# From Machine Translation to Code-Switching: Generating High-Quality Code-Switched Text

**Ishan Tarunesh**[*], **Syamantak Kumar**[*], **Preethi Jyothi**
Samsung Korea, Google India, IIT Bombay
{ishantarunesh, syamantak.kumar}@gmail.com, pjyothi@cse.iitb.ac.in

## Abstract

Generating code-switched text is a problem of growing interest, especially given the scarcity of corpora containing large volumes of real code-switched text. In this work, we adapt a state-of-the-art neural machine translation model to generate Hindi-English code-switched sentences starting from monolingual Hindi sentences. We outline a carefully designed curriculum of pretraining steps, including the use of synthetic code-switched text, that enable the model to generate high-quality code-switched text. Using text generated from our model as data augmentation, we show significant reductions in perplexity on a language modeling task, compared to using text from other generative models of CS text. We also show improvements using our text for a downstream code-switched natural language inference task. Our generated text is further subjected to a rigorous evaluation using a human evaluation study and a range of objective metrics, where we show performance comparable (and sometimes even superior) to code-switched text obtained via crowd workers who are native Hindi speakers.

## 1 Introduction

Code-switching (CS) refers to the linguistic phenomenon of using more than one language within a single sentence or conversation. CS appears naturally in conversational speech among multilingual speakers. The main challenge with building models for conversational CS text is that we do not have access to large amounts of CS text that is conversational in style. One might consider using social media text that contains CS and is more readily available. However, the latter is quite different from conversational CS text in its vocabulary (e.g., due to the frequent use of abbreviated slang terms, hashtags and mentions), in its sentence structure (e.g., due to character limits in tweets) and in its word forms (e.g., due to transliteration being commonly employed in social media posts). This motivates the need for a generative model of realistic CS text that can be sampled to subsequently train models for CS text.

In this work, we tackle the problem of generating high-quality CS text using only limited amounts of real CS text during training. We also assume access to large amounts of monolingual text in the component languages and parallel text in both languages, which is a reasonable assumption to make for many of the world's languages. We focus on Hindi-English CS text where the matrix (dominant) language is Hindi and the embedded language is English.[1] Rather than train a generative model, we treat this problem as a translation task where the source and target languages are monolingual Hindi text and Hindi-English CS text, respectively. We also use the monolingual Hindi text to construct synthetic CS sentences using simple techniques. We show that synthetic CS text, albeit being naive in its construction, plays an important role in improving our model's ability to capture CS patterns.

We draw inspiration from the large body of recent work on unsupervised machine translation (Lample et al., 2018a,b) to design our model, which will henceforth be referred to as **T**ranslation for **C**ode-**S**witching, or TCS. TCS, once trained, will convert a monolingual Hindi sentence into a Hindi-English CS sentence. TCS makes effective use of parallel text when it is available and uses backtranslation-based objective functions with monolingual text.

[*] Work done while first two authors were students at IIT Bombay.

[1] Given the non-trivial effort involved in collecting annotations from professional annotators and crowd workers, we focused on a single language pair (Hindi-English) and leave explorations on more language pairs for future work.

Below, we summarize our main contributions:

1. We propose a state-of-the-art translation model that generates Hindi-English CS text starting from monolingual Hindi text. This model requires very small amounts of real CS text, uses both supervised and unsupervised training objectives and considerably benefits from a carefully designed training curriculum, that includes pretraining with synthetically constructed CS sentences.

2. We introduce a new Hindi-English CS text corpus in this work.[2] Each CS sentence is accompanied by its monolingual Hindi translation. We also designed a crowdsourcing task to collect CS variants of monolingual Hindi sentences. The crowdsourced CS sentences were manually verified and form a part of our new dataset.

3. We use sentences generated from our model to train language models for Hindi-English CS text and show significant improvements in perplexity compared to other approaches.

4. We present a rigorous evaluation of the quality of our generated text using multiple objective metrics and a human evaluation study, and they clearly show that the sentences generated by our model are superior in quality and successfully capture naturally occurring CS patterns.

## 2 Related Work

Early approaches of language modeling for code-switched text included class-based $n$-gram models (Yeh et al.), factored language models that exploited a large number of syntactic and semantic features (Adel et al., 2015), and recurrent neural language models (Adel et al., 2013) for CS text. All these approaches relied on access to real CS text to train the language models. Towards alleviating this dependence on real CS text, there has been prior work on learning code-switched language models from bilingual data (Li and Fung, 2014b,a; Garg et al., 2018b) and a more recent direction that explores the possibility of generating synthetic CS sentences. (Pratapa et al., 2018) presents a technique to generate synthetic CS text that grammatically adheres to a linguistic theory of code-switching known as the equivalence constraint (EC) theory (Poplack, 1979; Sankoff, 1998). Lee and Li (2020) proposed a bilingual attention language model for CS text trained solely using a parallel corpus.

Another recent line of work has explored neural generative models for CS text. Garg et al. (2018a) use a sequence generative adversarial network (SeqGAN (Yu et al., 2017)) trained on real CS text to generate sentences that are used to aid language model training. Another GAN-based method proposed by Chang et al. (2019) aims to predict the probability of switching at each token. Winata et al. (2018) and Winata et al. (2019) use a sequence-to-sequence model enabled with a copy mechanism (Pointer Network (Vinyals et al., 2015)) to generate CS data by leveraging parallel monolingual translations from a limited source of CS data. Samanta et al. (2019) proposed a hierarchical variational autoencoder-based model tailored for code-switching that takes into account both syntactic information and language switching signals via the use of language tags. (We present a comparison of TCS with both Samanta et al. (2019) and Garg et al. (2018a) in Section 5.2.1.)

In a departure from using generative models for CS text, we view this problem as one of sequence transduction where we train a model to convert a monolingual sentence into its CS counterpart. Chang et al. (2019); Gao et al. (2019) use GAN-based models to modify monolingual sentences into CS sentences, while we treat this problem of CS generation as a translation task and draw inspiration from the growing body of recent work on neural unsupervised machine translation models (Lample et al., 2018a,b) to build an effective model of CS text.

The idea of using translation models for code-switching has been explored in early work (Vu et al., 2012; Li and Fung, 2013; Dhar et al., 2018). Concurrent with our work, there have been efforts towards building translation models from English to CS text (Solorio et al., 2021) and CS text to English (Gupta et al., 2021). While these works focus on translating from the embedded language (English) to the CS text or vice-versa, our approach starts with sentences in the matrix language (Hindi) which is the more dominant language in the CS text. Also, ours is the first work, to our knowledge, to repurpose an unsupervised neural machine translation model to translate monolingual sentences

[2]The new dataset and relevant code is available at: `https://www.cse.iitb.ac.in/~pjyothi/TCS`.

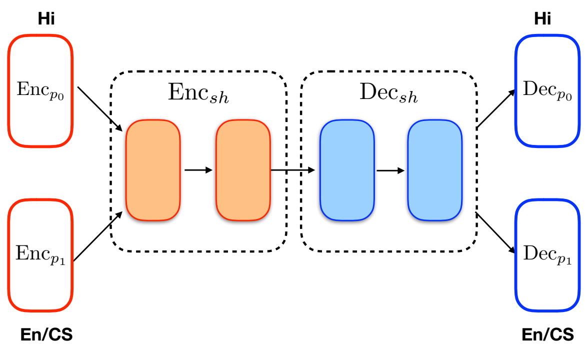
into CS text. Powerful pretrained models like mBART (Liu et al., 2020) have been used for code-mixed translation tasks in concurrent work (Gautam et al., 2021). We will further explore the use of synthetic text with such models as part of future work.

