Exploring Text-to-Text Transformers for English to Hinglish Machine Translation with Synthetic Code-Mixing

Ganesh Jawahar[1,2] **El Moatez Billah Nagoudi**[1]
Muhammad Abdul-Mageed[1,2] **Laks V.S. Lakshmanan**[2]
Natural Language Processing Lab[1]
Department of Computer Science[2]
The University of British Columbia
ganeshjwhr@gmail.com, {moatez.nagoudi, muhammad.mageed}@ubc.ca, laks@cs.ubc.ca

Abstract

We describe models focused at the understudied problem of translating between monolingual and code-mixed language pairs. More specifically, we offer a wide range of models that convert monolingual English text into Hinglish (code-mixed Hindi and English). Given the recent success of pretrained language models, we also test the utility of two recent Transformer-based encoder-decoder models (i.e., mT5 and mBART) on the task finding both to work well. Given the paucity of training data for code-mixing, we also propose a dependency-free method for generating code-mixed texts from bilingual distributed representations that we exploit for improving language model performance. In particular, armed with this additional data, we adopt a curriculum learning approach where we first finetune the language models on synthetic data then on gold code-mixed data. We find that, although simple, our synthetic code-mixing method is competitive with (and in some cases is even superior to) several standard methods (backtranslation, method based on equivalence constraint theory) under a diverse set of conditions. Our work shows that the mT5 model, finetuned following the curriculum learning procedure, achieves best translation performance (12.67 BLEU). Our models place first in the overall ranking of the English-Hinglish official shared task.

1 Introduction

Code-mixing is a phenomenon of mixing two or more languages in speech and text (Gumperz, 1982). Code-mixing is prevalent in multilingual societies, where the speakers typically have similar fluency in two or more languages (Sitaram et al., 2019). For example, Hindi, Tamil and Telugu speakers from India frequently code-mix with English. Code-mixing can happen between dialects, for example, Modern Standard Arabic is frequently code-mixed with Arabic dialects (Abdul-Mageed et al., 2020). Building NLP systems that can handle code-mixing is challenging as the space of valid grammatical and lexical configurations can be large due to presence of syntactic structures from more than one linguistic system (Pratapa et al., 2018).

In this work, we focus on building a machine translation (MT) system that converts a monolingual sequence of words into a code-mixed sequence. More specifically, we focus on translating from English to Hindi code-mixed with English (i.e., Hinglish). In the literature, work has been done on translating from Hinglish into English (Dhar et al., 2018; Srivastava and Singh, 2020). To illustrate both directions, we provide Figure 1. The Figure presents sample translation pairs for Hinglish to English as well as English to Hinglish. The challenges for solving this task include: (i) lack of Hindi data in roman script (words highlighted in cyan color), (ii) non-standard spellings (e.g., 'isme' vs 'is me'), (iii) token-level ambiguity across the two languages (e.g., Hindi 'main' vs. English 'main'), (iv) non-standard casing (e.g., ROTTEN TOMATOES), (v) informal writing style, and (vi) paucity of English-Hinglish parallel data. Compared with Hinglish to English translation, the English to Hinglish translation direction is a less studied research problem.

English-Hinglish translation can have several practical applications. For example, it can be used to create engaging conversational agents that mimic the code-mixing norm of a human user who uses code-mixing. Another use of resulting Hinglish data would be to create training data for some downstream applications such as token-level language identification.

Our proposed machine translation system exploits a multilingual text-to-text Transformer model along with synthetically generated code-mixed data. More specifically, our system utilizes the state-of-the-art pre-trained multilingual generative model, mT5 (a multilingual variant of "Text-to-Text Trans-

Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, pages 36–46

https://doi.org/10.26615/978-954-452-056-4_006

Hinglish to English translation (Dhar et al. (2018), Srivastava and Singh (2020))		
Hinglish: Hi there! Chat ke liye ready ho?	→	**English**: Hi there! Ready to chat?
Hinglish: isme kids keliye ache message hein, jo respectful hein sabhi keliye	→	**English**: It does show a really good message for kids, to be respectful of everybody
English to Hinglish translation (our task)		
English: Maybe it's to teach kids to challenge themselves	→	**Hinglish**: maybe kida ko teach karna unka challenge ho saktha hein
English: It's seem to do OK on rotten tomatoes I got a 79%	→	**Hinglish**: ROTTEN TOMATOES KA SCORE 79% DIYA HEIN JO OK HEIN KYA

