Language Agnostic Code-Mixing Data Augmentation by Predicting Linguistic Patterns

Anonymous ACL submission

Abstract

In this work, we focus on intrasentential code-mixing and propose several different Synthetic Code-Mixing (SCM) data augmentation methods that outperform the baseline on downstream sentiment analysis tasks across various amounts of labeled gold data. Most importantly, our proposed methods demonstrate that strategically replacing parts of sentences in the matrix language with a constant mask significantly improves classification accuracy, motivating further linguistic insights into the phenomenon of code-mixing. We test our data augmentation method in a variety of low-resource and cross-lingual settings, reaching up to a relative improvement of 7.73% on the extremely scarce English-Malayalam dataset. We conclude that the code-switch pattern in code-mixing sentences is also important for the model to learn. Finally, we propose a language-agnostic SCM algorithm that is cheap yet extremely helpful for low-resource languages.

1 Introduction

Code-mixing sentences have complex syntactic structures and a large vocabulary across languages. It is difficult for someone not fluent in both languages to understand a code-mixed conversion. Due to its spontaneous nature, code-mixing text data is hard to collect and therefore extremely low-resource. Most text-based code-mixing happens in casual settings such as blogs, chats, product ratings, and comments, and most predominantly on social media. A model for natural language processing (NLP) tasks for code-mixed languages is necessary, but current NLP studies on social media texts focus mostly on English (Farzindar and Inkpen, 2015; Coppersmith et al., 2018; Hodorog et al., 2022; Oyebode et al., 2022). However, the fact that English has become the *lingua franca* in most social media apps can be helpful in cross-lingual generalization of code-mixing languages with one of the languages being English (Choudhury, 2018). A recent survey on code-mixing datasets for any downstream tasks found that 84% of the code-mixing pairs contain English (Jose et al., 2020). Therefore, having a model for English-*X* (any other language) code-mixing languages is crucial for future NLP research on code-mixing.

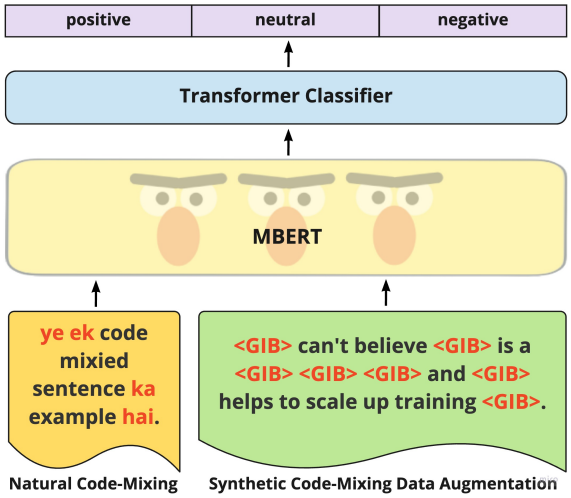

Figure 1: SCM Data Generation Model Pipeline. Large amounts of synthetic data are combined with limited natural data to finetune a multilingual PLM (Devlin et al., 2018) with a transformer classifier layer for sentiment analysis.

Sentiment analysis (SA) is an important research area in social media NLP, which learns to predict the sentiment of a piece of informal text, often in the form of a few sentences or a paragraph. SA models are widely used in social media for public emotion detection (Ortigosa et al., 2014), video/post suggestions, and commercially for analyzing customer feedback, business trend, etc (Drus and Khalid, 2019). In the code-mixing setting, there is also a need for SA tools as the switching and choice of language for a multilingual speaker contains higher-level implications that are valuable for downstream needs (Kim et al., 2014). Code-mixing sentences are composed of multiple languages, making multilingual Pre-trained Language

Models (PLM) (Devlin et al., 2018) the natural choice for the starting point to explore NLP tasks in this new domain. However, as we will discuss in Section 2.3, there are limitations on the ability of multilingual PLMs to be directly adopted for code-mixing languages.

As shown in Figure 1, we attempt to solve the domain mismatch between multilingual PLM and extremely low-resource, unseen code-mixing data by introducing a Synthetic Code-Mixing (SCM) data augmentation pipeline for code-mixing social media sentiment analysis. In our proposed data augmentation pipeline, large SCM data is combined with limited Natural Code-Mixing (NCM) data to fine-tune a pre-trained language mode with a classification layer.