## 3 Our Approach

Figure 1 shows the overall architecture of our model. This is largely motivated by prior work on unsupervised neural machine translation (Lample et al., 2018a,b). The model comprises of three layers of stacked Transformer (Vaswani et al., 2017) encoder and decoder layers, two of which are shared and the remaining layer is private to each language. Monolingual Hindi (i.e. the source language) has its own private encoder and decoder layers (denoted by $\text{Enc}_{p_0}$ and $\text{Dec}_{p_0}$, respectively) while English and Hindi-English CS text jointly make use of the remaining private encoder and decoder layers (denoted by $\text{Enc}_{p_1}$ and $\text{Dec}_{p_1}$, respectively). In our model, the target language is either English or CS text. Ideally, we would like $\text{Enc}_{p_1}$ and $\text{Dec}_{p_1}$ to be trained only using CS text. However, due to the paucity of CS text, we also use text in the embedded language (i.e. English) to train these layers. Next, we outline the three main training steps of TCS.

**(I) Denoising autoencoding (DAE).** We use monolingual text in each language to estimate language models. In Lample et al. (2018b), this is achieved via denoising autoencoding where an autoencoder is used to reconstruct a sentence given a noisy version as its input whose structure is altered by dropping and swapping words arbitrarily (Lample et al., 2018a). The loss incurred in this step is denoted by $\mathcal{L}_{DAE}$ and is composed of two terms based on the reconstruction of the source and target language sentences, respectively.

**(II) Backtranslation (BT):** Once the layers are initialized, one can use non-parallel text in both languages to generate a pseudo-parallel corpus of backtranslated pairs (Sennrich et al., 2015). That is, a corpus of parallel text is constructed by translating sentences in the source language via the pipeline, $\text{Enc}_{p_0}$, $\text{Enc}_{sh}$, $\text{Dec}_{sh}$ and $\text{Dec}_{p_1}$, and translating target sentences back to the source language via $\text{Enc}_{p_1}$, $\text{Enc}_{sh}$, $\text{Dec}_{sh}$ and $\text{Dec}_{p_0}$. The backtranslation loss $\mathcal{L}_{BT}$ is composed of cross-entropy losses from using these pseudo-parallel



$$\mathcal{L}_{DAE}\colon \text{Enc}_{p_0}\text{Enc}_{sh}\text{Dec}_{sh}\text{Dec}_{p_0}\ ;\ \text{Enc}_{p_1}\text{Enc}_{sh}\text{Dec}_{sh}\text{Dec}_{p_1}$$
$$\mathcal{L}_{BT}\colon \text{Enc}_{p_1}\text{Enc}_{sh}\text{Dec}_{sh}\text{Dec}_{p_0}\ ;\ \text{Enc}_{p_0}\text{Enc}_{sh}\text{Dec}_{sh}\text{Dec}_{p_1}$$
$$\mathcal{L}_{CE}\colon \text{Enc}_{p_0}\text{Enc}_{sh}\text{Dec}_{sh}\text{Dec}_{p_1}\ ;\ \text{Enc}_{p_1}\text{Enc}_{sh}\text{Dec}_{sh}\text{Dec}_{p_0}$$

Figure 1: Model architecture. Each loss term along with all the network components it modifies are shown. During unsupervised training with non-parallel text, $\mathcal{L}_{DAE}$ and $\mathcal{L}_{BT}$ are optimized while for supervised training with parallel text, $\mathcal{L}_{DAE}$ and $\mathcal{L}_{CE}$ are optimized.

sentences in both directions.

**(III) Cross-entropy loss (CE):** Both the previous steps used unsupervised training objectives and make use of non-parallel text. With access to parallel text, one can use the standard supervised cross-entropy loss (denoted by $\mathcal{L}_{CE}$) to train the translation models (i.e. going from $\text{Enc}_{p_0}$ to $\text{Dec}_{p_1}$ and $\text{Enc}_{p_1}$ to $\text{Dec}_{p_0}$ via the common shared layers).

### 3.1 Synthetic CS text

Apart from the use of parallel text and monolingual text employed in training TCS, we also construct large volumes of synthetic CS text using two simple techniques. This synthetic CS text is non-parallel and is used to optimize both $\mathcal{L}_{DAE}$ and $\mathcal{L}_{BT}$. The role of the synthetic CS text is to expose TCS to various CS patterns (even if noisy), thereby encouraging the model to code-switch. The final step of finetuning using All-CS enables model to mimic switching patterns of real CS texts

The first technique (named LEX) is a simple heuristic-based technique that constructs a CS sentence by traversing a Hindi sentence and randomly replacing a word by its English translation using a bilingual lexicon (Conneau et al., 2017). The probability of replacing a word is chosen to match the switching distribution in real CS text. The second technique (named EMT) is more linguistically aware. Following the methodology proposed by Bhat et al. (2016) that is based on the embedded matrix theory (EMT) for code-switching, we apply clause substitution methods to monolingual text to construct synthetic CS text. From inspecting English parse trees, we found that replacing embedded sentence clauses or subordinate clauses with their Hindi translations would likely produce CS text that appears somewhat natural.

## 4 Description of Datasets

### 4.1 A New Hindi-English CS Dataset

We introduce a new Hindi-English CS dataset, that we will refer to as All-CS. It is partitioned into two subsets, Movie-CS and Treebank-CS, based on their respective sources. Movie-CS consists of conversational Hindi-English CS text extracted from 30 contemporary Bollywood scripts that were publicly available.[3] The Hindi words in these sentences were all Romanized with potentially multiple non-canonical forms existing for the same Hindi token. We employed a professional annotation company to convert the Romanized Hindi words into their respective back-transliterated forms rendered in Devanagari script. We also asked the annotators to provide monolingual Hindi translations for all these sentences. Using these monolingual Hindi sentences as a starting point, we additionally crowdsourced for CS sentences via Amazon's Mechanical Turk (MTurk) (Amazon, 2005). Table 1 shows two Hindi sentences from Movie-CS and Treebank-CS, along with the different variants of CS sentences.

Turkers were asked to convert a monolingual Hindi sentence into a natural-sounding CS variant that was semantically identical. Each Turker had to work on five Hindi sentences. We developed a web interface using which Turkers could easily copy parts of the Hindi sentence they wanted to retain and splice in English segments. More details about this interface, the crowdsourcing task and worker statistics are available in Appendix A.

All-CS comprises a second subset of CS sentences, Treebank-CS, that was crowdsourcing using MTurk. We extracted 5292 monolingual Hindi sentences (with sentence lengths less than or equal to 15 words) from the publicly available Hindi Dependency Treebank that contains dependency parses.[4] These annotations parse each Hindi sentence into chunks, where a chunk is defined as a minimal, non recursive phrase. Turkers were asked to convert at least one Hindi chunk into English. This was done in an attempt to elicit longer spans of English segments within each sentence. Figure 2 shows the sentence length distributions for Movie-CS and Treebank-CS, along with histograms accumulating English segments of different lengths in both subsets. We clearly see a larger fraction of English segments with lengths within the range [2-6] in Treebank-CS compared to Movie-CS.