Table 1: Sample translation pairs for Hinglish to English and English to Hinglish machine translation task. Words highlighted in cyan color are in Hindi language in roman script, while non-highlighted words are in English language.

fer Transformer" model (Raffel et al., 2020)) as a backbone. The mT5 model is pretrained on large amounts of monolingual text from 107 languages, making it a good starting point for multilingual applications such as question answering and MT. It is not clear, however, how mT5's representation fares in a code-mixed context such as ours. *This is the question we explore, empirically, in this paper, on our data.* We also introduce a simple approach for generating code-mixed data and show that by explicitly finetuning the model on this code-mixed data we are able to acquire sizeable improvements. For this finetuning, we adopt a *curriculum learning* method, wherein the model is finetuned on the synthetically generated code-mixed data and then finetuned on the gold code-mixed data.

To synthetically generate code-mixed data, we propose a novel lexical substitution method that exploits bilingual word embeddings trained on shuffled context obtained from English-Hindi bitext. The method works by replacing select n-grams in English sentences with their Hindi counterparts obtained from the bilingual word embedding space. For meaningful comparisons, we also experiment with five different methods to create code-mixed training data: (i) *romanization of monolingual Hindi* from English-Hindi parallel data, (ii) *paraphrasing of monolingual English* from English-Hinglish parallel data, (iii) *backtranslation* of output from the mT5 model trained on English-Hinglish parallel data, (iv) *adapting social media data* containing parallel English-Hinglish sentences by removing emoticons, hashtags, mentions, URLs (Srivastava and Singh, 2020), and (v) code-mixed data generated based on *equivalence constraint theory* (Pratapa et al., 2018). We study the impact of different settings (e.g., size of training data, number of paraphrases per input) applicable for most methods on the translation performance. We observe that the mT5 model finetuned on the code-mixed data generated by our proposed method based on bilingual word embeddings followed by finetuning on gold data achieves a BLEU score of 12.67 and places us first in the overall ranking for the shared task. Overall, our major contributions are as follows:

1. We propose a simple, yet effective and dependency-free, method to generate English-Hinglish parallel data by leveraging bilingual word embeddings trained on shuffled context obtained via English-Hindi bitext.

2. We study the effect of several data augmentation methods (based on romanization, paraphrasing, backtranslation, etc.) on the translation performance.

3. Exploiting our code-mixing generation method in the context of curriculum learning, we obtain state-of-the-art performance on the English-Hinglish shared task data with a BLEU score of 12.67.

2 Related Work

Our work involves code-mixed data generation, machine translation involving code-mixed language, and multilingual pretrained generative models.

2.1 Code-Mixed Data Generation

Due to the paucity of code-mixed data, researchers have developed various methods to automatically generate code-mixed data. An ideal method for

code-mixed data generation should aim to generate *syntactically valid* (i.e., fluent), semantically correct words (i.e., adequate), *diverse* code-mixed data of *varying lengths*. To create grammatically valid code-mixed sentences, Pratapa et al. (2018) leverages a linguistically motivated technique based on equivalence constraint theory (Poplack, 1980). They observe that the default distribution of synthetic code-mixed sentences created by their method can be quite different from the distribution of real code-mixed sentences in terms of code-mixing measures. This distribution gap can be largely bridged by post-processing the generated code-mixed sentences by binning them into switch point fraction bins and appropriately sampling from these bins. However, the method depends on availability of a word alignment model, which can be erroneous for distant languages (e.g., Hindi and Chinese) (Gupta et al., 2020). Winata et al. (2019) show that a Seq2Seq model with a copy mechanism can be trained to consume parallel monolingual data (concatenated) as input and produce code-mixed data as output, that is distributionally similar to real code-mixed data. Their method needs an external NMT system to obtain monolingual fragment from code-switched text and is expensive to scale to more language pairs. Garg et al. (2018) introduces a novel RNN unit for an RNN based language model that includes separate components to focus on each language in code-switched text. They utilize training data generated from SeqGAN along with syntactic features (e.g., Part-of-Speech tags, Brown word clusters, language ID feature) to train their RNN based language model. Their method involves added cost to train SeqGAN model and expensive to scale to more language pairs.