We introduce a low-cost, language-agnostic, and label-preserving algorithm to mass-produce synthetic code-mixing sentences. In this work, our contributions include:

- Our synthetic English-Hindi code-mixing data augmentation technique is shown to be extremely helpful for SA across a variety of model complexity and low-resource levels.
- We investigate the linguistic nature of code-mixing from a computational perspective, concluding that the models learn from the form of code-mixing more so than from the semantics of individual constituents.
- We introduce a low-cost, *language-agnostic*, and label-preserving algorithm that allows for the rapid production of a universal corpus for code-mixing sentences. We demonstrated improvement over all the language pairs and a 7.73 percent improvement in the weighted F1-score on extremely low-resource languages.

2 Related Work

2.1 Code-Mixing

The earliest studies on code-mixing mostly focus on linguistics (Bokamba, 1988). This branch of work inspires a lot of computational efforts to model code-mixing based on syntactic constraints Pratapa et al. (2018) However, code-mixing is also a social phenomenon, where the choice of language and the switching encode implicit meanings that only people within the same community would understand, and it often reflects intimacy and group identity (Ho et al., 2007; Kim et al., 2014). In order to model code-mixing sentences, one would need knowledge of the degree of multilinguality in the community, the speaker-audience relationship, the occasion, and the intended effect of the communication (Bokamba, 1989).

Code-mixing can be considered a different language from either of the parent languages and it is extremely low-resource. The dominant language that controls the syntax is called a matrix language ($\mathcal{M}$) and the language that supplies the phrase meaning is called the embedded language ($\mathcal{E}$) (Auer and Muhamedova, 2005; Myers-Scotton, 1992). In our work, we will focus on intrasentential code-mixing where words or phrases in the same sentence are from different languages.

2.2 Code Mixing NLP

Code-mixing has gained growing interest in the NLP community in recent years. Since it happens spontaneously during casual conversations, code-mixing is relatively more well-studied in the speech domain, such as automatic speech recognition (Chan et al., 2009). In the text format, spelling errors and script inconsistency in code-mixing languages make code-mixing NLP a harder challenge on top of its low-resource nature (Thara and Poornachandran, 2018; Rudra et al., 2016). Existing works have been attempting to improve the quality of code-mixing entity extraction (Rao and Devi, 2016), question answering (Obrocka et al., 2019), and fine-tuning multilingual PLMs for intent prediction and slot filling (Krishnan et al., 2021). Currently, available parallel corpora rely on non-scalable manual annotations, adding bias and noise to the data (Chakravarthi et al., 2020b; Dhar et al., 2018; Srivastava and Singh, 2020).

Code-Mixing Generation Some existing attempts to generate synthetic code-mixing data use subjective rule-based systems on parallel corpora, but standardized metrics like BLEU (Papineni et al., 2002; Doddington, 2002) have proven them ineffective (Srivastava and Singh, 2021). Pratapa et al. (2018) used the Equivalence Constraint theory to force align the parse tree of the two languages to replace words in the source sentence. This approach results in natural-sounding English-Hindi code-mixing sentences but relies on the assumption that parallel sentences in English and Hindi - both Indo-European languages - can be parsed with similar parse trees (Chang et al., 2015), which is not often the case for more distant language pairs.

Sentiment Analysis A sentiment analysis model specifically for code-mixing NLP. Such a model

would be able to capture richer and more fine-grained sentiments in the switching of the language rather than the mere semantics of individual words. Some work code-mixing SA follows a *translate-then-classify* paradigm to translate the code-mixing sentences in Hinglish into English first, and then use a monolingual classifier for the SA task (Gautam et al., 2021). Others have used carefully crafted *meta embeddimgs* to predict the sentiments (Dowlagar and Mamidi, 2021).

2.3 Multilingual Pre-trained Models

Since code-mixing is a mixture of two or multiple languages, the use of multilingual language models is an important addition to code-mixing NLP. In recent years, there have been a lot of transformer-based large pre-trained models trained on monolingual data from multiple languages in an attempt to capture multilingual information (Devlin et al., 2018; Conneau et al., 2019; Liu et al., 2020b; Xue et al., 2020; Ouyang et al., 2020). XLM-T is a multilingual language model specifically focusing on the domain of social media, significantly outperforming its competitors on sentiment analysis and the TweetEval benchmark (Barbieri et al., 2022).

These models are trained on shared multilingual subword embeddings, but the context for training is still monolingual due to the nature of the training corpus. For example, the sentence embedding (not individual token) space of different languages in mBERT shows nearly no overlap, which limits its ability to understand code-mixing languages (Qin et al., 2020). Therefore, directly adopting multilingual language models trained on monolingual corpora from different languages is not enough for code-mixing NLP due to its rich linguistic structure (Krishnan et al., 2021).

3 Methods

In this section, we introduce a language-agnostic zero-cost data augmentation method that encodes the code-switch pattern with English and a constant mask, which provides an efficient universal data augmentation for any code-mixing sentences containing English. Then, we propose three algorithms used to generate the SCM.