Table 2 provides detailed statistics of the new CS dataset. We also report two metrics proposed by Guzmán et al. (2017) to measure the amount of code-switching present in this new corpus. Monolingual Index (M-Index) is a value between 0 and 1

| | |
|---|---|
| Movie-CS | पर हँसी चिकित्सा ने मेरा जीवन बदल दिया वास्तव में |
| (Eng) | (But laughter medicine really changed my life) |
| (Gold) | but laughter therapy ने मेरी life बदल दी actually |
| MTurk | पर laughter therapy ने मेरा जीवन बदल दिया वास्तव में |
| MTurk | but laughter therapy ने really में मेरी life change कर दी |
| MTurk | पर हँसी therapy ने मेरा life बदल दिया वास्तव में |
| Treebank-CS | मेले से आमदनी 7.20 करोड़ रुपये आंकी गई |
| (Eng) | (Income from the fair was estimated at Rs 7.20 crore) |
| MTurk | fair से income 7.20 करोड़ रुपये evaluate की गई |
| MTurk | मेले से income 7.20 करोड़ रुपये आंकी गई |

Table 1: Two All-CS examples. English translations in blue.

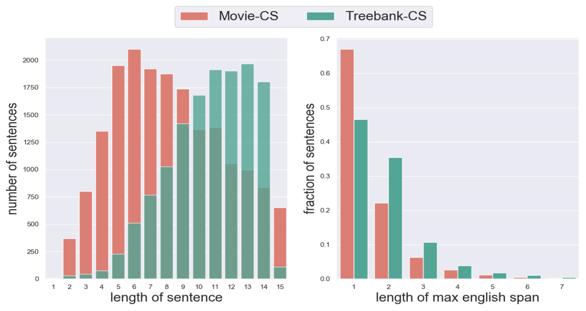


Figure 2: Distribution across overall sentence lengths and distribution across lengths of continuous English spans in Movie-CS and Treebank-CS.

| Quantity/Metric | Movie-CS | Treebank-CS | All-CS |
|---|---|---|---|
| \|Train\| | 15509 | 5914 | 21423 |
| \|Test\| | 1500 | 1000 | 2500 |
| \|Valid\| | 500 | 500 | 1000 |
| # Tokens | 196300 | 87979 | 284279 |
| # Hindi Sentences | 9290 | 5292 | 14582 |
| # NEs | 4342 | 4810 | 9152 |
| Fraction of NEs | 0.0221 | 0.0547 | 0.0322 |
| M-Index | 0.5542 | 0.6311 | 0.5774 |
| I-Index | 0.2852 | 0.3434 | 0.3023 |

Table 2: Key statistics of CS datasets.

[3] https://www.filmcompanion.in/category/fc-pro/scripts/ https://moifightclub.com/category/scripts/

[4] http://ltrc.iiit.ac.in/treebank_H2014/

that quantifies the amount of mixing between languages (0 denotes a purely monolingual corpus and 1 denotes equal mixing from both languages) and I-Index measures the fraction of switching points in the corpus. We observe Treebank-CS exhibits higher M-index and I-index values compared to Movie-CS indicating more code-switching overall. All-CS also contains a non-trivial number of named entities (NEs) which are replaced by an NE tag in all our language modeling experiments.

### 4.2 Other Datasets

**Parallel Hindi-English Text.** As described in Section 5, TCS uses parallel text for supervised training. For this purpose, we use the IIT Bombay English-Hindi Corpus (Kunchukuttan et al., 2017) containing parallel Hindi-English text. We also construct a larger parallel corpus using text from the OpenSubtitles (OpSub) corpus (Lison and Tiedemann, 2016) that is more conversational and hence more similar in style to Movie-CS. We chose ~1 million English sentences (OpSub-EN), where each sentence contained an embedded clause or a subordinate clause to support the construction of EMT lines. We used the Google Translate API to obtain Hindi translations for all these sentences (OpSub-HI). Henceforth, we use OpSub to refer to this parallel corpus of OpSub-EN paired with OpSub-HI. We extracted 318K sentences from the IITB corpus after thresholding on length (5-15) and considering overlap in vocabulary with OpSub. (One could avoid the use of an external service like Google Translate and use existing parallel text (Zhang et al., 2020)) in conjunction with a word aligner to construct EMT lines. OpSub, being more conversational in style, turns out to be a better pretraining corpus. A detailed comparison of these choices is described in Appendix H.)

**Synthetic CS Datasets.** As mentioned in Section 3.1, we use two simple techniques LEX and EMT to generate synthetic CS text, which in turn is used to train TCS in an unsupervised training phase. For each Hindi monolingual sentence in OpSub, we generate two LEX and two EMT synthetic CS sentences giving us OpSub-LEX and OpSub-EMT, respectively. We also generate five LEX and five EMT lines for each monolingual sentence in All-CS. In order to generate EMT lines, we first translate the monolingual Hindi sentences in All-CS to English using Google Translate and then follow the EMT generation scheme. This results in two datasets, All-CS-LEX and All-CS-EMT, which appear in later evaluations. (Appendix B contains more details about EMT applied to OPUS and All-CS.)

**Datasets from existing approaches.** (I) VACS (Samanta et al., 2019) is a hierarchical variational autoencoder-based model designed to generate CS text. We train two VACS models, one on All-CS (VACSv1) and the other on OpSub-EMT followed by All-CS (VACSv2). (II) Garg et al. (2018a) use SeqGAN (Yu et al., 2017) – a GAN-based sequence generation model – to generate CS sentences by providing an RNNLM as the generator. As with VACS, we train two SeqGAN[5] models, one on All-CS (SeqGANv1) and one on OpSub-EMT followed by All-CS (SeqGANv2). Samples are drawn from both SeqGAN and VACS by first drawing a random sample from the standard normal distribution in the learned latent space and then decoding via an RNN-based generator for SeqGAN and a VAE-based decoder for VACS. We sample ~2M lines for each dataset to match the size of the other synthetic datasets.

## 5 Experiments and Results

First, we investigate various training curricula to train TCS and identify the best training strategy by evaluating BLEU scores on the test set of All-CS (§5.1). Next, we compare the output from TCS with synthetic CS text generated by other methods (§5.2). We approach this via language modeling (§5.2.1), human evaluations (§5.2.2) and two downstream tasks—Natural Language Inference and Sentiment Analysis—involving real CS text (§5.2.3). Apart from these tasks, we also present four different objective evaluation metrics to evaluate synthetic CS text: BERTScore, Accuracy of a BERT-based classifier and two diversity scores (§5.3).

