CoMeT: Towards Code-Mixed Translation Using Parallel Monolingual Sentences

Devansh Gautam[†] **Prashant Kodali**[†] **Kshitij Gupta**[†] **Anmol Goel**[††]
Manish Shrivastava[†] **Ponnurangam Kumaraguru**[‡]
[†]International Institute of Information Technology Hyderabad
[‡]Indraprastha Institute of Information Technology Delhi
[††]Guru Gobind Singh Indraprastha University, Delhi
{devansh.gautam,prashant.kodali,kshitij.gupta}@research.iiit.ac.in,
agoel00@gmail.com, m.shrivastava@iiit.ac.in, pk@iiitd.ac.in

Abstract

Code-mixed languages are very popular in multilingual societies around the world, yet the resources lag behind to enable robust systems on such languages. A major contributing factor is the informal nature of these languages which makes it difficult to collect code-mixed data. In this paper, we propose our system for Task 1 of CACLS 2021[1] to generate a machine translation system for English to Hinglish in a supervised setting. Translating in the given direction can help expand the set of resources for several tasks by translating valuable datasets from high resource languages. We propose to use mBART, a pre-trained multilingual sequence-to-sequence model, and fully utilize the pre-training of the model by transliterating the roman Hindi words in the code-mixed sentences to Devanagri script. We evaluate how expanding the input by concatenating Hindi translations of the English sentences improves mBART's performance. Our system gives a BLEU score of 12.22 on test set. Further, we perform a detailed error analysis of our proposed systems and explore the limitations of the provided dataset and metrics.

1 Introduction

Code-mixing[2] is the mixing of two or more languages where words from different languages are interleaved with each other in the same conversation. It is a common phenomenon in multilingual societies across the globe. In the last decade, due to the increase in the popularity of social media and various online messaging platforms, there has been an increase in various forms of informal writing, such as emojis, slang, and the usage of code-mixed languages.

Due to the informal nature of code-mixing, code-mixed languages do not follow a prescriptively defined structure, and the structure often varies with the speaker. Nevertheless, some linguistic constraints (Poplack, 1980; Belazi et al., 1994) have been proposed that attempt to determine how languages mix with each other.

Given the increasing use of code-mixed languages by people around the globe, there is a growing need for research related to code-mixed languages. A significant challenge to research is that there are no formal sources like books or news articles in code-mixed languages, and studies have to rely on sources like Twitter or messaging platforms. Another challenge with Hinglish, in particular, is that there is no standard system of transliteration for Hindi words, and individuals provide a rough phonetic transcription of the intended word, which often varies with individuals.

In this paper, we describe our systems for Task 1 of CALCS 2021, which focuses on translating English sentences to English-Hindi code-mixed sentences. The code-mixed language is often called *Hinglish*. It is commonly used in India because many bilingual speakers use both Hindi and English frequently in their personal and professional lives. The translation systems could be used to augment datasets for various Hinglish tasks by translating datasets from English to Hinglish. An example of a Hinglish sentence from the provided dataset (with small modifications) is shown below:

- **Hinglish Sentence:** Bahut strange choice thi ye.
- **Gloss of Hinglish Sentence:** Very [strange choice] was this.
- **English Sentence:** This was a very strange choice.

We propose to fine-tune mBART for the given task by first transliterating the Hindi words in the

[1] https://code-switching.github.io/2021

[2] Code-switching is another term that slightly differs in its meaning but is often used interchangeably with code-mixing in the research community. We will also be following the same convention and use both the terms interchangeably in our paper.

Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, pages 47–55

https://doi.org/10.26615/978-954-452-056-4_007

target sentences from Roman script to Devanagri script to utilize its pre-training. We further translate the English input to Hindi using pre-existing models and show improvements in the translation using parallel sentences as input to the mBART model. The code for our systems, along with error analysis, is public[3].

The main contributions of our work are as follows:

- We explore the effectiveness of fine-tuning mBART to translate to code-mixed sentences by utilizing the Hindi pre-training of the model in Devanagri script. We further explore the effectiveness of using parallel sentences as input.
- We propose a normalized BLEU score metric to better account for the spelling variations in the code-mixed sentences.
- Along with BLEU scores, we analyze the code-mixing quality of the reference translations along with the generated outputs and propose that for assessing code-mixed translations, measures of code-mixing should be part of evaluation and analysis.