3.1 Language Pair Selection

Language-Specific We generate code-mixing sentences that are in the same language pair as the test data. We take the higher-resource language in the pair as the matrix language ($\mathcal{M}$) leveraging the labeled monolingual data. The other language is taken as the embedded language ($\mathcal{E}1$). Labeled data in $\mathcal{M}$ can be fully utilized as we assume that code-mixing sentence generation from a monolingual sentence is label-preserving. We translate part of the source sentence in $\mathcal{M}$ into $\mathcal{E}1$ and put the translation back to the source sentence to get the final SCM data, which is described more in detail in Section 3.2. The language-specific synthetic data, $\mathcal{M}$-$\mathcal{E}1$,is shown in the dotted red box of Figure 2.

Cross-lingual Code-mixing sentences from social media often contain that share the same matrix language, $\mathcal{M}$, and various embedded languages $\mathcal{E}i$ (Choudhury, 2018). Therefore, we investigate the effect of using code-mixing (both NCM and SCM) in $\mathcal{M}$-$\mathcal{E}2$ as a cross-lingual data augmentation technique for SA tasks on code-mixing sentences in $\mathcal{M}$-$\mathcal{E}1$. Similar to the Language-Specific method above, this approach leverages labeled data outside of the limited $\mathcal{M}$-$\mathcal{E}1$ domain: $\mathcal{M}$-$\mathcal{E}2$ NCM uses labeled, yet still limited, NCM data in $\mathcal{M}$-$\mathcal{E}2$, and $\mathcal{M}$-$\mathcal{E}2$ SCM uses labeled monoligual data in $\mathcal{M}$ similar to the language-specific generation process above. This cross-lingual method can be used in conjunction with the language-specific SCM method described in Section 3.1.

Language-Agnostic Finally, we abandon any semantic information in $\mathcal{E}$ so the model focuses more on learning the pattern of code-switching rather than the semantics of individual words. To do this, we replace the embedded language tokens with a constant mask: `<GIB>`, and create SCM datasets in the embedding-agnostic space $\mathcal{M}$-$\mathcal{E}_{<GIB>}$. This data generation method is extremely low-cost, as the labeled SA monolingual dataset in $\mathcal{M}$ is abundant. But more importantly, the translation into `<GIB>` is zero-cost - with no additional runtime or noise added during this trivial translation process.

As shown in Figure 2, our method is the first to leverage cross-lingual $\mathcal{M}$-$\mathcal{E}2$ and language-agnostic $\mathcal{M}$-$\mathcal{E}_{<GIB>}$ data augmentation for code-mixing (dotted red box) in addition to the language-specific augmentation $\mathcal{M}$-$\mathcal{E}1$ (dotted black box). The synthetic data is mixed with NCM in one-shot or without NCM in zero-shot settings to train different SA models in the second stage. We evaluate the SCM datasets on the labeled Hinglish NCM SA test set of 3000 sentences, as well as different language pairs and across different low-resource natural training dataset sizes.

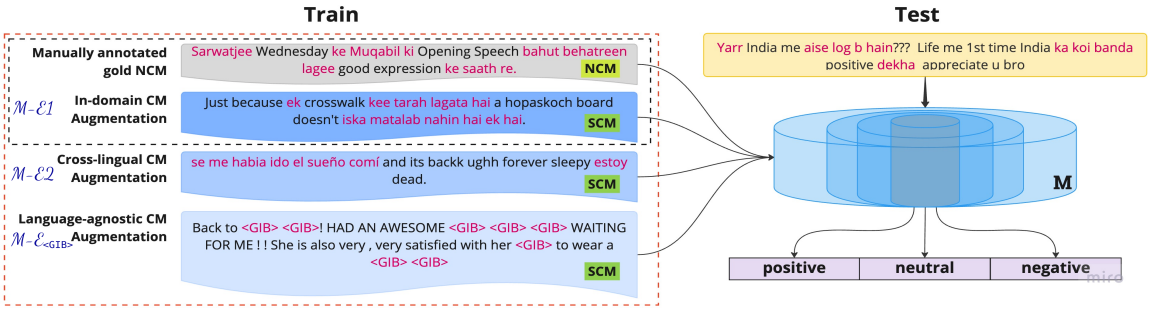

Figure 2: Different Synthetic Data Augmentation Methods. Inside the dotted black rectangular box are current data augmentation methods from the literature; inside the dotted red rectangular box is our novel data augmentation. The data (natural and synthetic) is used to fine-tune the SA model $\mathcal{M}$, which outputs one of the three sentiment labels.