### 5.1 Improving Quality of TCS Outputs

Table 3 shows the importance of various training curricula in training TCS; these models are evaluated using BLEU (Papineni et al., 2002) scores computed with the ground-truth CS sentences for

[5] https://github.com/suragnair/seqGAN

| | Curriculum | Training | BLEU (HI    CS) |
|---|---|---|---|
| $\mathbb{O}$ | All-CS | S | 19.18 |
| $\mathbb{A}$ | IITB + OpSub | S | 1.51 |
| $\mathbb{B}$ | $\mathbb{A}$ \| All-CS | S | 27.84 |
| $\mathbb{C}$ | $\mathbb{A}$ \| OpSub-HI + OpSub-LEX | U | 15.23 |
| $\mathbb{D}$ | $\mathbb{A}$ \| OpSub-HI + OpSub-EMT | U | 17.73 |
| $\mathbb{C}_1$ | $\mathbb{C}$ \| All-CS | U | 32.71 |
| $\mathbb{C}_2$ | $\mathbb{C}$ \| All-CS | S | 39.53 |
| $\mathbb{D}_1$ | $\mathbb{D}$ \| All-CS | U | 35.52 |
| $\mathbb{D}_2$ | $\mathbb{D}$ \| All-CS | S | 43.15 |

Table 3: BLEU score on (HI    CS) for different curricula measured on All-CS (test). The first column gives names to each training curriculum. $\mathbb{A}$ | X represents starting with model denoted by $\mathbb{A}$ and further training using dataset(s) X. "S" and "U" refer to supervised and unsupervised training phases, respectively.

the test set of All-CS. We start with supervised pretraining of TCS using the two parallel datasets we have in hand – IITB and OpSub (System $\mathbb{A}$). $\mathbb{A}$ is then further finetuned with real CS text in All-CS. The improvements in BLEU scores moving from System $\mathbb{O}$ (trained only on All-CS) to System $\mathbb{B}$ illustrate the benefits of pretraining TCS using Hindi-English parallel text.

Systems $\mathbb{C}$ and $\mathbb{D}$ in Table 3 use our synthetic CS datasets OpSub-LEX and OpSub-EMT, respectively. These systems are further finetuned on All-CS using both unsupervised and supervised training objectives to give $\mathbb{C}_1$, $\mathbb{C}_2$, $\mathbb{D}_1$ and $\mathbb{D}_2$, respectively. Comparing these four systems with System $\mathbb{B}$ shows the importance of using synthetic CS for pretraining. Further, comparing $\mathbb{C}_1$ against $\mathbb{D}_1$ and $\mathbb{C}_2$ against $\mathbb{D}_2$, we observe that OpSub-EMT is indeed a better choice for pretraining compared to OpSub-LEX. Also, supervised finetuning with All-CS is clearly superior to unsupervised finetuning. Henceforth, Systems $\mathbb{D}_1$ and $\mathbb{D}_2$ will be referred to as TCS (U) and TCS (S), respectively.

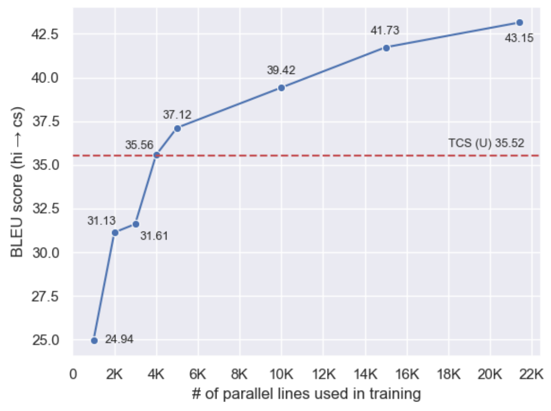


Figure 3: Variation of BLEU score with amount of All-CS parallel training data.

While having access to parallel CS data is an advantage, we argue that the benefits of having parallel data only marginally increase after a threshold. Figure 3 shows how BLEU scores vary when changing the amount of parallel CS text used to train $\mathbb{D}_2$. We observe that BLEU increases substantially when we increase CS data from 1000 lines to 5000 lines, after which there is a trend of diminishing returns. We also find that $\mathbb{D}_1$ (that uses the data in All-CS as non-parallel text) is as good as the model trained using 4000 lines of parallel text.

## 5.2 Comparing TCS with Other Synthetic CS

### 5.2.1 Language Modeling

We use text generated by our model to train a language model (LM) and evaluate perplexities on the test set of All-CS to show how closely sentences from TCS mimic real CS text. We use a state-of-the-art RNNLM model AWD-LSTM-LM Merity et al. (2018) as a blackbox LM and only experiment with different training datasets. The model uses three LSTM layers of 1200 hidden units with weight tying and 300-dimensional word embeddings. In initial runs, we trained our language model on the large parallel/synthetic CS datasets and finetuned on the All-CS data. However, this training strategy was prone to overfitting on All-CS data. To counter this problem of forgetting during the pretrain-finetuning steps, we adopted the Mix-review strategy proposed by He et al. (2021). The training sentences from All-CS remain constant through the epochs and the amount of pretraining data is exponentially decayed with each epoch. This greatly alleviates the forgetting problem in our model, and leads to better overall perplexities. Additional details about these LMs are provided in Appendix E.

Table 4 shows test perplexities using different training curricula and data generated using two prior approaches, VACS and SeqGAN. Sentences generated using TCS yield the largest reductions in test perplexities, compared to all other approaches.

| Pretraining Corpus | Train | Test PPL OpSub | Test PPL All-CS |
|---|---|---|---|
| OpSub + OpSub-LEX | 4.00M | 56.83 | 332.66 |
| OpSub + OpSub-EMT | 4.03M | 55.56 | 276.56 |
| OpSub + VACSv1 | 4.05M | 64.77 | 335.79 |
| OpSub + VACSv2 | 4.05M | 62.41 | 321.12 |
| OpSub + SeqGANv1 | 4.03M | 57.32 | 336.62 |
| OpSub + SeqGANv2 | 4.03M | 56.50 | 317.81 |
| OpSub + TCS (U) | 3.99M | 57.45 | 271.19 |
| OpSub + TCS (S) | 3.96M | 56.28 | **254.37** |

Table 4: Test perplexities on All-CS using different pretraining datasets.

### 5.2.2 Human Evaluation

We evaluated the quality of sentences generated by TCS using a human evaluation study. We sampled 150 sentences each, using both TCS (U) and TCS (S), starting from monolingual Hindi sentences in the evaluation sets of All-CS. The sentences were chosen such that they were consistent with the length distribution of All-CS. For the sake of comparison, corresponding to the above-mentioned 150 monolingual Hindi samples, we also chose 150 CS sentences each from All-CS-LEX and All-CS-EMT. Along with the ground-truth CS sentences from All-CS, this resulted in a total of 750 sentences.[6] These sentences were given to three linguistic experts in Hindi and they were asked to provide scores ranging between 1 and 5 (1 for worst, 5 for best) under three heads: "Syntactic correctness", "Semantic correctness" and "Naturalness". Table 5 shows that the sentences generated using TCS (S) and TCS (U) are far superior to the EMT and LEX sentences on all three criteria. TCS (S) is quite close in overall quality to the real sentences and TCS (U) fares worse, but only by a small margin.

