approaches in the shared task. The experiments were performed on 19 language pairs, that involve 14 languages. We only present the results of 4 systems on Bengali and Hindi test sets due to their relevance to this thesis. We also provide average accuracy of each system on all language pairs to show their overall performance. Among the individual systems, DirecTL+ achieves the best accuracy on both languages. The linear combination generally outperforms individual systems,

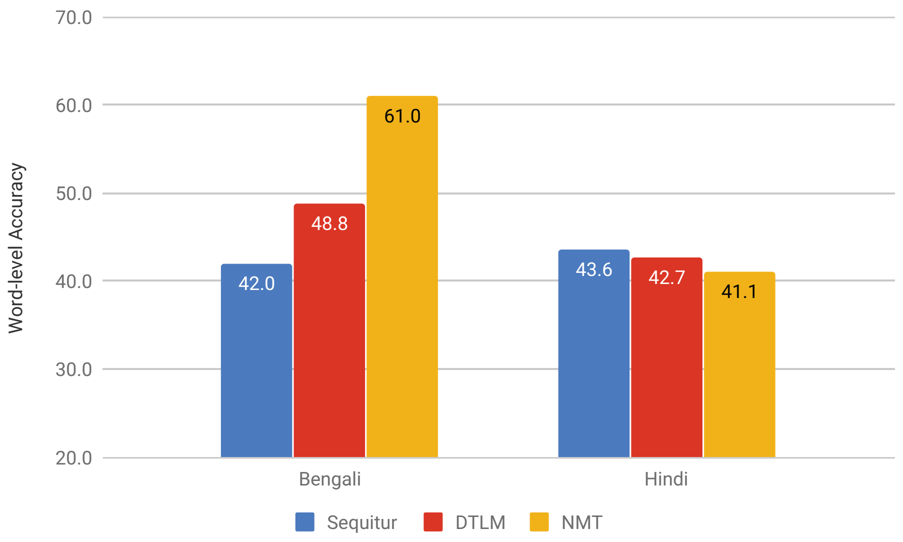


Figure 4.4: Back-transliteration word accuracy (Top-1) on test sets (in %).

except for Hindi, where it is closely beaten by DirecTL+. Overall, our team (UALB) is the only team to participate in all language pairs and achieves competitive results in most of them.

The back-transliteration results on the test sets are shown in Table 4.8. The neural system has by far 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 (e.g., 82.7% on the Bengali test set). All three systems obtain similar results on Hindi. Though DTLM outperforms both Sequitur and OpenNMT on the NEWS 2018 shared task on transliteration in all the evaluated language pairs, it falls behind Sequitur and OpenNMT on Bengali and Hindi datasets, respectively. We conjecture that the drop in performance can be due to the out-of-domain data for training. Also, the test set contains search queries and many of the words in those queries are not dictionary words. As a result, DTLM cannot use the target word list to guide its predictions. For example, Bengali dev set contains 60.7% words from the target word list, however, the test set contains 44.4% words from that list.

System	Bengali		Hindi
	Dev	Test	Test
Sequitur	47.6	51.0	49.2
Our approach	**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 4.9: Deromanization word accuracy (in %).

4.3.3 Sequence Prediction

We evaluate two variants of our system: the complete system that incorporates all three modules (our approach), and a restricted variant without language identification (w/o language ID), which attempts to deromanize every input word. For the baseline, we take the the top-1 prediction of the Sequitur system. As we are unaware of another publicly-available system for deromanization of code-mixed texts, we compare our complete system to the output of Google Translate, which incorporates a general approach for transliteration of romanized input to native scripts of several South Asian languages (Hellsten et al., 2017). An example of the Google Translate interface is provided in Figure 4.6.[6]

The end results are presented in Table 4.9. Our complete system substantially outperforms Google Translate on both Bengali and Hindi test sets. On average, 27% of errors made by Google Translate on test sets are due to deromanization of English words. Similar to our restricted variant, Google Translate unconditionally deromanizes all words regardless of whether they are native or English. Google Translate outperforms this restricted variant which has about 25% higher error rate than our complete system. The performance improvement of Google Translate in this setting can be attributed to the superior resources available to Google Translate. These results show the importance of handling the code-mixing issue in the deromanization task.