3.2 Code-Mixing Generation - SCM

For both the language-specific and language-agnostic above, the data augmentation relies on effective SCM generation methods. In this section, we investigate two replacement-based algorithms to produce SCM sentences of any given language pair $\mathcal{M}$-$\mathcal{E}$. Section 3.2.1 introduces ways to create synthetic code-mixing sentences by replacing select tokens from the source sentence; in Section 3.2.2, we take advantage of additional syntactic information to generate SCM by replacing phrases with Part-of-speech (POS) tagging.

3.2.1 Lexical Replacement

For each monolingual sentence in the matrix language $\mathcal{M}$, a word-level alignment translator translates select words into the embedded language $\mathcal{E}$ and a SCM sentence is generated by replacing those words from the matrix language to the embedded language.

Code-Mixing Index For a given language pair $(\mathcal{M}, \mathcal{E})$, we first calculate the Code-Mixing Index (CMI) (Gambäck and Das, 2014) as follows:

$$CMI = 100(1 - \frac{\max(w_i)}{n-u}) \text{ if } n > u, \text{ else } 0 \quad (1)$$

where w_i is the number of tokens for language i, n is the total number of tokens and u is the number of language-independent tokens. The CMI measures the degree of code-mixing in a sentence when comparing different code-mixed corpora to each other. When selecting tokens from the source sentence, we match the number of tokens to the CMI of the NCM data using a hyperparameter Temperature (τ) so that the synthetic data has a similar distribution to the natural data.

Token Selection As a strong SCM baseline, we randomly select tokens so that the CMI of the SCM matches that of the NCM corpus inspired by Krishnan et al. (2021), and we name this replacement method *random word replacement*. Next, since switching points happens at the phrasal-level (Bokamba, 1989), we select random phrases in the source sentence to be replaced by the embedded language. This *random phrase replacement* algorithm is described in more detail in Appendix B.

Word level translation After token selection, we use a word-level translation method to replace the selected tokens in $\mathcal{M}$ with tokens in $\mathcal{E}$ to generate the final SCM sentence. We investigate a word-level alignment method and fine-tuning mBART (Liu et al., 2020b) to translate the phrases as described in Appendix D. The translated phrase tokens are put back to the original position in the sentence of the matrix language to produce the final code-mixing sentence. Figure 3 shows the SCM generated with word and phrase-level lexical replacement of the language pair $\mathcal{M}$-$\mathcal{E}_{<GIB>}$.

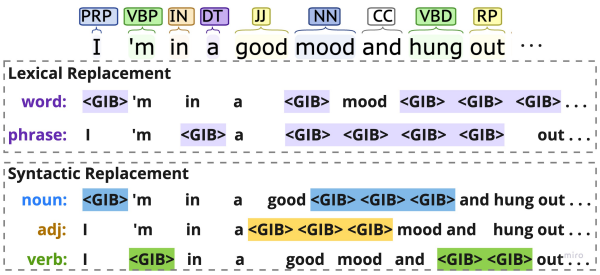

Figure 3: SCM Generation
The lexical and syntactic algorithms to select parts of the sentence in $\mathcal{M}$ and replace them with tokens in $\mathcal{E}$.

3.2.2 Syntactic Replacement

Code-mixing is such a complex linguistic and social phenomenon that it is hard to come up with any universal syntactic constraints to formalize the switching of languages (Bokamba, 1989). Instead of randomly selecting words and phrases as in Section 3.2.1, we utilize more syntactic information to synthesize code-mixing sentences by tagging the POS of each word in the source sentence. We use

the Flair monolingual English POS tagger (Akbik et al., 2018) with an accuracy of 97.85%. Let p be the pre-terminals from the Penn Treebank, our Syntactic Replacement Algorithm (more details in Appendix C) returns a corpus $\mathcal{S}_p$, which is the dataset created by replacing all words with POS tag p from the source sentences with translation in the embedded language. And finally we take

$$\mathcal{S}_{cm} = \mathcal{S}_{p_1} \cup \mathcal{S}_{p_2} \cup \ldots \cup \mathcal{S}_{p_k} \quad (2)$$

as the combined dataset of all the above variations. We expect these to be a more effective data augmentation. The lower half of Figure 3 shows the SCM with p_i = noun, adj, and verb, respectively.

4 Experimental Setup

4.1 Datasets

SA is the main task of our data augmentation and we use both the accuracy and weighted F1-score (for 3-way classification) to evaluate the predictions. The dataset used for training and testing is the SemEval-2020 Task 9 on Sentiment Analysis of Code-Mixed Tweets (Patwa et al., 2020)[1], which consists of 14000, 3000, and 3000 Hinglish code-mixing sentences for training, we call this dataset Natural Code Mixing (NCM). Next, the source of the SCM generation is taken from the Stanford Sentiment140 dataset (Go et al., 2009) [2], which consists of monolingual English tweets labeled with positive and negative sentiments. Finally, we use the English-Hindi bitext from IITB (Kunchukuttan et al., 2018) for the word alignment dictionary and for mining neutral SCM source sentences.