Table 6 shows some illustrative examples of code-switching using TCS (U) on test samples. We also show some examples of code-switching

[6]We only chose CS sentences from TCS that did not exactly match the ground-truth CS text.

| Method | Syntactic | Semantic | Naturalness |
|---|---|---|---|
| Real | 4.47±0.73 | 4.47±0.76 | 4.27±1.06 |
| TCS (S) | 4.21±0.92 | 4.14±0.99 | 3.77±1.33 |
| TCS (U) | 4.06±1.06 | 4.01±1.12 | 3.58±1.46 |
| EMT | 3.57±1.09 | 3.48±1.14 | 2.80±1.44 |
| LEX | 2.91±1.11 | 2.87±1.19 | 1.89±1.14 |

Table 5: Mean and standard deviation of scores (between 1 and 5) from 3 annotators for 150 samples from 5 datasets.

| Generated using MovieCS |
|---|
| मैं खुश हूँ तुमने नोटिस किया<br>(I am glad you noticed)<br>i am happy तुमने notice किया |
| नहीं मैं तुमसे बहुत प्यार करता हूँ सच में लेकिन सिर्फ एक दोस्त की तरह<br>(No i really love you but just like a friend)<br>नहीं i love you very much सच में but सिर्फ एक friend की तरह |
| **Generated using TreebankCS** |
| बैठक अगले हफ़्ते होने की संभावना है<br>(Meeting will likely be next week)<br>meeting next week होने की possibility है |
| उन्होंने कहा कि इनका नाम लेना उचित नहीं होगा लेकिन यह स्पष्ट है<br>(He said that it would not be appropriate to name them but it is clear)<br>उन्होंने कहा कि इनका नाम लेना fair नहीं होगा but it is clear |
| **Generated using OpSub** |
| आपको अपने भीतर उन भावनाओं को संसाधित करने के लिए खुद को समय देना होगा<br>(You have to give yourself time to process those feelings within you)<br>आपको अपने भीतर उन emotions को process करने के लिए खुद को time देना होगा |
| क्योंकि मुझे पता है कि मुख्य पकवान क्या होगा<br>(Because i know what the main dish will be)<br>because i know main dish क्या होगा |

Table 6: Examples generated by TCS (U) on validation and test data. For each example the first line is the monolingual sentence, followed by its English translation and finally the translation from TCS (U). More examples are in Appendix F.

within monolingual sentences from OpSub. We observe that the model is able to introduce long contiguous spans of English words (e.g. "meeting next week", "but it is clear", etc.). The model also displays the ability to meaningfully switch multiple times within the same sentence (e.g., "i love you very much", "but", "friend"). There are also interesting cases of English segments that appear to be ungrammatical but make sense in the CS context (e.g., "because i know main dish", etc.).

### 5.2.3 GLUECoS Benchmark

GLUECoS (Khanuja et al., 2020) is an evaluation benchmark spanning six natural language tasks for code-switched English-Hindi and English-Spanish data. The authors observe that M-BERT (Pires et al., 2019) consistently outperforms cross-lingual embedding techniques. Furthermore, pretraining M-BERT on small amounts of code-switched text improves its performance in most cases. For our evaluation, we select two tasks that require semantic understanding: Natural Language Inference (NLI) and Sentiment Analysis (SA).

We sample 100K monolingual sentences from

| Pretraining Data | NLI (Accuracy) | Sentiment Analysis (F1) |
|---|---|---|
| Baseline | 57.88±1.22 | 57.97±0.06 |
| OpSub-HI | 58.47±0.36 | 58.13±0.25 |
| OpSub-LEX | 58.67±0.94 | 58.40±0.33 |
| OpSub-EMT | 58.96±0.70 | 58.79±0.37 |
| TCS (S) | 59.57±0.57 | **59.39±0.81** |
| All-CS | **59.74±0.96** | 58.77±0.44 |

Table 7: GLUECoS Evaluation: Mean and standard deviation of scores after evaluating on 5 seeds. Baseline denotes the M-BERT model without any MLM pretraining.

OpSub-HI and select corresponding LEX, EMT and TCS (S) sentences. M-BERT is then trained using the masked language modelling (MLM) objective on text from all 4 systems (including OpSub-HI) for 2 epochs. We also train M-BERT on 21K sentences from All-CS (real CS). Finally, these pretrained models are fine-tuned on the selected GLUECoS tasks. (More details are in Appendix G.)

Table 7 lists the accuracies and F1 scores using different pretraining schemes for both NLI and sentiment analysis, respectively. Plain monolingual pretraining by itself leads to performance improvements on both tasks, presumably due to domain similarity between GLUECoS (movie scripts, social media etc.) and OpSub. As mentioned in Khanuja et al. (2020), pretraining on CS text further improves performance for both NLI and SA. Among the synthetic methods, TCS (S) has consistently better scores than LEX and EMT. For SA, TCS (S) even outperforms pretraining on real CS text from All-CS.

## 5.3 Other Objective Evaluation Metrics

**BERTScore.** BERTScore (Zhang* et al., 2020) is a recently-proposed evaluation metric for text generation. Similarity scores are computed between each token in the candidate sentence and each token in the reference sentence, using contextual BERT embeddings (Devlin et al., 2018) of the tokens. We use this as an additional objective metric to evaluate the quality of the sentences generated using TCS. We use the real monolingual sentence as the reference and the generated CS sentence as the candidate, excluding sentences from TCS (S) and TCS (U) that exactly match the real sentence. Since our data is Hindi-English CS text, we use Multilingual BERT (M-BERT) (Pires et al., 2019) for high-quality multilingual representations.

Table 8 outlines our main results on the test set of All-CS. TCS sometimes generates purely monolingual sentences. This might unfairly tilt the scores in favour of TCS since the reference sentences are also monolingual. To discount for such biases, we remove sentences generated by TCS (U) and TCS (S) that are purely monolingual (Row label "Mono" in BERTScore). Sentences having <UNK> tokens (labeled "UNK") are also filtered out since these tokens are only generated by TCS for out-of-vocabulary words. "UNK & Mono" refers to applying both these filters.

EMT lines consistently show the worst performance, which is primarily due to the somewhat poor quality of translations involved in generating these lines (refer to Appendix B). With removing both monolingual and <UNK> tokens, we observe that TCS (U) and TCS (S) yield the highest BERTScores, even outperforming the BERTScore on real data obtained from the Turkers.

**BERT-based Classifier.** In this evaluation, we use M-BERT (Pires et al., 2019) to build a classifier that distinguishes real CS sentences from synthetically generated ones (fake). When subject to examples from high-quality generators, the classifier should find it hard to tell apart real from fake

| Evaluation Metric | | Real | LEX | EMT | TCS (S) | TCS (U) |
|---|---|---|---|---|---|---|
| BERTScore | All (3500) | 0.812 | 0.796 | 0.627 | 0.764 | 0.788 |
| | Mono (3434) | 0.812 | 0.782 | 0.623 | 0.755 | 0.772 |
| | UNK (1983) | 0.809 | 0.804 | 0.636 | 0.827 | 0.846 |
| | UNK & Mono (1857) | **0.808** | 0.785 | 0.633 | **0.813** | **0.821** |
| BERT-based Classifier | \|Sentences\| | 4767 | 12393 | 12484 | 12475 | 12475 |
| | Accuracy(fake) | **42.76** | 96.52 | 97.83 | **80.31** | **88.62** |
| Diversity | Gzip (**D**) | **22.13** | 24.12 | 33.17 | **21.37** | **17.59** |
| | Self-BLEU | **61.3** | 29.7 | 24.6 | **63.6** | **64.2** |

Table 8: (a) BERTScores on test split of All-CS. Each row corresponds to a different data filter. The numbers in parenthesis denote the number of sentences in the data after filtering. (b) Accuracies from the classifier for samples generated by various methods as being fake. The |Sentences| refer to size of dataset for each system. TCS models have the lowest accuracy among synthetic methods. (c) Diversity Scores for different techniques using Gzip and Self-BLEU based diversity measures.

samples. We add a fully connected layer over the M-BERT base architecture that takes the [CLS] token as its input to predict the probability of the sentence being real or fake. Fake sentences are drawn from the union of TCS (U), TCS (S), All-CS-LEX and All-CS-EMT. In order to alleviate the class imbalance problem, we oversample the real sentences by a factor of 5 and shuffle the data. The model converges after training for 5 epochs. We see in Table 8 that the classification accuracy of whether a sample is fake or not is lowest for the outputs from TCS among the different generation techniques.