The rest of the paper is organized as follows. We discuss prior work related to code-mixed language processing, machine translation, and synthetic generation of code-mixed data. We describe our translation systems and compare the performances of our approaches. We discuss the amount of code-mixing in the translations predicted by our systems and discuss some issues present in the provided dataset. We conclude with a direction for future work and highlight our main findings.

2 Background

Code-mixing occurs when a speaker switches between two or more languages in the context of the same conversation. It has become popular in multilingual societies with the rise of social media applications and messaging platforms.

In attempts to progress the field of code-mixed data, several code-switching workshops (Diab et al., 2014, 2016; Aguilar et al., 2018b) have been organized in notable conferences. Most of the workshops include shared tasks on various of the language understanding tasks like language identification (Solorio et al., 2014; Molina et al., 2016), NER (Aguilar et al., 2018a; Rao and Devi, 2016), IR (Roy et al., 2013; Banerjee et al., 2018), PoS tagging (Jamatia et al., 2016), sentiment analysis (Patra et al., 2018; Patwa et al., 2020), and question answering (Chandu et al., 2018).

Although these workshops have gained traction, the field lacks standard datasets to build robust systems. The small size of the datasets is a major factor that limits the scope of code-mixed systems.

Machine Translation refers to the use of software to translate text from one language to another. In the current state of globalization, translation systems have widespread applications and are consequently an active area of research.

Neural machine translation has gained popularity only in the last decade, while earlier works focused on statistical or rule-based approaches. Kalchbrenner and Blunsom (2013) first proposed a DNN model for translation, following which transformer-based approaches (Vaswani et al., 2017) have taken the stage. Some approaches utilize multilingual pre-training (Song et al., 2019; Conneau and Lample, 2019; Edunov et al., 2019; Liu et al., 2020); however, these works focus only on monolingual language pairs.

Although a large number of multilingual speakers in a highly populous country like India use English-Hindi code-mixed language, only a few studies (Srivastava and Singh, 2020; Singh and Solorio, 2018; Dhar et al., 2018) have attempted the problem. Enabling translation systems in the following pair can bridge the communication gap between several people and further improve the state of globalization in the world.

Synthetic code-mixed data generation is a plausible option to build resources for code-mixed language research and is a very similar task to translation. While translation focuses on retaining the meaning of the source sentence, generation is a simpler task requiring focus only on the quality of the synthetic data generated.

Pratapa et al. (2018) started by exploring linguistic theories to generate code-mixed data. Later works attempt the problem using several approaches including Generative Adversarial Networks (Chang et al., 2019), an encoder-decoder framework (Gupta et al., 2020), pointer-generator networks (Winata et al., 2019), and a two-level

[3] https://github.com/devanshg27/cm_translation

	Train	Valid	Test
# of sentences	8,060	942	960
# of tokens in source sentences	98,080	12,275	12,557
# of tokens in target sentences	101,752	12,611	-
# of Hindi tokens in target sentences	68,054	8,310	-
# of English tokens in target sentences	21,502	2,767	-
# of 'Other' tokens in target sentences	12,196	1,534	-

Table 1: The statistics of the dataset. We use the language tags predicted by the CSNLI library[4]. Since the target sentences of the test set are not public, we do not provide its statistics.

variational autoencoder (Samanta et al., 2019). Recently, Rizvi et al. (2021) released a tool to generate code-mixed data using parallel sentences as input.

3 System Overview

In this section, we describe our proposed systems for the task, which use mBART (Liu et al., 2020) to translate English to Hinglish.

3.1 Data Preparation

We use the dataset provided by the task organizers for our systems, the statistics of the datasets are provided in Table 1. Since the target sentences in the dataset contain Hindi words in Roman script, we use the CSNLI library[4] (Bhat et al., 2017, 2018) as a preprocessing step. It transliterates the Hindi words to Devanagari and also performs text normalization. We use the provided train:validation:test split, which is in the ratio 8:1:1.

3.2 Model

We fine-tune mBART, which is a multilingual sequence-to-sequence denoising auto-encoder pre-trained using the BART (Lewis et al., 2020) objective on large-scale monolingual corpora of 25 languages including English and Hindi. It uses a standard sequence-to-sequence Transformer architecture (Vaswani et al., 2017), with 12 encoder and decoder layers each and a model dimension of 1024 on 16 heads resulting in ~680 million parameters. To train our systems efficiently, we prune mBART's vocabulary by removing the tokens which are not present in the provided dataset or the dataset released by Kunchukuttan et al. (2018) which contains 1,612,709 parallel sentences for English and Hindi.