To explore the cross-lingual generalizability of our data augmentation, we use the Spanglish data in the SemEval dataset (train: 9002, eval: 3000, test: 3000) and an English-Malayalam code-mixing dataset (train: 3000, eval: 1452, test: 1000) labeled for SA (Chakravarthi et al., 2020a) as cross-lingual evaluation test sets.

4.2 Preprocessing

A coarse filtering (Liu et al., 2020a) is performed on both the SemEval NCM data and the Sentiment140 English data.Empty strings, hash symbols, and URLs are removed from the text. All emojis and emoticons are replaced by their English descriptions using the emoji library[3].

[1] https://ritual-uh.github.io/sentimix2020/

[2] http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip

[3] https://pypi.org/project/emoji/

Since the Sentiment140 dataset for SCM generation only contains binary sentiments, we use roBERTa-base finetuned on English Twitter SA (Barbieri et al., 2020) to mine neutral sentences in English from the IITB parallel English-Hindi corpus. During mining, we collect sentences that are classified as neutral with a confidence score higher than 0.85 into the final source language dataset.

4.3 Training

To make the data augmentation more effective, we adopt a gradual fine-tuning approach (Xu et al., 2021). Treating the SCM data as out-of-domain data, since it does not have the exact distribution as the human-produced NCM sentences, we iteratively fine-tune on the mixed SCM and NCM with decreasing amounts of out-of-domain data. This gradual fine-tuning approach allows the model to better fit the distribution of the target domain, as the training data gets more and more similar to the domain of the test data, which is NCM in our case. We fine-tune the model in 5 stages with the amount of SCM data size of $[30k, 10k, 3k, 1k, 0]$ with 3 epochs in each stage. An embedding length of 56 is used for the Hinglish corpus and 40 for the Spanglish corpus following the baseline method in Patwa et al. (2020). AdamW optimizer is used with a linear scheduler and a lr of 4e-6, which is determined empirically by preliminary experimentation. All training was done on Google Colab GPU and takes approx. 65 GPU minutes per experiment.

4.4 Multilingual PLMs

To stay consistent with the baseline accuracy provided in Patwa et al. (2020), we primarily evaluate our synthetic data augmentation on mBERT with a transformer classifier, but we also explore different models (XLM-R and XLM-T) in our ablation studies. As shown in Figure 2, our method fine-tunes a multilingual PLM with a combination of natural and synthetic data of the same language pair as the previous work has done, then we expand into cross-lingual and language-agnostic data augmentation. Thus, the number of parameters of our model stays consistent with the choice of PLM. In order to directly compare the effectiveness of different SCM datasets, we fix the size of the augmented data throughout our experiments at 30000 sentences and the size of the natural data at 3000 sentences. We evaluate our data augmentation on the labeled Hinglish NCM SA test set of 3000 sentences and repeat all experiments for 5 trials.

4.5 Baselines and Hyperparameters

In order to highlight the effect of our data augmentation method, we use the same model used in the shared task baseline (Patwa et al., 2020) and we were able to reproduce the baseline F1-score of 0.65. The model uses mBERT (Devlin et al., 2018) and a transformer classification layer to predict the sentiment to be positive, negative, or neutral. To demonstrate the effectiveness of our data augmentation in low-resource settings, we cut down the training dataset to 3000 sentences for all future experiments. A temperature value of $\tau = 0.4$ is used to during SCM generation. The hyperparameter tuning process is described in Appendix A.

In our 5-stage gradual fine-tuning procedure, the NCM data is passed through the model $5 \times 3 = 15$ times (3 epochs per stage), so we consider two candidates for the baseline model without data augmentation. We fine-tune the model for 3 epochs (1 stage) and 15 epochs (5 stages) with only NCM and compare the model performance. As shown in exp #1 and #12 in Table 2, the two procedures produced results that are not statistically different, and a converging loss curve indicates that the 5-stage fine-tuning will lead to over-fitting. Therefore, we use the one-stage fine-tuning on the NCM as a baseline for comparisons. Additionally, we propose a strong baseline of using the source monolingual sentences in $\mathcal{M}$ that we use to generate the SCM as a data augmentation to fine-tune the model (exp # 13). This allows us to separate the effect of the code-switch pattern from the sentence semantics.