**Measuring Diversity.** We are interested in finding out how diverse the predictions from TCS are. We propose a simple measure of diversity in the CS variants that is based on how effectively sentences can be compressed using the gzip utility.[7] We considered using Byte Pair Encoding (BPE) (Gage, 1994) as a measure of data compression. However, BPE operates at the level of individual words. Two word sequences "w1 w2 w3" and "w3 w2 w1" would be identically compressed by a BPE tokenizer. We would ideally like to account for such diversity and not discard this information. gzip uses Lempel-Ziv coding (Ziv and Lempel, 1977) that considers substrings of characters during compression, thus allowing for diversity in word ordering to be captured.

Our diversity measure **D** is simply the following: For a given set of CS sentences, run gzip on each sentence individually and sum the resulting file sizes ($S_1$). Next, paste all the CS sentences into a single file and run gzip on it to get a file of size $S_2$. Then, $\mathbf{D} = S_1 - S_2$. Smaller **D** scores indicate larger diversity. If the variants of a sentence are dissimilar to one another and hence very diverse, then $S_2$ would be large thus leading to smaller values of **D**. Table 8 shows the diversity scores for different techniques. Both TCS (S) and TCS (U) have a higher diversity score compared to LEX and EMT. TCS (U) exceeds even the responses received via MTurk (Real) in diversity. We note here that diversity, by itself, is not necessarily a desirable trait. Our goal is to generate sentences that are diverse while being natural and semantically meaningful. The latter properties for text from TCS (S) and TCS (U) have already been verified in our human evaluation study.

Zhu et al. (2018) propose self-BLEU score as a metric to evaluate the diversity of generated data. However, using self-BLEU is slightly problematic in our setting as systems like LEX that switch words at random positions would result in low self-BLEU (indicating high diversity). This is indeed the case, as shown in Table 8 - LEX, EMT give lower self-BLEU scores as compared to TCS. However, note that the scores of the TCS models are comparable to that of real CS data.

[7]`http://www.gzip.org/`

## 6 Conclusions

In this work, we present a neural translation model for CS text that transduces monolingual Hindi sentences into realistic Hindi-English CS text. Text generated by our model is evaluated using a number of different objective metrics, along with LM, NLI and sentiment analysis tasks, and a detailed human evaluation study. The role of synthetic data in training such models merits a more detailed investigation which we leave for future work.

## 7 Acknowledgements

We thank all the anonymous reviewers for their constructive feedback which helped improve the presentation of this work. We also thank all the volunteers who helped with the collection of CS text that is released as part of our dataset, All-CS.

## References

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *IEEE/ACM transactions on audio, speech, and language Processing*, 23(3):431–440.

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2013. Recurrent neural network language modeling for code switching conversational speech. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8411–8415. IEEE.

Amazon. 2005. Amazon mechanical turk website. Visited on 2020-01-02.

Gayatri Bhat, Monojit Choudhury, and Kalika Bali. 2016. Grammatical constraints on intra-sentential code-switching:from theories to working models. *arXiv:1612.04538.*

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, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Yingying Gao, Junlan Feng, Ying Liu, Leijing Hou, Xin Pan, and Yong Ma. 2019. Code-switching sentence generation by bert and generative adversarial networks. In *INTERSPEECH*, pages 3525–3529.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2018a. 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.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2018b. Dual language models for code switched speech recognition. *Proceedings of Interspeech*, pages 2598–2602.

Devansh Gautam, Prashant Kodali, Kshitij Gupta, Anmol Goel, Manish Shrivastava, and Ponnurangam Kumaraguru. 2021. Comet: Towards code-mixed translation using parallel monolingual sentences. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 47–55.

Abhirut Gupta, Aditya Vavre, and Sunita Sarawagi. 2021. Training data augmentation for code-mixed translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5760–5766.

Gualberto A. Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E. Bullock, and Almeida Jacqueline Toribio. 2017. Metrics for modeling code-switching across corpora. In *INTERSPEECH*.

Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P. Bigham. 2018. *A Data-Driven Analysis of Workers' Earnings on Amazon Mechanical Turk*, page 1–14. Association for Computing Machinery, New York, NY, USA.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2021. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1121–1133, Online. Association for Computational Linguistics.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The IIT bombay english-hindi parallel corpus. *CoRR*, abs/1710.02855.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Grandee Lee and Haizhou Li. 2020. Modeling code-switch languages using bilingual parallel corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 860–870, Online. Association for Computational Linguistics.

Ying Li and Pascale Fung. 2013. Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7368–7372. IEEE.

Ying Li and Pascale Fung. 2014a. Code switch language modeling with functional head constraint. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4913–4917.

Ying Li and Pascale Fung. 2014b. Language modeling with functional head constraint for code switching speech recognition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 907–916, Doha, Qatar. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pretraining for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

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 on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

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

Shana Poplack. 1979. "sometimes i'll start a sentence in spanish y termino en español": Toward a typology of code-switching.

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 ACL 2018*. ACL.

Mohd Sanad Zaki Rizvi, Anirudh Srinivasan, T. Ganu, M. Choudhury, and Sunayana Sitaram. 2021. Gcm: A toolkit for generating synthetic code-mixed text. In *EACL*.

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.

David Sankoff. 1998. A formal production-based explanation of the facts of code-switching. *Bilingualism: Language and Cognition*, 1(1):39–50.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Thamar Solorio, Shuguang Chen, Alan W. Black, Mona Diab, Sunayana Sitaram, Victor Soto, and Emre Yilmaz, editors. 2021. *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*. Association for Computational Linguistics, Online.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li. 2012. A first speech recognition system for mandarin-english code-switch conversational speech. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4889–4892. IEEE.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Learn to code-switch: Data augmentation using copy mechanism on language modeling. *CoRR*, abs/1810.10254.

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 *CoNLL*.

Ching Feng Yeh, Chao Yu Huang, Liang Che Sun, and Lin Shan Lee. An integrated framework for transcribing mandarin-english code-mixed lectures with improved acoustic and language modeling. In *2010 7th International Symposium on Chinese Spoken Language Processing*, pages 214–219. IEEE.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research  Development in Information Retrieval*, SIGIR '18, page 1097–1100, New York, NY, USA. Association for Computing Machinery.

J. Ziv and A. Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343.

## A MTurk Task Details

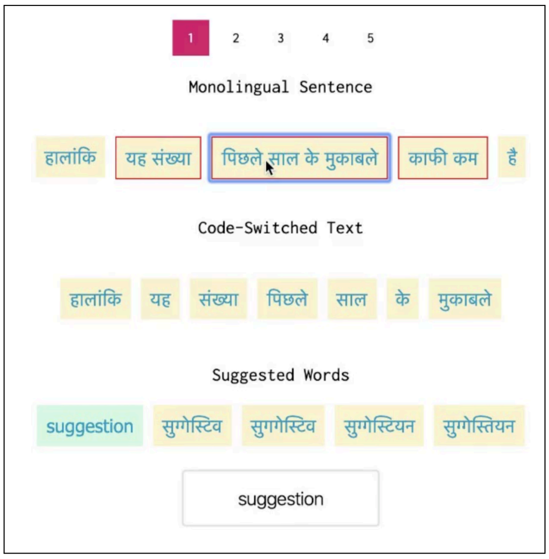


Figure 4: A snapshot of the web interface used to collect Movie-CS and Treebank-CS data via Amazon Mechanical Turk.