We compare the following two strategies for fine-tuning mBART:

- **mBART-en:** We fine-tune mBART on the train set, feeding the English sentences to the encoder and decoding Hinglish sentences. We use beam search with a beam size of 5 for decoding.
- **mBART-hien:** We fine-tune mBART on the train set, feeding the English sentences along with their parallel Hindi translations to the encoder and decoding Hinglish sentences. For feeding the data to the encoder, we concatenate the Hindi translations, followed by a separator token '##', followed by the English sentence. We use the Google NMT system[5] (Wu et al., 2016) to translate the English source sentences to Hindi. We again use beam search with a beam size of 5 for decoding.

3.3 Post-Processing

We transliterate the Hindi words in our predicted translations from Devanagari to Roman. We use the following methods to transliterate a given Devanagari token (we use the first method which provides us with the transliteration):

1. When we transliterate the Hindi words in the target sentences from Roman to Devanagari (as discussed in Section 3.1), we store the most frequent Roman transliteration for each Hindi word in the train set. If the current Devanagari token's transliteration is available, we use it directly.
2. We use the publicly available Dakshina Dataset (Roark et al., 2020) which has 25,000 Hindi words in Devanagari script along with their attested romanizations. If the current Devanagari token is available in the dataset, we use the transliteration with the maximum number of attestations from the dataset.
3. We use the `indic-trans` library[6] (Bhat et al., 2015) to transliterate the token from Devanagari to Roman.

4 Experimental Setup

4.1 Implementation

We use the implementation of mBART available in the fairseq library[7] (Ott et al., 2019). We fine-tune on 4 Nvidia GeForce RTX 2080 Ti GPUs

[4] https://github.com/irshadbhat/csnli

[5] https://cloud.google.com/translate

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

[7] https://github.com/pytorch/fairseq

Model	Validation Set		Test Set	
	BLEU	$\text{BLEU}_{\text{normalized}}$	BLEU	$\text{BLEU}_{\text{normalized}}$
mBART-en	**15.3**	18.9	**12.22**	–
mBART-hien	14.6	**20.2**	11.86	–

Table 2: Performance of our systems on the validation set and test set of the dataset. Since the target sentences of the test set are not public, we do not calculate the scores ourselves. We report the BLEU scores of our systems on the test set from the official leader board.

with an effective batch size of 1024 tokens per GPU. We use the Adam optimizer ($\epsilon = 10^{-6}, \beta_1 = 0.9, \beta_2 = 0.98$) (Kingma and Ba, 2015) with 0.3 dropout, 0.1 attention dropout, 0.2 label smoothing and polynomial decay learning rate scheduling. We fine-tune the model for 10,000 steps with 2,500 warm-up steps and a learning rate of $3 * 10^{-5}$. We validate the models for every epoch and select the best checkpoint based on the best BLEU score on the validation set. To train our systems efficiently, we prune mBART's vocabulary by removing the tokens which are not present in any of the datasets mentioned in the previous section.

4.2 Evaluation Metrics

We use the following two evaluation metrics for comparing our systems:

1. **BLEU:** The BLEU score (Papineni et al., 2002) is the official metric used in the leader board. We calculate the score using the SacreBLEU library[8] (Post, 2018) after lowercasing and tokenization using the `TweetTokenizer` available with the NLTK library[9] (Bird et al., 2009).

2. **$\text{BLEU}_{\text{normalized}}$:** Instead of calculating the BLEU scores on the texts where the Hindi words are transliterated to Roman, we calculate the score on texts where Hindi words are in Devanagari and English words in Roman. We transliterate the target sentences using the CSNLI library and we use the outputs of our system before performing the post-processing (Section 3.3). We again use the SacreBLEU library after lowercasing and tokenization using the `TweetTokenizer` available with the NLTK library.