Samanta et al. (2019) propose a two-level hierarchical variational autoencoder that models syntactic signals in the lower layer and language switching signals in the upper layer. Their model can leverage modest real code-switched text and large monolingual text to generate large amounts of code-switched text along with its language at token level. The code-mixed data generated by their model seems syntactically valid, yet distributionally different from real code-mixed data and their model is harder to scale for large training data. Gupta et al. (2020) proposes a two-phase approach: (i) creation of synthetic code-mixed sentences from monolingual bitexts (English being one of the languages) by replacing aligned named entities and noun phrases from English; and (ii) training a Seq2Seq model to take English sentence as input and produce the code-mixed sentences created in the first phase. Their approach depends on the availability of a word alignment tool, a part-of-speech tagger, and knowledge of what constituents to replace in order to create a code-mixed sentence. By contrast, our proposed method based on bilingual word embeddings to generate code-mixed data does not require external software such as a word alignment tool, part-of-speech tagger, or constituency parser. Rizvi et al. (2021) develops the toolkit for code-mixed data generation for a given language pair using two linguistic theories: equivalence constraint (code-mixing following the grammatical structure of both the languages) and matrix language theory (McClure, 1995) (code-mixing by fixing a language that lends grammatical structure while other language lends its vocabulary). For comparison, we use this tool to implement the code-mixed data generation method based on equivalence constraint theory.

2.2 Code-Mixed MT

Building MT systems involving code-mixed language is a less researched area. Existing MT systems trained on monolingual data fail to translate code-mixed language such as from Hinglish to English (Dhar et al., 2018; Srivastava and Singh, 2020). Given that neural MT systems require large training data, Dhar et al. (2018) collects a parallel corpus of $6,096$ Hinglish-English bitexts. They propose a machine translation pipeline where they first identify the languages involved in the code-mixed sentence, determine the matrix language, translate the longest word sequence belonging to the embedded language to the matrix language, and then translate the resulting sentence into the target language. The last two steps are performed by monolingual translation systems trained to translate embedded language to matrix language and matrix language to target language respectively. Their proposed pipeline improves the performance of Google and Bing translation systems. Srivastava and Singh (2020) collect a large parallel corpus (called PHINC) of $13,738$ Hinglish-English bitexts that they claim is topically diverse and has better annotation quality than the corpus collected by Dhar et al. (2018). They propose a translation pipeline where they perform token level language identifica-

tion and translate select phrases involving mostly Hindi to English using a monolingual translation system, while keeping the rest of phrases intact. This proposed pipeline outperforms Google and Bing systems on the PHINC dataset. For our work, we make use of the PHINC dataset by adapting the text by removing mentions, hashtags, emojis, emoticons as well as non-meaning bearing constituents such as URLs.

2.3 Multilingual Pretrained Models

Neural models pretrained on monolingual data using a self-supervised objective such as BERT (Devlin et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2020) have become integral to NLP systems as they serve as a good starting point for building SOTA models for diverse monolingual tasks. Recently, there is increasing attention to pretraining neural models on multilingual data, resulting in models such as mBERT (Devlin et al., 2019), XLM (Conneau et al., 2019), mBART (Liu et al., 2020) and mT5 (Xue et al., 2021). Especially, generative multilingual models such as mBART (Liu et al., 2020) and mT5 (Xue et al., 2021) can be utilized directly without additional neural network components to solve summarization, MT, and other natural language generation tasks. These generative models are trained using a self-supervised pretraining objective based on span-corruption objective (mBART and mT5) and sentence shuffling objective (mBART). Training data for these models are prepared by concatenating monolingual texts from multiple languages (e.g., 25 for mBART, 107 for mT5). It is not clear how much code-mixed data these models have seen during pretraining, making it an important question to investigate how they fare in processing text in varieties such as Hinglish. In this work, we target this question by exploring the challenges of applying one of these models (mT5) for the English to Hinglish translation task.

3 Shared Task

The goal of the shared task is to encourage MT involving code-mixing. We focus on translating English to Hinglish. A sentence in Hinglish may contain English tokens and roman Hindi tokens, as shown in Figure 1. The organizers provide $8,060$, 942 and 960 examples for training, validation, and test respectively.

4 Our Approach

Our approach to the English-Hinglish MT task is simple. We first identify the best text-to-text Transformer model on the validation set and follow a curriculum learning procedure to finetune the model for the downstream task. The curriculum learning procedure works such that we first finetune the model using synthetic code-mixed data from our generation method, then further finetune on the gold code-mixed data. This training recipe has been explored previously by Choudhury et al. (2017) and Pratapa et al. (2018) to build code-mixed language models. Curriculum learning itself has been explored previously for different NLP tasks such as parsing (Spitkovsky et al., 2010) and language modeling (Graves et al., 2017). We now present our proposed method to *generate* synthetic code-mixed text for a given language pair.