Figure 4 depicts the portal used to collect data using Amazon's Mechanical Turk platform. The collection was done in two rounds, first for Movie-CS and then for Treebank-CS. With Treebank-CS, the sentences were first divided into chunks and the Turkers were provided with a sentence grouped into chunks as shown in Figure 4. They were required to switch at least one chunk in the sentence entirely to English so as to ensure a longer span of English words in the resulting CS sentence. A suggestion box converted transliterated Hindi words into Devanagari and also provided English suggestions to aid the workers in completing their task. With Movie-CS, since there were no chunk labels associated with the sentences, they were tokenized into words.

On MTurk, we selected workers with HIT approval rate of 90% and location restricted to countries with significant Hindi speakers - Australia, Bahrain, Canada, India, Kuwait, Malaysia, Mauritius, Myanmar, Nepal, Netherlands, New Zealand, Oman, Pakistan, Qatar, Saudi Arabia, Singapore, South Africa, Sri Lanka, Thailand, United Arab Emirates, United Kingdom, United States of America. It was clearly specified in the guidelines that the task must be attempted by native Hindi speakers. Each response was manually checked before approving. Turkers were paid $0.15 for working on 5 sentences (roughly takes 3-4 minutes). This amounts to $2.25-$3/hr which is in the ballpark of a median hourly wage on MTurk of ~$2/hr (Hara et al., 2018).

## B EMT lines generation

Following the methodology described in (Bhat et al., 2016), we apply clause substitution methodology to produce EMT sentences. To create OpSub-EMT, we start with the gold English sentence that contains either embedded sentence clauses (S) or subordinate clauses (SBAR) and swap one or more of them with their Hindi translations to produce an EMT synthetic CS sentence. Due to the lack of gold English translations available for All-CS sentences, we used the Google Translate API to first acquire their English translation. Many of the sentences in All-CS are shorter in length and do not contain the abovementioned clauses. So, we also considered inverted declarative sentence clauses (SINV), inverted question clauses (SQ) and direct question clauses (SBARQ) in addition to S and SBAR. In case none of the clause level tags were present, we considered the following phrase level tags as switching candidates: Noun Phrase (NP), Verb Phrase (VP), Adjective Phrase (ADJP) and Adverb Phase (ADVP). Owing to the shorter length and lack of clause-level tags, we switch only one tag per sentence for All-CS-EMT. The choice of which clause to switch was made empirically by observing what switches caused the resulting sentence to resemble a naturally occurring CS sentence. One can also use the toolkit provided by Rizvi et al. (2021) for generating EMT lines.

## C Implementation Details: TCS

As an initialisation step, we learn the token embeddings (Mikolov et al., 2013) on the same corpus using skipgram. The embedding dimension was set to be 256 and the encoder-decoder layers share these lookup tables. Adam optimiser with a learning rate of 0.0001 was used to train the model. Validation BLEU scores on (HI $\rightarrow$ ENG/CS) translations and (EN $\rightarrow$ HI $\rightarrow$ EN) reconstructions were used as metrics to save the best model for TCS (S) and TCS (U), respectively.

## D Human Evaluation

The 150 samples evaluated in Table 5 were taken entirely from test/validation splits. We undertook an alternate human evaluation experiment involving 100 real CS sentences and its corresponding

CS sentences using LEX, EMT, TCS (U) and TCS (S). Out of these 100 sentences, 40 of them came entirely from the test and validation splits and the remaining 60 are training sentences which we filtered to make sure that sentences generated by TCS (S) and TCS (U) never exactly matched the real CS sentence. The table below (Table 9) reports the evaluations on the complete set of 100 sentences from 5 datasets. We observe that the trend remains exactly the same as in Table 5, with TCS (S) being very close to real CS sentences in its evaluation and TCS (U) trailing behind TCS (S).

| Method | Syntactic | Semantic | Naturalness |
|---|---|---|---|
| Real | 4.36±0.76 | 4.39±0.80 | 4.20±1.00 |
| TCS (S) | 4.29±0.84 | 4.30±0.89 | 4.02±1.16 |
| TCS (U) | 3.96±1.06 | 3.93±1.13 | 3.52±1.45 |
| EMT | 3.47±1.25 | 3.53±1.23 | 2.66±1.49 |
| LEX | 3.10±2.16 | 3.05±1.35 | 2.01±1.32 |

Table 9: Mean and standard deviation of scores (between 1 and 5) from 3 annotators for 100 samples from 5 datasets.

## E Language Model Training

The AWD-LSTM language model was trained for 100 epochs with a batch size of 80 and a sequence length of 70 in each batch. The learning rate was set at 30. The model uses NT-ASGD, a variant of the averaged stochastic gradient method, to update the weights. The mix-review decay parameter was set to 0.9. This implies that the fraction of pretraining batches being considered at the end of $n$ epochs is $0.9^n$, starting from all batches initially. Two decay coefficients $\{0.8, 0.9\}$ were tested and 0.9 was chosen based on validation perplexities.

## F Code-switching examples

The sentences in Table 10 have been generated on the test and validation splits of All-CS as well as the OpSub dataset. Overall, they depict how the model is able to retain context over long sentences (e.g. "and social sectors") and perform meaningful switching over large spans of words (e.g. "old conversation writer media", "regularly security practices"). We also note that at times, the model uses words which are different from the natural English translations of the sentence, which are appropriate within the context of a CS sentence (e.g. the use of "manage" instead of "manageable").

## G Details of GLUECoS Experiments

For masked language modeling (MLM), we select the default parameters for the learning rate (5e-5), batch masking probability (0.15), sequence length (512). The models are trained for 2 epochs with a batch size of 4 and gradient accumulation step of 10. For task specific fine tuning we rely on the official training scripts provided by GLUECoS repository. [8] We train the models for 5 seed (0,1,2,3 and 4) and report mean and standard deviations of Accuracy and F1 for NLI and Sentiment Analysis respectively

## H Additional Dataset and Experiments

**Dataset** The additional corpus on which experiments were performed is OPUS-100 (Zhang et al., 2020) which was sampled from the original OPUS corpus (Tiedemann, 2012). The primary difference between OpSub and OPUS-100 is that OpSub does not have manual Hindi translations

[8]https://github.com/microsoft/GLUECoS

| Generated using Movie-CS |
|---|
| सारे पुराने बातचीत लेखक मीडिया और राजनीति में जमा हो गए हैं<br>(All the old conversation writers have gathered in media and politics)<br>सारे old conversation writer media और politics में जमा हो गए हैं |
| क्या बात है तुमने आखरी बार कब पार्टी की थी<br>(What is the last time you had a party)<br>क्या बात है तुमने last time party कब की थी |
| तू अपने कमरे में जा यार आप दोनों कृपया शांत हो जाओ<br>(You go to your room man please relax both of you)<br>तू अपने room में जा यार आप दोनों please calm down |
| Generated using TreebankCS |
| यह पॉलिसी पति पत्नी के संयुक्त नाम से थी<br>(This policy was in the joint name of husband and wife)<br>यह policy husband wife के joint नाम से थी |
| स्कूलों में तो नियमित रूप से सुरक्षा अभ्यास कराए जाने लगे हैं<br>(Regular safety exercises are being conducted in schools)<br>schools में तो regularly security practice किये जाने लगे हैं |
| इसमें बुनियादी कृषि और सामाजिक क्षेत्रों में सार्वजनिक निवेशभी शामिल है<br>(It also includes public investment in basic agricultural and social sectors)<br>इसमें बुनियादी farming and social areas में public investment भी शामिल है |
| Generated using OpSub |
| इस सम्मेलन का चौथा विषय मानव पूंजी विकास उपायों पर बल देना है<br>(The fourth theme of this conference is to emphasize human capital development measures.)<br>इस सम्मेलन का fourth subject human पूंजी development उपायों पर बल देना है |
| देश का आंतरिक कर्ज प्रबन्ध किए जाने योग्य सीमा में है<br>(The country's internal debt is within manageable limits)<br>देश का internal loan manage किए जाने योग्य सीमा में है |