5 Results and Analysis

We first compare the SCM results with different language pair selection schemes in Section 5.1, and then discuss multiple strong baselines and compare different lexical replacement variants in Section 5.2. In the ablation studies, we examine different syntactic replacement variants, effectiveness across low-resource settings, and the generalizability of our SCM across different PLMs.

5.1 Cross-lingual SCM Data Augmentation

There are two SCM datasets being evaluated in this experiment - English-Hindi SCM generated using the phrase level lexical replacement algorithm (hiSCM) and English-XX SCM generated using the syntactic replacement algorithm (gibSCM). In addition to the English-Hindi NCM, we test on two previously unseen English-Spanish and English-Malayalam NCM datasets. Spanish and English have a small Levenshtein distance (Serva and Petroni, 2008), while Malayalam and English come from different language families. Additionally, it is important to note that although both Hindi and Malayalam are widely spoken in India, Hindi is in the Indo-European language family whereas Malayalam is from the Dravidian language family (Emeneau, 1967).

First, the baselines in rows 1, 4, and 7 of Table 1 are the weighted F1-scores obtained by fine-tuning mBERT on *only* the NCM dataset. The in-domain language-specific data augmentation is evaluated by fine-tuning on the combined English-Hindi NCM and SCM as shown in exp #2. Then, we have the cross-lingual experiments that uses English-Hindi SCM as a data augmentation for English-Spanish and English-Malayalam tasks as shown in exp #5 and #8, respectively. Finally, our novel language-agnostic augmentation of masked English-XX code-mixing is tested with all language pairs as shown in exp #3, #6, and #9.

#	Language	Data	SCM	↑ (%)
1	Hindi	hiNCM	0.5505	0
2	Hindi	+hiSCM	**0.5860**	6.45
3	Hindi	+gibSCM	0.5853	6.32
4	Spanish	esNCM	0.4956	0
5	Spanish	+hiSCM	0.5053	1.96
6	Spanish	+gibSCM	**0.5061**	2.12
7	Malayalam	mlNCM	0.6703	0
8	Malayalam	+hiSCM	0.7202	7.45
9	Malayalam	+gibSCM	**0.7221**	7.73

Table 1: Cross-Lingual Data Augmentation
Synthetic data generated by translating select tokens into either Hindi or the <GIB> mask. '+' means combining synthetic data with NCM for gradual fine-tuning.

As shown in Table 1, the synthetic English-Hindi corpus is extremely helpful for all language pairs. It is important to note that the language-agnostic gibSCM augmentation (exp #6 and #9) achieved better performance compared to the English-Hindi SCM augmentation (exp #5 and #8) on the cross-lingual datasets of English-Spanish and English-Malayalam. This implies that Hindi phrases in the English-Hindi SCM might be biasing the model during training, making the language-agnostic data a more powerful universal data augmentation.

5.2 English-Hindi Lexical Replacement

In this section, we present a few baselines and evaluate the English-Hindi language-specific SCM generated using the Lexical Replacement Algorithm

(Algorithm 1). Weighted F1-scores over 5 repeated runs are shown in Table 2.

Baselines When there is no available labeled data, fine-tuning on *only* the synthetic data (exp #11) achieved better results than the zero-shot performance (exp #10) on the un-fine-tuned mBERT. This indicates that SCM is an effective zero-resource data augmentation for code-mixing SA. As described in Section 4.5, exp #1 and #12 report the one-stage and five-stage fine-tuning on the in-domain NCM data, which returns similar results. Finally, exp #13 shows the model performance when fine-tuned on the monolingual source English data used to generate the SCM.

SCM performance The more complex phrase-level lexical replacement (exp #2) achieves a more significant improvement than the naive random word lexical replacement (exp #14). This supports our intuition that code-mixing is more complex than just a random combination of words from different languages, in which a bilingual speaker picks any word that comes to mind. Rather, intrasentential code-mixing happens at the phrase level.

#	Experiment	Weighted F1
10	Zero-shot mBERT	0.1696 ± 0.0248
11	Zero-shot SCM	0.3144 ± 0.0787
1	Baseline (one-stage)	0.5505 ± 0.0041
12	Baseline (five-stage)	0.5518 ± 0.0075
13	Baseline (NCM + en)	0.5643 ± 0.0124
14	NCM +hiSCM (word)	0.5719 ± 0.0099
2	NCM +hiSCM (phrase)	**0.5860** ± 0.0112

Table 2: Hinglish Lexical Replacement
SCM data is made by translating select words/phrases from English to Hindi. '+' means combining synthetic data with NCM for gradual fine-tuning.