For our method, we assume having access to large amounts of bitext from a given pair of languages (LG_1 and LG_2) for which we need to generate code-mixed data. Let $B_i = \{x_i, y_i\}$ denote the bitext data, where x_i and y_i correspond to sentences in LG_1 and LG_2, respectively. Let $\texttt{ngrams}(n, x_i, y_i)$ denote the set of unique n-grams in x_i and y_i. Let $\texttt{cumulative-ngrams}(n, x_i, y_i) = \cup_{j=1}^{j=n} \texttt{ngrams}(j, x_i, y_i)$ denote the cumulative set of unique n-grams in the set of pairs x_i and y_i. We shuffle the n-grams in the cumulative set and create a "shuffled" code-mixed sentence by concatenating the shuffled set with n-grams separated by a space. For example, let LG_1 denote English and LG_2 denote Hindi (assuming Roman script for illustration). A sample bitext instance B_i can be "I've never seen it" (x_i) and "maine ye kabhi nah dekhi" (y_i). Set of unique 1-grams will be {"I've", "never", "seen", "it", "maine", "ye", "kabhi", "nah", "dekhi"} ($\texttt{ngrams}(1, x_i, y_i)$, assuming a whitespace tokenizer for simplicity). Then, $\texttt{cumulative-ngrams}(2, x_i, y_i)$ correspond to {"I've", "never", "seen", "it", "maine", "ye", "kabhi", "nah", "dekhi", "I've never", "never seen", "seen it", "maine ye", "ye kabhi", "kabhi nah", "nah dekhi"}. A shuffled code-mixed sentence can be, "I've ye_kabhi never seen_it seen never_seen it kabhi_nah I've_never maine_ye ye kabhi nah dekhi maine nah_dekhi". We create one shuffled code-mixed sentence per bitext instance, thereby creating a shuffled code-mixed corpus. We train a word2vec model on this shuffled code-mixed

corpus to learn embeddings for n-grams in both languages. The resulting word embeddings seem cross-lingually aligned (based on manual inspection), thereby allowing us to do n-gram translation from one language to another language. For example, nearest English neighbor of a Hindi 1-gram "nah" can be "never".

Once the word embeddings are learned, we can create a code-mixed sentence for the given languages: LG_1 and LG_2. We first find the n-grams in $x_i \in LG_1$ and then sort all the n-grams by cosine similarity of the n-gram with its most similar n-gram in LG_2. Let `num-substitutions` denote the number of substitutions performed to convert x_i to a code-mixed sentence. We pick one n-gram at a time from the sorted list and replace all occurrences of that n-gram with its top n-gram belonging to language LG_2 based on word embeddings. We continue this substitution process until we exhaust the `num-substitutions`.

For our machine translation task, we assume LG_1 and LG_2 to be English and Hindi (native) respectively. [1] We feed the OPUS corpus [2] containing 17.2M English-Hindi bitexts (Hindi in native script) as input to the algorithm that outputs English-Hinglish code-mixed parallel data.

5 Experiments

In this section, we first discuss how we choose a text-to-text Transformer model from available models and then introduce our five baseline methods.

5.1 Choosing a Text-to-Text Transformer

Multilingual encoder-decoder models such as mT5 (Xue et al., 2021)[3] and mBART (Liu et al., 2020)[4] are suited to the MT task, and already cover both English and Hindi. It is not clear, however, how these models will perform on a task involving code-mixing at the target side such as ours (where we need to output Hinglish). For this reason, we first explore the potential of these two models on the code-mixed translation task to select the best model among these two. Once we identify the best model, we use it as the basis for further experiments as we will explain in Section 5.3. For both mT5 and mBART, we use the implementation provided by the HuggingFace library (Wolf et al., 2020) with the default settings for all hyperparameters except the maximum number of training epochs, which we choose based on the validation set.

[1] We can assume LG_1 and LG_2 to be Hindi and English respectively, but we leave this exploration for future.