Similar to the language identification subtask, we are unable to directly compare our full system results with the systems participated in the FIRE 2014 shared task. The best reported F-score results of Bengali and Hindi on

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

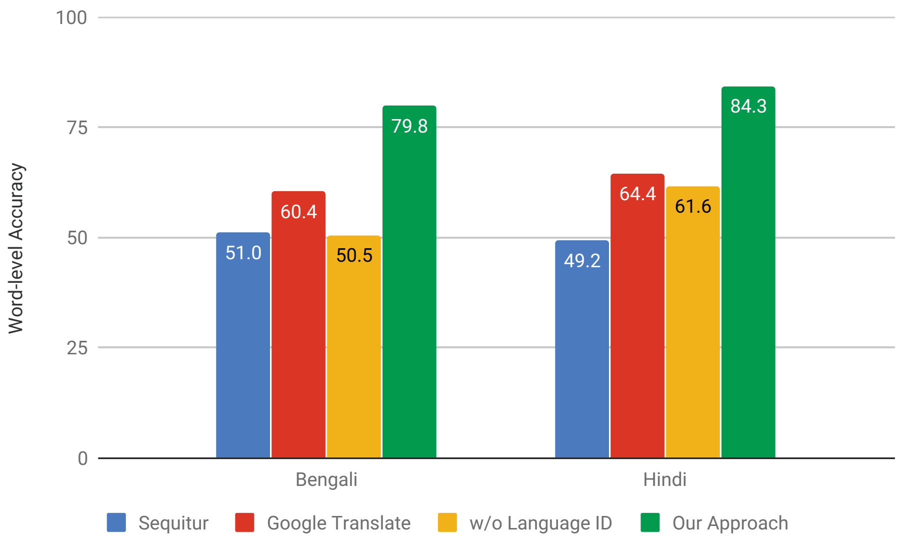
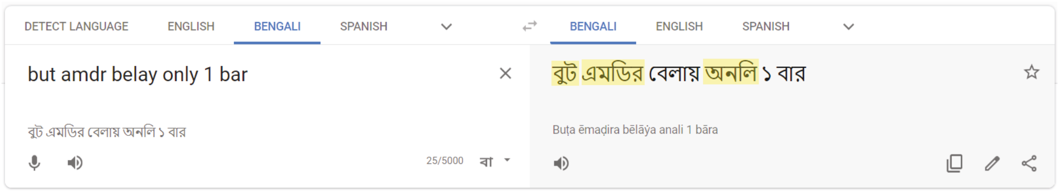


Figure 4.5: Deromanization word accuracy on test sets (in %).



but amdr belay only 1 bar

বুট এমডির বেলায় অনলি ১ বার

বুট এমডির বেলায় অনলি ১ বার

Buṭa ēmaḍira bēlāẏa anali 1 bāra

Figure 4.6: Google Translate interface for the transliteration, where the left side contains the original message and the right side contains the transliteration. The errors are highlighted in yellow.

the deromanization of transliterated search subtask are 7.3% (Gupta et al., 2014a) and 30.4% (Mukherjee et al., 2014), respectively. The low performance of these systems can be primarily attributed to their inability to handle the spelling variations found in the romanized code-mixed texts. However, our proposed approach can effectively handle these issues, which is evident from the superior results on both languages.

4.4 Error Analysis

A few examples of the proposed deromanization system is presented in Figure 4.4. The errors made by the language identification and back-transliteration

1(a)	but amdr belay only 1 bar
1(b)	but/E **আমার**/B বেলায়/B only/E **1/E bar/E**
1(c)	but/E আমার/B বেলায়/B only/E ১/B বার/B
1(d)	but only once in our case.
1(e)	**বুট/B এমডির**/B বেলায়/B **অনলি/B** ১/B বার/B
2(a)	sudhu 1 bar mon theke khuje dekho
2(b)	শুধু/B ১/B বার/B মন/B থেকে/B খুঁজে/B দেখো/B
2(c)	শুধু/B ১/B বার/B মন/B থেকে/B খুঁজে/B দেখো/B
2(d)	find out from the heart only once.
2(e)	শুধু/B ১/B বার/B মন/B থেকে/B খুঁজে/B দেখো/B
3(a)	but the way you affect me is weird
3(b)	but/E the/E way/E you/E affect/E me/E is/E weird/E
3(c)	but/E the/E way/E you/E affect/E me/E is/E weird/E
3(d)	but the way you affect me is weird
3(e)	**বুট/B টি/B ওয়ে/B ইউ/B এফেক্ট/B মে/B ইস/B বেয়ার্ড/B**

Figure 4.7: Examples of the proposed deromanization system: (a) original message; (b) system output with identified language labels; (c) gold output with language labels; (d) translation of the original message; (e) Google Translate output.

systems are in bold. In the first example, the language identification system identifies the language of words "1" and "bar" incorrectly as English. We conjecture that the mistake is due to these words' presence in English vocabulary and presence of another English word "only" in the context. This example shows the difficulties of identifying languages in the romanized code-mixed texts due to ambiguity. It is also observed that the word "amdr" is incorrectly transliterated, which is actually a contraction of the word "amader", where the *vowels* are omitted. Google Translate does not consider the language identification and thus, transliterates the English words "but" and "only" as shown in Figure 4.4. It also makes an error transliterating the word "amdr".