5.3 Ablation Studies

5.3.1 English-XX Syntactic Replacement

After observing the cross-lingual effectiveness of our SCM algorithms, we attempt to remove the semantic information of the embedded language completely by using a constant <GIB> mask in place of translated Hindi tokens. This way, the model learns the pattern of intrasentential code-switching rather than simply gathering the semantics of the lexicons. Additionally, this reduces the cost of SCM generation as no translation process is involved. Table 3 shows the classification accuracy with English-XX SCM datasets generated with the Lexical Replacement Algorithm and the Syntactic Replacement Algorithm with different token-selection strategies and their performance. The Syntactic Replacement Algorithm with a mixture of the POS tags "translated" into <GIB> (as described in Eq 2) outperformed the baselines and all other SCM algorithms, achieving a 6.32% relative improvement over the NCM baseline performance. This provides a language-agnostic data augmentation method for any code-mixing languages with English as the matrix language, which is the most common language in code-mixing sentence pairs in social media (Thara and Poornachandran, 2018).

#	Experiment	F1	↑ (%)
1	Baseline (only NCM)	0.5505	0
13	Baseline (NCM + en)	0.5643	2.51
15	Lexical (word)	0.5686	3.29
16	Lexical (phrase)	0.5664	2.88
17	Syntactic (Adj)	0.5633	2.33
18	Syntactic (Verb)	0.5723	3.97
19	Syntactic (Noun)	0.5786	5.10
3	Syntactic (mixed)	**0.5853**	**6.32**

Table 3: SCM Generation Algorithms
Synthetic English-<GIB> SCM generated by lexical and syntactic strategies with slightly different token selection methods: `Algorithm(replacement)`. The stronger performance of the <GIB> SCM variants over the baseline of fine-tuning on the English source sentences (exp #12) shows the effectiveness of *only* the code-switch pattern as a data augmentation.

5.3.2 Low Resource Levels

The difficulty to collect code-mixing data, especially in the text form, makes this prevalent linguistic phenomenon common in real life yet scarce in NLP research. We artificially limit the size of training data to create extremely low-resource scenarios, while keeping the test set constant to evaluate the effect of SCM augmentation across various low-resource settings. Figure 4 shows the weighted F1-scores for SA on the English-Hindi NCM test set of 3000 sentences when gradually fine-tuned on a combination of varying amounts of NCM training data and 30000 SCM augmented data. The SCM data augmentation is helpful for training across all low-resource levels. As the size of natural training data becomes more and more limited, our data augmentation becomes significantly more effective. The F1-scores of this experiment can be found in Appendix F.

In the extremely low resource case (100 NCM

sentences), fine-tuning the model with SCM improves the F1-score from 0.3062 to 0.4743, resulting in a 54.9% relative improvement. As more labeled NCM training data becomes available, both the baseline and the data-augmented models become better at predicting the correct label. Even in the highest-resource setting with 14000 labeled NCM sentences, the data augmentation still produces a 2.08% relative improvement.

Fine-tuning on only the large NCM corpus of 14000 sentences achieves a weighted F1-score of 0.6543, which outperforms the SCM data augmentation in the extreme low-resource scenario. These results show that human-produced gold data is still undoubtedly superior to synthetic data, but synthetic data is still helpful in the absence of natural data. We see a smoother loss curve during training when the NCM data size is large (i.e. when the SCM/NCM ratio is smaller), attributing to the domain consistency of the NCM data. This indicates further opportunities to stabilize the synthetic data style so that the model's search path has less noise.

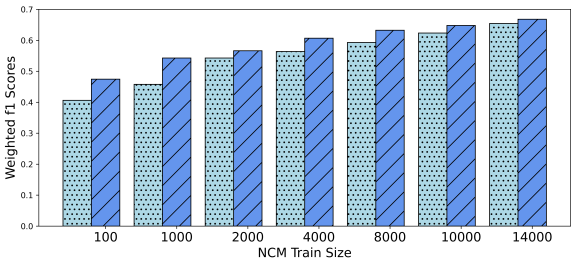

Figure 4: Effect of SCM Across Low Resource Levels
Weighted F1-scores with SCM English-XX Lexical Replacement data augmentation across different low-resource levels. The SCM is more effective with decreased natural dataset size.

5.3.3 Generalizability Across Models

We use our data augmentation method on more powerful models to evaluate the generalizability of the SCM corpus across different models. Table 4 shows the baseline of one-shot fine-tuning on NCM vs. SCM performance of mBERT (Devlin et al., 2018), XLM-R (Conneau et al., 2019), and XLM-T (Barbieri et al., 2022) on the Hinglish SA task. XLM-R performs extremely well on low-resource languages, and XLM-T is an XLM-R-based model pre-trained on millions of tweets in over thirty languages.

The data augmentation improves model performance for all pre-trained multilingual language models. This indicates that even powerful multilingual PLMs are not able to completely capture the complex structure of code-mixing languages.