[8] https://github.com/mjpost/sacrebleu
[9] https://www.nltk.org/

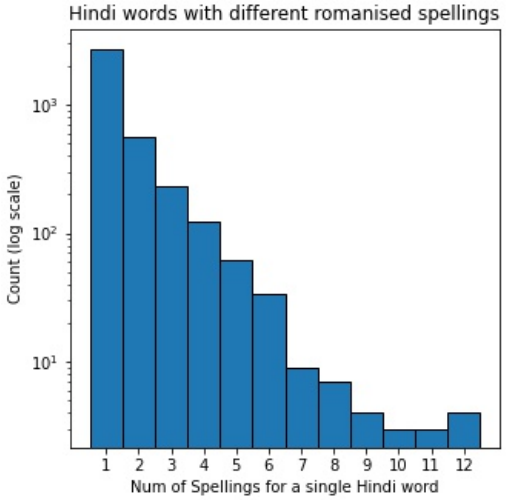

Figure 1: Multiple roman spellings for the same Hindi Word. These spelling variations can cause the BLEU score to be low, even if the correct Hindi word is predicted.

5 Results

Table 2 shows the BLEU scores of the outputs generated by our models described in Section 3.2. In Hinglish sentences, Hindi tokens are often transliterated to roman script, and that results in spelling variation. Since BLEU score compares token/n-gram overlap between source and target, lack of canonical spelling for transliterated words, reduces BLEU score and can mischaracterize the quality of translation. To estimate the variety in roman spellings for a Hindi word, we perform normalization by back transliterating the Hindi words in a code-mixed sentence to Devanagari and aggregated the number of different spellings for a single Devanagari token. Figure 1 shows the extent of this phenomena in the dataset released as part of this shared task, and it is evident that there are Hindi words that have multiple roman spellings. Thus, even if the model is generating the correct Devanagari token, the BLEU scores will be understated due to the spelling variation in the transliterated reference sentence. By back-transliterating Hindi tokens to Devanagari, **$\text{BLEU}_{\text{normalized}}$** score thus provides a better representation of translation quality.

5.1 Error Analysis of Translations of Test set

Since BLEU score primarily look at n-gram overlaps, it does not provide any insight into the quality of generated output or the errors therein. To

	mBART-en	mBART-hien
Mistranslated/Partially Translated	28	23
MWE/NER mistranslation	7	4
Morphology/Case Marking/Agreement/Syntax Issues	13	2
No Error	52	71

Table 3: Error Analysis of 100 randomly sampled translations from test set for both mBART-en and mBART-hien model

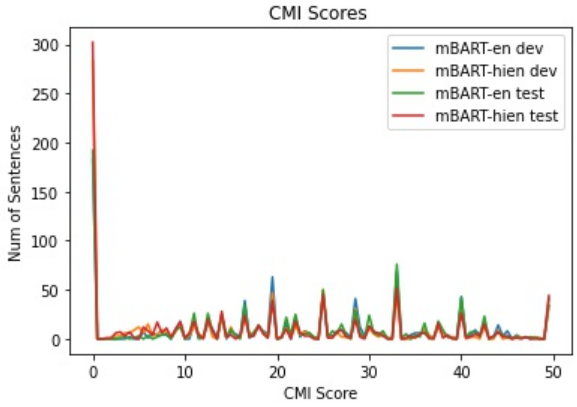

Figure 2: Code Mixing Index(CMI) for the generated translation of dev and test set .

	Avg CMI Score	% of Sents.with CMI = 0
Train Gold	19.4	26.1%
Dev Gold	21.6	19.3%
mBART-en Dev	21.8	19.4%
mBART-hien Dev	16.9	30.0%
mBART-en Test	21.8	20.0%
mBART-hien Test	16.7	31.4%

Table 4: Avg. CMI scores, Percentage of sentences with CMI = 0. Train Gold and Dev Gold are calculated on the target sentences given in the dataset. Rest are calculated on the outputs generated by our models.

	Validation Set	Test Set
mBART-en		
# of English tokens	3,282 (25.5%)	3,571 (27.6%)
# of Hindi tokens	8,155 (63.4%)	8,062 (62.3%)
# of 'Other' tokens	1,435 (11.1%)	1,302 (10.1%)
mBART-hien		
# of English tokens	2,462 (18.5%)	2,519 (18.8%)
# of Hindi tokens	9,471 (71.3%)	9,616 (72.0%)
# of 'Other' tokens	1,356 (10.2%)	1,233 (9.2%)

Table 5: The number of tokens of each language in our predicted translations. The language tags are based on the script of the token.

analyse the quality of translations on the test set, we randomly sampled 100 sentences (> 10% of test set) from the outputs generated by the two models: **mBART-en** and **mBART-hien**, and bucketed them into various categories. Table 3 shows the categories of errors and their corresponding frequency. Mistranslated/partially translated category indicates that the generated translation has no or very less semantic resemblance with the source sentence. Sentences, where Multi-Word Expressions/Named Entities are wrongly translated, is the second category. Morphology/Case Marking/Agreement/Syntax Issues category indicates sentences where most of the semantic content is faithfully captured in the generated output. However, the errors on a grammatical level render the output less fluent. **mBART-hien** makes fewer errors when compared to **mBART-en**, but that can possibly be attributed to the fact that this model generates a higher number of Hindi tokens while being low in code-mixing quality, and makes lesser grammatical errors. A more extensive and fine-grained analysis of these errors will undoubtedly help improve the models' characterization, and we leave it for future improvements.