[2] https://opus.nlpl.eu/

[3] https://github.com/google-research/multilingual-t5

[4] https://github.com/pytorch/fairseq/tree/master/examples/mbart

cs method (hyper.)	Valid	Test
Romanization		
IIT-B (100K)	14.27	12.95
IIT-B (250K)	**14.74**	**12.75**
IIT-B (500K)	14.12	12.46
OPUS (100K)	14.67	12.62
OPUS (250K)	14.57	12.71
Paraphrasing		
Para (1)	14.39	12.72
Para (2)	14.4	12.62
Para (3)	**15.07**	**12.63**
Backtranslation		
Forward model	**14.07**	**12.16**
Backward model	**14.51**	**13.03**
Social media		
PHINC	**14.71**	**12.68**
CMDR (ours)		
CMDR-unigram	**14.6**	**12.69**
CMDR-bigram	14.58	12.4

Table 2: Performance in BLEU of mT5 model fine-tuned using curriculum learning — finetuning on one of the different code-mixed data generation method followed by finetuning on gold data. **CMDR:** Code-Mixing from Distributed Representations refers to our proposed method. Note that we did not study the method based on equivalence constraint theory in this experiment. For CMDR, we perform n-gram translation of Hindi from native to roman script.

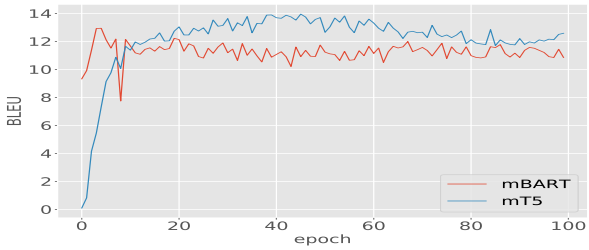

Figure 1: Validation BLEU of mBART and mT5 model on 541 randomly picked examples from the official training set after deduplication, while the rest of the 7000 examples are used for training.

Data Splits. For this set of experiments, we use "custom" splits using the official shared task training data after deduplication[5] and shuffling, as follows: 7,000 examples for training set and 541 examples for validation set. For testing test, we use the official validation data (n=942 examples). We finetune both mT5 and mBART on the custom

[5] Deduplication is done based on exact overlap of source and target text.

split, and show results in Figure 1. We observe that mBART converges quickly within 5 epochs, while mT5 model takes $\sim$ 46 epochs for convergence. Importantly, the best validation performance of 13.95 BLEU is obtained by the mT5 model, which helped us choose mT5 as the backbone model to build our final MT system. For subsequent experiments, we choose 50 as the maximum number of epochs to finetune the mT5 model. We now introduce our baseline code-mixing data generation methods.

5.2 Baseline Code-Mixing Generation Methods

We experiment with five different baseline methods to generate English-Hinglish bitexts that can be used in the first stage of finetuning. We now describe each of these methods.

5.2.1 Monolingual Target Romanization

In this method, we focus on creating monolingual bitexts by taking the Hindi sentence from parallel English-Hindi data and changing the script of Hindi from native script (Devanagari) to Roman script while keeping the English sentence intact. Although the resulting Hindi sentence is monolingual, the generated bitexts can help mT5 model to learn the semantics of Romanized Hindi language (mT5 model might be pretrained on native Hindi), along with the relationships between English and romanized Hindi language. To this end, we exploit two large parallel data sources for English-Hindi pairs (Hindi in native script) — IIT Bombay Parallel corpus (Kunchukuttan et al., 2018) (1.49M bitexts) and OPUS corpus (17.2M bitexts). We utilize the Aksharamukha tool to convert native Hindi to romanized Hindi.[6]

5.2.2 Monolingual Source Paraphrasing

Here, we paraphrase each English source sentence in the gold data to create a new training example, while keeping the target Hinglish sentence intact. Since good paraphrases can typically retain the meaning of the original sentence although the form can be different, we hypothesize the resulting bitext can improve the robustness of our translation system. To generate paraphrases, we use the $T5_{BASE}$ (Raffel et al., 2020) model finetuned on paraphrases from diverse English sources. For our experiments, we use *n* paraphrases of each source sentence, with *n* chosen from the set {1,2,3}. Details about our paraphrasing model are in Appendix A.1.

5.2.3 Backtranslation

We also use the traditional backtranslation pipeline to generate more data for our task. Specifically, we create two models: *forward model* that is obtained by finetuning the mT5 model on English as source and Hinglish as target, *backward model* that is obtained by finetuning mT5 on Hinglish as source and English as target, on the gold training data in both cases. For each gold bitext, the process involves two steps: *forward model inference*, where the gold English sentence is fed to the forward model that generates the intermediate Hinglish sentence; *backward model inference*, where the intermediate Hinglish sentence is fed to the backward model that generates the final English sentence. The new bitext is obtained by pairing up the final English sentence with the gold Hinglish sentence (which is parallel to the English fed to the forward model as source). This method can be treated as an alternative method to creating paraphrases of an English sentence.