Table 10: More examples of code-switching generated by TCS (U).

of its sentences and requires the use of an external API such as Google Translate for translation. However, OPUS-100 has manually annotated sentences as part of the corpus. The source of OPUS-100 ranges from movie subtitles to GNOME documentation to the Bible. We extract 340K sentences from OPUS-100 corpus after thresholding on length (5-15). We offer this comparison of systems trained on OpSub and OPUS-100 to show how our models fare when using two datasets that are very different in their composition.

**LEX lines generation.** Generation of LEX lines is straightforward and requires only a bilingual lexicon. For each monolingual Hindi sentence we generate ~5 sentences on OPUS-100 resulting in OPUS-100-LEX (to roughly match the size of OpSub-LEX).

**EMT lines generation.** For generation of EMT lines we have two strategies depending on the availability of tools (parsers, translation service, aligners, etc). The first strategy requires a translation service (either in-house or publicly available). We substitute the embedded clause from parse trees of English sentences with their Hindi translations. This strategy does not require a parallel Hindi corpus and has been previously used for generating OpSub-EMT and All-CS-EMT (Described in detail in Appendix B).

The second strategy, that is used to generate OPUS-100-EMT, requires a parallel corpus, a constituent parser in English and a word aligner between parallel sentences. OPUS-100 sentences are aligned using SimAlign (Jalili Sabet et al., 2020) and embedded clauses from parse trees of English sentences are replaced by Hindi clauses using word aligners. Here again, for each monolingual Hindi sentenece we generate ~5 EMT sentences (strategy-2) on OPUS-100 resulting in OPUS-100-EMT.

**Curriculum Training Experiments.** Table 11 provides a walkthrough of systems using various training curricula that are evaluated for two different choices of datasets - OpSub vs OPUS-100 differing in the generation of EMT lines. The models are evaluated using BLEU (Papineni et al., 2002) scores computed on the test set of All-CS. The vocabulary is generated by combining train sets of all datasets to be used in the curricula. It is 126,576 when X = OpSub and 164,350 when X = OPUS-100 (OpSub shows a higher overlap in vocabulary with All-CS compared to OPUS-100). The marginal difference in System $\mathbb{O}$ for OpSub and OPUS-100 is attributed to differences in the size of the vocabulary. OpSub being conversational in nature, is a better pretraining corpus compared to OPUS-100 as seen from System $\mathbb{A}$, the sources of the latter being GNOME documentations and The Bible, apart from movie subtitles.

|  | Curriculum | X=OpSub | X=OPUS-100 |
|---|---|---|---|
| $\mathbb{O}$ | All-CS (S) | 19.18 | 19.14 |
| $\mathbb{A}$ | IITB + X (S) | 1.51 | 0.29 |
| $\mathbb{B}$ | $\mathbb{A}$ \| All-CS (S) | 27.84 | 25.63 |
| $\mathbb{C}$ | $\mathbb{A}$ \| X-HI + X-LEX (U) | 15.23 | 14.17 |
| $\mathbb{C}_1$ | $\mathbb{C}$ \| All-CS (U) | 32.71 | 31.48 |
| $\mathbb{C}_2$ | $\mathbb{C}$ \| All-CS (S) | 39.53 | 37.51 |
| $\mathbb{D}$ | $\mathbb{A}$ \| X-HI + X-EMT (U) | 17.73 | 15.03 |
| $\mathbb{D}_1$ | $\mathbb{D}$ \| All-CS (U) | 35.52 | 33.91 |
| $\mathbb{D}_2$ | $\mathbb{D}$ \| All-CS (S) | 43.15 | 40.32 |

Table 11: BLEU score on (HI CS) for different curricula measured on All-CS (test). $\mathbb{X}$ | Y represents starting with model X and further training using dataset Y. Values from Table 3 are replicated here for ease of comparison.

The results for $\mathbb{C}_1$, $\mathbb{C}_2$, $\mathbb{D}_1$, $\mathbb{D}_2$ are consistently better when X = OpSub versus when X = OPUS-100. We choose to highlight four models from Table 11 which together demonstrate multiple use-cases of TCS in Table 12. TCS (LEX) refers to ($\mathbb{C}_2$, X=OpSub), TCS (U) refers to ($\mathbb{D}_1$, X=OpSub), TCS (S) refers to ($\mathbb{D}_2$, X=OpSub) and TCS (simalign) refers to ($\mathbb{D}_2$, X=OPUS-100).

**Language Modelling Experiments.** Table 13 shows results from LM experiments (using the same setup as in Section 5.2.1). The values for TCS (S) and TCS (U) have been reproduced here

| TCS Model | Use-Case |
|---|---|
| TCS (LEX) | Easy generation of sentences,<br>only requires a bilingual lexicon |
| TCS (U) and TCS (S) | Requires parser and translation service<br>Does not require parallel data |
| TCS (simalign) | Requires parser along with parallel data<br>Alignment can be generated<br>using SimAlign |

Table 12: Use cases for different TCS models.

| Pretraining Corpus | \| Train \| | Test PPL OpSub | Test PPL All-CS |
|---|---|---|---|
| OpSub + TCS (LEX) | 4.03M | 57.24 | 268.54 |
| OpSub + TCS (U) | 3.99M | 57.45 | 271.19 |
| OpSub + TCS (simalign) | 4.03M | 60.01 | 314.28 |
| OpSub + TCS (S) | 3.96M | 56.28 | **254.37** |

Table 13: Test perplexities on OpSub and All-CS using different pretraining datasets.

for ease of comparison. (Note that TCS (simalign) does not perform as well as the other models since the sentences for training the language model are generated on OpSub for all the models here, but TCS (simalign) has been trained on OPUS-100.)

**Evaluation Metrics.** Table 14 shows the results of the three objective evaluation metrics on the additional TCS models. In comparison with the results in Table 8, we observe that TCS (LEX) and TCS (simalign) perform comparably to TCS (S) and TCS (U) on all metrics.

| Evaluation Metric | | TCS (LEX) | TCS (simalign) |
|---|---|---|---|
| BERTScore | All (3500) | 0.773 | 0.768 |
| | Mono (3434) | 0.769 | 0.753 |
| | UNK (1983) | 0.832 | 0.829 |
| | UNK & Mono (1857) | **0.817** | **0.822** |
| BERT-based Classifier | \|Sentences\| | 12475 | 12475 |
| | Accuracy(fake) | **84.17** | **82.98** |
| Diversity | Gzip (**D**) | **19.62** | **19.83** |
| | Self-BLEU | **56.3** | **59.8** |

Table 14: Evaluation metrics for the additional TCS models. Please see Table 8 for a comparison with other models.