5.2 Code Mixing Quality of generated translations

In the code-mixed machine translation setting, it is essential to observe the quality of the code-mixing in the generated translations. While BLEU scores indicate how close we are to the target translation in terms of n-gram overlap, a measure like Code-Mixing Index (CMI) (Gambäck and Das, 2016) provides us means to assess if the generated output is a mix of two languages or not. Relying on just the BLEU score for assessing translations can misrepresent the quality of translations, as models could generate monolingual outputs and still have a basic BLEU score due to n-gram overlap. If a measure of code mixing intensity, like CMI, is also part of the evaluation regime, we would be able to assess the code mixing quality of generated outputs as well. Figure 2 shows us that the distribution of CMI for outputs generated by our various models (mBART-en and mBART-hien) for both validation and test set.

Figure 2 and Table 4 show that the code mixing quality of the two models is is more or less similar across the validation and test set. The high

	Num of Pairs
Meaning of target similar to source	759
Meaning of target distored compared to source	141
Total	900

Table 6: Statistics of the errors in randomly sampled subset of train + dev.

percentages of sentences having a 0 CMI score shows that in a lot of sentences, the model does not actually perform code-mixing. We also find that even though the outputs generated by the **mBART-hien** model have a higher $\text{BLEU}_{\text{normalized}}$ score, the average CMI is lower and the percentage of sentences with a 0 CMI score is higher. This suggests that **mBART-hien** produces sentences with a lower amount of code-mixing. This observation, we believe, can be attributed to the **mBART-hien** model's propensity to generate a higher percentage of Hindi words, as shown in Table 5. We also find that in the train set, more than 20% of the sentences have a CMI score of 0. Replacing such samples with sentence pairs with have a higher degree of code mixing will help train the model to generate better code mixed outputs. Further analysis using different measures of code-mixing can provide deeper insights. We leave this for future work.

5.3 Erroneous Reference Translations in the dataset

We randomly sampled ~10% (900 sentence pairs) of the parallel sentences from the train and validation set and annotated them for translation errors. For annotation, we classified the sentence pairs into one of two classes : 1) Error - semantic content in the target is distorted as compared to source; 2) No Error - semantic content of source and target are similar and the target might have minor errors. Minor errors in translations that are attributable to agreement issues, case markers issues, pronoun errors etc were classified into the No Error bucket. Out of the 900 samples that were manually annoatated, 141 samples, i.e 15% of annotated pairs, had targets whose meaning was distorted as compared to source sentence. One such example is shown below:

- **English Sentence:** I think I know the football player it was based on.
- **Hinglish Sentence:** Muje lagtha ki yeh football player ke baare mein hein.
- **Translation of Hinglish Sentence:** I thought that this is about football player.

Table 6 shows the analysis of these annotated subset. The annotated file with all 900 examples can be found in our code repository. Filtering such erroneous examples from training and validation datasets, and augmenting the dataset with better quality translations will certainly help in improving the translation quality.

6 Discussion

In this paper, we presented our approaches for English to Hinglish translation using mBART. We analyse our model's outputs and show that the translation quality can be improved by including parallel Hindi translations, along with the English sentences, while translating English sentences to Hinglish. We also discuss the limitations of using BLEU scores for evaluating code-mixed outputs and propose using $\text{BLEU}_{\text{normalized}}$ - a slightly modified version of BLEU. To understand the code-mixing quality of the generated translations, we propose that a code-mixing measure, like CMI, should also be part of the evaluation process. Along with the working models, we have analysed the model's shortcomings by doing error analysis on the outputs generated by the models. Further, we have also presented an analysis on the shared dataset : percentage of sentences in the dataset which are not code-mixed, the erroneous reference translations. Removing such pairs and replacing them with better samples will help improve the translation quality of the models.