5.2.4 Social Media Adaptation

We adapt a publicly available English-Hinglish social media dataset, PHINC (Srivastava and Singh, 2020), to our task. PHINC consists of 13,738 manually annotated English-Hinglish code-mixed sentences, mainly sourced from social media platforms such as Twitter and Facebook. It covers a wide range of topics (such as sports and politics) and has high quality text (e.g., it handles spelling variations and filters abusive and ambiguous sentences). We perform post-processing on PHINC by removing tokens particular to the social media context such as hashtags, mentions, emojis, emoticons and URLs. We use the resulting, adapted, dataset to finetune mT5 for the first stage (as explained in Section 4).

5.2.5 Equivalence Constraint Theory

This method generates code-mixed data based on *equivalence constraint theory* (EC), as originally proposed by Pratapa et al. (2018). The method works by producing parse trees for English-Hindi sentence pair and replaces common nodes between the two trees based on the EC theory. We use the implementation provided by the GCM tool (Rizvi et al., 2021). We feed the English-Hindi bitexts (Hindi in native script) from the OPUS corpus to generate English-Hinglish (Hindi in native script)

[6] https://aksharamukha.appspot.com

parallel data. We now describe our results with mT5 on our custom splits.

5.3 Performance With mT5

As briefly introduced earlier, we finetune the mT5 model using curriculum learning where we have two stages. In stage one, we finetune one of the code-mixed data generation methods. We follow that by stage two where we finetune on the gold code-mixed data (official shared task training data). Also, for stage one, to cap GPU hours with the large synthetic code-mixed data, we experiment with a maximum of 5 epochs. For the stage two, where we have smaller amount of gold data, we experiment with 50 as the maximum number of epochs choosing the best epoch on the validation set.

Table 2 displays the validation and the test performance of mT5 finetuned using curriculum learning. [7] For romanization of monolingual target method, as the Table shows, more data does not strictly improve validation (nor test) performance. That is, there seems to be a 'sweet spot' after which quality deteriorates with noise. The behavior is similar for the models exploiting paraphrases of the source monolingual English data: Adding more paraphrases for a single gold instance can lead to overfitting of the model, as noticed by consistent degradation in test performance. For backtranslation, we experiment with two variants: *forward model* where predictions (Hinglish) from the forward model is paired up with English sentence from the gold, *backward model* which corresponds to the traditional backtranslation bitext. Performance of the backward model is consistently better on both the validation and the test set. For the social media adaptation method, mT5 achieves validation performance that is better than any of the methods based on romanization or backtranslation. For our proposed method based on code-mixing from bilingual distributed representations (CMDR), we experiment with different values of `num-substitutions` and change the script of replaced Hindi words from native to roman script using the Aksharamukha tool. Manual inspection of the data reveals that script conversion at word level is noisy due to lack of sentential context. This might lead to decline in the performance as our method makes more substitutions. Nevertheless, our proposed method, simple as it is, leads to results competitive with any of the other methods.

[7] The best epoch for each stage in the pipeline is displayed in Appendix B.

English (Gold): And they grow apart. She is the protector of the Moors forest. **Hinglish (Prediction)**: Aur wo apart grow karte hai. Wo Moors forest ka (ki) protector hai.
English (Gold): I watched it at least twice.. it was that good. I love female superheros **Hinglish (Prediction)**: Maine ise kam se kam ek (do) baar dekha hai. Ye itni achi thi. Mujhe female superheros pasand hai.
English (Gold): I loved the story & how true they made it in how teen girls act but I honestly don't know why I didn't rate it highly as all the right ingredients were there. I cannot believe it was 2004 it was released though, 14 years ago! **Hinglish (Prediction)**: mujhe story bahut pasand aaya aur teen girls ka act kaise hota lekin main honestly nahi janta kyon ki main ise highly rate nahi kar raha tha kyunki sahi ingredients wahan they. mujhe yakin nahi hota ki 2004 mein release huyi thi, 14 saal pehle!

Table 3: Translations of our proposed system that uses native script and 3 as `num-substitutions`. Errors in translations are highligthed in red color, with their the right translation in paranthesis and highlighted in green color.