In the second example, the language identification system identifies all the words correctly. Although this example also contains the words “1” and “bar”, the system identifies them correctly this time. It supports our conjecture that context plays an important role in the language identification, and the presence of the Bengali word “shudhu” helps to determine the origin of the following words. The sequence prediction system chooses all the transliterations correctly for this sentence. Similarly, Google Translate is able to deromanize all the words correctly because there is no code-mixing involved in this case.

Finally, the third example involves an English sentence, which is correctly identified by our language identification system. As the language labels are English, the back-transliteration system keeps all the words unchanged. The sequence prediction system then copies the words to the output. The Google Translate fails completely in this case due to the absence of a language identification component, and transliterates all the words to Bengali. This example illustrates the importance of considering code-mixing and deromanization together.

4.5 Summary

We have provided the results of our experiments in this chapter. Our proposed approach shows that considering the code-mixing and deromanization together can accurately recover the romanized messages, and achieves the state-of-the-art results on the deromanization task.

Chapter 5

Conclusion

In this thesis, we have proposed an approach for addressing the task of deromanization in code-mixed texts. Though the individual tasks of code-mixing and deromanization have been addressed in the previous works, this is the first approach that considers both tasks together in a single system. The proposed approach identifies the native words in the romanized code-mixed texts using a language identification component and generates transliteration candidates and their prediction scores through three diverse back-transliteration systems. A sequence prediction component then predicts the most probable sequence for each given sentence utilizing the scores from the transliteration systems and a word-level language model. The experiments are performed on the datasets of two languages-Bengali and Hindi. The detailed experimental analysis and ablation study show that considering the tasks together leads to the state of the art results on the task, which outperforms the Google Translate system on all the datasets.

Though the proposed approach achieved state of the art results on the deromanization of code-mixed texts, further research can be performed in several directions. The experiments are conducted on two Indic languages, Bengali and Hindi. In order to evaluate the generality of the approach, we plan to apply it to other languages and scripts that involve both code-mixing and romanization. The sequence prediction component employs the word-level language model for predicting the most probable deromanization sequence using the contextual cues. However, the presence of English words between the native words makes it difficult to effectively utilize the language model and results in isolated predictions. One of the ways to address this issue can be translating the English words to the native language and converting the translated words to correct inflected forms to match the surrounding contexts. Furthermore, some English words might be borrowed to the native language, and there might be standard transliterations for those words. Incorporating the borrowing phenomena effectively in the proposed approach can be an interesting study. Finally, studies can be conducted on the impact of the proposed approach on end tasks, such as POS tagging, machine translation, sentiment analysis etc.

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. *ocial- ndia 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.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Monojit Choudhury, Gokul Chittaranjan, Parth Gupta, and Amitava Das. 2014. Overview of fire 2014 track on transliterated search. In *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.

Mona Diab, Julia Hirschberg, Pascale Fung, and Thamar Solorio. 2014. Proceedings of the first workshop on computational approaches to code switching. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

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.

Spandana Gella, Kalika Bali, and Monojit Choudhury. 2014. “ye word kis lang ka hai bhai?” testing the limits of word level language identification. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 368–377.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

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.

Aaron Jaech, George Mulcaire, Shobhit Hathi, Mari Ostendorf, and Noah A. Smith. 2016. Hierarchical character-word models for language identification. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 84–93. 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.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

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 M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. of Association for Computational Linguistics*.

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

Soumil Mandal, Sourya Dipta Das, and Dipankar Das. 2018. Language identification of Bengali-English code-mixed data using character & phonetic based lstm models. *arXiv preprint arXiv:1803.03859*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

Ramón Antonio Martínez. 2010. Spanglish as literacy tool: Toward an understanding of the potential role of Spanish-English code-switching in the development of academic literacy. *Research in the Teaching of English*, 45(2):124.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49.

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.

Carol Myers-Scotton. 1995. *Social motivations for codeswitching: Evidence from Africa*. Oxford University Press.

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

Saeed Najafi, Bradley Hauer, Rashed Rubby Riyadh, Leyuan Yu, and Grzegorz Kondrak. 2018b. Comparison of assorted models for transliteration. In *Proceedings of the Seventh Named Entities Workshop*, pages 84–88.

Dong Nguyen and A. Seza Doğruöz. 2013. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 857–862. Association for Computational Linguistics.

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.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1971–1982.

Rashed Rubby Riyadh and Grzegorz Kondrak. 2019. Joint approach to deromanization of code-mixed texts. *In Proceedings of Sixth Workshop on NLP for Similar Languages, Varieties and Dialects.*

Fatiha Sadat, Farnazeh Kazemi, and Atefeh Farzindar. 2014. Automatic identification of Arabic dialects in social media. In *Proceedings of the first international workshop on Social media retrieval and analysis*, pages 35–40. ACM.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

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.