As part of future work, we would like to improve our translation quality by augmenting the current dataset with parallel sentences with a higher degree of code-mixing and good reference translations. We would also like to further analyse the nature of code-mixing in the generated outputs, and study the possibility of constraining the models to generated translations with a certain degree of code-mixing.

References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018a. Named entity recognition on code-switched data: Overview of the CALCS 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147, Melbourne, Australia. Association for Computational Linguistics.

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Thamar Solorio, Mona Diab, and Julia Hirschberg, editors. 2018b. *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*. Association for Computational Linguistics, Melbourne, Australia.

Somnath Banerjee, Kunal Chakma, Sudip Kumar Naskar, Amitava Das, Paolo Rosso, Sivaji Bandyopadhyay, and Monojit Choudhury. 2018. Overview of the mixed script information retrieval (msir) at fire-2016. In *Text Processing*, pages 39–49, Cham. Springer International Publishing.

Hedi M. Belazi, Edward J. Rubin, and Almeida Jacqueline Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic Inquiry*, 25(2):221–237.

Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. 2017. Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 324–330, Valencia, Spain. Association for Computational Linguistics.

Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. 2018. Universal Dependency parsing for Hindi-English code-switching. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 987–998, New Orleans, Louisiana. Association for Computational Linguistics.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Khyathi Chandu, Ekaterina Loginova, Vishal Gupta, Josef van Genabith, Günter Neumann, Manoj Chinnakotla, Eric Nyberg, and Alan W. Black. 2018. Code-mixed question answering challenge: Crowdsourcing data and techniques. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 29–38, Melbourne, Australia. Association for Computational Linguistics.

Ching-Ting Chang, Shun-Po Chuang, and Hung-Yi Lee. 2019. Code-Switching Sentence Generation by Generative Adversarial Networks and its Application to Data Augmentation. In *Proc. Interspeech 2019*, pages 554–558.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. 2018. Enabling code-mixed translation: Parallel corpus creation and MT augmentation approach. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 131–140, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Mona Diab, Pascale Fung, Mahmoud Ghoneim, Julia Hirschberg, and Thamar Solorio, editors. 2016. *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics, Austin, Texas.

Mona Diab, Julia Hirschberg, Pascale Fung, and Thamar Solorio, editors. 2014. *Proceedings of the First Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics, Doha, Qatar.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1850–1855, Portorož, Slovenia. European Language Resources Association (ELRA).

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2267–2280, Online. Association for Computational Linguistics.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2016. Collecting and annotating indian social media code-mixed corpora. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 406–417. Springer.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations,*

ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation.

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, Austin, Texas. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Braja Gopal Patra, Dipankar Das, and Amitava Das. 2018. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task @icon-2017.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 774–790, Barcelona (online). International Committee for Computational Linguistics.

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en espaÑol: toward a typology of code-switching 1. *Linguistics*, 18:581–618.

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

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553, Melbourne, Australia. Association for Computational Linguistics.

Pattabhi R. K. Rao and S. Devi. 2016. Cmee-il: Code mix entity extraction in indian languages from social media text @ fire 2016 - an overview. In *FIRE*.

Mohd Sanad Zaki Rizvi, Anirudh Srinivasan, Tanuja Ganu, Monojit Choudhury, and Sunayana Sitaram. 2021. GCM: A toolkit for generating synthetic code-mixed text. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 205–211, Online. Association for Computational Linguistics.

Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020. Processing South Asian languages written in the Latin script: the dakshina dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2413–2423, Marseille, France. European Language Resources Association.

Rishiraj Saha Roy, Monojit Choudhury, Prasenjit Majumder, and Komal Agarwal. 2013. Overview of the fire 2013 track on transliterated search. In *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation*, FIRE '12 & '13, New York, NY, USA. Association for Computing Machinery.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code switched text. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5175–5181. International Joint Conferences on Artificial Intelligence Organization.

Thoudam Doren Singh and Thamar Solorio. 2018. Towards translating mixed-code comments from social media. In *Computational Linguistics and Intelligent Text Processing*, pages 457–468, Cham. Springer International Publishing.

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, Doha, Qatar. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936. PMLR.

Vivek Srivastava and Mayank Singh. 2020. PHINC: A parallel Hinglish social media code-mixed corpus for machine translation. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 41–49, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019. Code-switched language models using neural based synthetic data from parallel sentences. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 271–280, Hong Kong, China. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.