5.4 Qualitative Analysis

We manually inspect translations from our proposed system that uses native script and 3 as `num-substitutions` on 25 randomly picked examples from the official test set. 64% of the translations are correct, while 24% and 12% of the translations have grammatical error (e.g., incorrect gender) and semantic errors (e.g., factual inconsistency) respectively. 12% of the translations exactly match with the source. Few of these translations are shown in Table 3. The first translation has grammatical gender error, as it contains male posessive noun, 'ka' (instead of female possessive noun, 'ki'). The second translation has semantic error, where the number of times that the movie has been watched is incorrectly translated as one time ('ek') when the source mentions it as two ('do') times. The third example is long (43 words), which our system translates without errors.

6 Official Results

In this section, we describe the official test performance obtained by our models. First, we experiment with mT5 model finetuned using promising code-mixing methods identified in our previous experiments (see Section 5.3). The best performing baseline method is based on equivalence constraint theory for $100K$ examples and yields a

cs method	BLEU
baseline (mBART model)	11.00
LinCE leaderboard (only best results)	
LTRC Team	12.22
IITP-MT Team	10.09
CMMTOne Team	2.58
Romanization	
OPUS	12.38
Paraphrasing	
Para	12.1
Backtranslation	
Backward model	11.47
Social media	
PHINC	11.9
Equivalence constraint theory	
ECT (100K)	12.45
CMDR (ours)	
CMDR-unigram (roman)	12.25
CMDR-bigram (native)	12.63
CMDR-bigram (roman)	12.08
CMDR-trigram (native)	**12.67**
CMDR-trigram (roman)	12.05
Method Combinations	
CMDR-unigram (roman) + PHINC	11.58
ECT (100K) + CMDR-trigram (native)	12.27

Table 4: Official test performance of mT5 model finetuned using curriculum learning — finetuning on one of the different code-mixed data generation method (max. epochs is 5) followed by finetuning on concatenation of gold training data and gold validation data (leaving out 200 examples for validation) (max. epochs is 50)

BLEU score of 12.45. For the proposed CMDR method, we experiment not only with the value for `num-substitutions`, but also the script type. Surprisingly, the best combination for our proposed method is based on maximum substitutions of 3, sticking to the original native script, and yields the highest BLEU score of 12.67. The variants of our proposed method that romanizes the replacement n-gram consistently perform poorly, which confirms our observation that n-gram level romanization is deprived of sentential context and is prone to errors.

7 Discussion

The lessons learned in this shared task can be summarized as follows.

Similar Text-to-Text Models. *Off-the-shelf mT5 and mBART models perform similarly, with mT5 being slightly better in our experiments (for English-Hinglish MT).* A down side of mT5 is that it takes many more epochs than mBART to converge. In the future, it will be interesting to explore recent extensions of mBART[8], which are already finetuned for multilingual translation. These extensions involve training on English-Hindi (native) bitexts, and so can act as an interesting zero-shot translation baseline without further finetuning. They may also serve as a better baseline when finetuned using the curriculum learning approach adopted in our work.

Code-Mixing from Distributed Representations is Useful. *Our proposed code-mixed data generation method based on bilingual word embeddings can be exploited by mT5 model to achieve the state-of-the-art translation performance,* especially when the number of substitutions is high and the script remains in native form. It will be interesting to see the sweet spot for the number of substitutions, as too low value can result in very less code-mixing while too high value can result in more code-mixing along with more noise (possibly grammatically incorrect and unnatural to bilingual speaker).

Combinations of Code-Mixed Data not Ideal. *Combining code-mixed generations from two methods likely introduces more noise and does not improve the performance of the mT5 model compared to performance obtained using generations from individual method*, as seen in the 'Misc.' section of Table 4. It might be interesting to explore more than two stages of curriculum learning, where the mT5 model is successively finetuned on code-mixed data generated using different methods.

8 Conclusion

We proposed an MT pipeline for translating between English and Hinglish. We test the utility of existing pretrained language models on the task and propose a simple, dependency-free, method for generating synthetic code-mixed text from bilingual distributed representations of words and phrases. Comparing our proposed method to five baseline methods, we show that our method achieves competitively. The method results in best translation performance on the shared task blind test data, placing us first in the official competition. In the future, we plan to (i) scale up the size of code-mixed data, (ii) experiment with different domains of English-Hindi bitexts such as Twitter, (iii) experiment with recent extensions of mBART, and (iv) assess the generalizability of our proposed code-mixing method to other NLP tasks such as question answering and dialogue modeling.

[8]These are `mbart-large-50-many-to-many-mmt`, `mbart-large-50-one-to-many-mmt`, and `mbart-large-50-many-to-one-mmt`.

Acknowledgements

We gratefully acknowledges support from the Natural Sciences and Engineering Research Council of Canada, the Social Sciences and Humanities Research Council of Canada, Canadian Foundation for Innovation, Compute Canada (www.computecanada.ca), and UBC ARC-Sockeye (https://doi.org/10.14288/SOCKEYE).

References

Muhammad Abdul-Mageed, Chiyu Zhang, AbdelRahim Elmadany, and Lyle Ungar. 2020. Micro-dialect identification in diaglossic and code-switched environments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5855–5876.

Monojit Choudhury, Kalika Bali, Sunayana Sitaram, and Ashutosh Baheti. 2017. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 65–74, Kolkata, India. NLP Association of India.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Mathias Creutz. 2018. Open subtitles paraphrase corpus for six languages. *arXiv preprint arXiv:1809.06142*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2018. Code-switched language models using dual RNNs and same-source pretraining. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3078–3083, Brussels, Belgium. Association for Computational Linguistics.

Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks.

John J. Gumperz. 1982. *Discourse Strategies*. Studies in Interactional Sociolinguistics. Cambridge University Press.

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.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

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).

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

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.

Erica McClure. 1995. Duelling languages: Grammatical structure in codeswitching. *Studies in Second Language Acquisition*, 17(1):117–118.

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en espaÑol: toward a typology of code-switching. 18(7-8):581–618.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code-switched text. *CoRR*, abs/1906.08972.

Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W. Black. 2019. A survey of code-switched speech and language processing. *CoRR*, abs/1904.00784.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California. Association for Computational Linguistics.

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.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. SemEval-2015 task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 1–11, Denver, Colorado. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer.

Appendix

A Baseline Code-Mixing Generation Methods

A.1 Monolingual Source Paraphrasing

To generate paraphrases, we use the T5_{BASE} (Raffel et al., 2020) model finetuned on paraphrases from diverse English sources: paraphrase and semantic similarity in Twitter shared task (PIT-2015) (Xu et al., 2015), LanguageNet (tweet) (Lan et al., 2017), Opusparcus (Creutz, 2018) (video subtitle), and Quora question pairs (Q&A website). [9] For all datasets excluding Quora question pairs, we keep sentence pairs with a semantic similarity score $\geq 70\%$. We merge all the datasets, split the resulting data into training, validation, and testing (80%, 10%, and 10%). The T5 model is finetuned on the training split for 20 epochs with constant learning rate of $3e^{-4}$. Given an English sentence to paraphrase, the finetuned model uses top-p sampling (Holtzman et al., 2020) during inference to generate 10 diverse paraphrases. We pick relevant paraphrases for a given sentence by ranking all the generated paraphrases based on the semantic similarity score with the original English sentence and discarding those paraphrases whose semantic similarity score $\geq 95\%$.

B Performance With mT5 On Custom Splits

Table 5 presents the performance of our proposed system on custom splits, along with best epoch for each stage in the pipeline.

[9] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

cs method (hyper.)	S1 epoch	S2 epoch	Valid	Test
Romanization				
IIT-B (100K)	3	50	14.27	12.95
IIT-B (250K)	5	47	**14.74**	**12.75**
IIT-B (500K)	3	46	14.12	12.46
OPUS (100K)	3	43	14.67	12.62
OPUS (250K)	3	50	14.57	12.71
Paraphrasing				
Para (1)	5	43	14.39	12.72
Para (2)	5	43	14.4	12.62
Para (3)	5	44	**15.07**	**12.63**
Backtranslation				
Forward model	3	37	**14.07**	**12.16**
Backward model	3	36	**14.51**	**13.03**
Social media				
PHINC	5	29	**14.71**	**12.68**
CMDR (ours)				
CMDR-unigram	3	48	**14.6**	**12.69**
CMDR-bigram	5	42	14.58	12.4

Table 5: Performance in BLEU of mT5 model finetuned using curriculum learning — finetuning on one of the different code-mixed data generation method (max. epochs is 5) followed by finetuning on gold data (max. epochs is 50). **CMDR:** Code-Mixing from Distributed Representations refers to our proposed method. Validation performance is calculated on 541 randomly picked examples from the official training set after deduplication, while the rest of the $7,000$ examples are used for training. Test performance is calculated on the official validation set. Note that we did not study the method based on equivalence constraint theory in this experiment. For CMDR, we perform n-gram translation of Hindi from native to Roman script.