#	Model	Baseline	+gibSCM	↑ (%)
1	mBERT	0.5505	0.5853	**6.32**
20	RoBERTa-T	0.5982	0.6274	4.88
21	XLM-R	0.6034	0.6564	5.30
22	XLM-T	0.6491	**0.6600**	1.68

Table 4: Generalizability across Models
gibSCM focuses on teaching code-switch patterns; it outperform one-shot NCM baselines on all PLMs.

Although the improvement becomes smaller as the model gets more complex, the data augmentation methods consistently outperform the strong baselines. The smaller improvement of our data augmentation technique on XLM-R and XLM-T might also result from the fact that they are trained on Common-Crawl and Twitter data, which contains code-mixing sentences occasionally. And XLM-T has already seen a large amount of Twitter data. Overall, the synthetic data mimics the pattern of code-switching and helps the model adjust to the in-domain code-mixing training data.

5.3.4 Other Generation Methods

All our methods above rely on replacement-based generation methods. Howver, there are other data augmentation techniques such as switching point prediction, machine translation based approach, and code mixing language modeling. As our work focuses on the cross-lingual and language-agnostic nature of the SCM, other data augmentation algorithms were not explored. In preliminary experiments, we found that the language-agnostic design also works on other statistical and neural data generation techniques such as n-gram language modeling, demonstrating that the language choice, including $\mathcal{M}$-$\mathcal{E}_{<GIB>}$, is technique-agnostic. We describe some of these techniques and results in Appendix E.

6 Conclusion

In this work, we introduce two replacement-based algorithms for synthetic code-mixing for training data augmentation. Most importantly, we prove a low-cost, language-agnostic solution to the data scarcity problem of code-mixing corpus for NLP tasks, specifically sentiment analysis. Analyzing the performance of the SCM as a data augmentation gives insight into code-mixing as a complex linguistic phenomenon. Our algorithm is flexible enough to be easily extended to any other replacement strategies, making it a universal framework for future explorations of the pattern of code-mixing languages.

7 Limitations

We cannot control the quality of the original monolingual source data mined from Twitter as we use the mined Sentiment140 dataset. Second, we cannot empirically verify that no data pollution is present (i.e. if the source sentence for generating the SCM was involved in the pre-training corpus of mBERT). Finally, the parent languages used in our experiments are either originally in Roman script or can be transliterated into Roman script. Our language-agnostic methods are not currently tested on languages with various scripts due to the absence of labeled data.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Peter Auer and Raihan Muhamedova. 2005. Embedded language'and 'matrix language'in insertional language mixing: Some problematic cases. *Rivista di linguistica*, 17(1):35–54.

Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 258–266.

Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *arXiv preprint arXiv:2010.12421*.

Eyamba G Bokamba. 1988. Code-mixing, language variation, and linguistic theory:: Evidence from bantu languages. *Lingua*, 76(1):21–62.

Eyamba G Bokamba. 1989. Are there syntactic constraints on code-mixing? *World Englishes*, 8(3):277–292.

Bharathi Raja Chakravarthi, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P McCrae. 2020a. A sentiment analysis dataset for code-mixed Malayalam-English. In *Proceedings of the 1st Joint Workshop of SLTU (Spoken Language Technologies for Under-resourced languages) and CCURL (Collaboration and Computing for Under-Resourced Languages) (SLTU-CCURL 2020)*, Marseille, France. European Language Resources Association (ELRA).

Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyadharshini, and John P McCrae. 2020b. Corpus creation for sentiment analysis in code-mixed tamil-english text. *arXiv preprint arXiv:2006.00206*.

Joyce YC Chan, Houwei Cao, PC Ching, and Tan Lee. 2009. Automatic recognition of cantonese-english code-mixing speech. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 14, Number 3, September 2009*.

Will Chang, David Hall, Chundra Cathcart, and Andrew Garrett. 2015. Ancestry-constrained phylogenetic analysis supports the indo-european steppe hypothesis. *Language*, pages 194–244.

Monojit Choudhury. 2018. Cracking code-mixing — an important step in making human-computer interaction more engaging.

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*.

Glen Coppersmith, Ryan Leary, Patrick Crutchley, and Alex Fine. 2018. Natural language processing of social media as screening for suicide risk. *Biomedical informatics insights*, 10:1178222618792860.

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.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Suman Dowlagar and Radhika Mamidi. 2021. Cmsaone@ dravidian-codemix-fire2020: A meta embedding and transformer model for code-mixed sentiment analysis on social media text. *arXiv preprint arXiv:2101.09004*.

Zulfadzli Drus and Haliyana Khalid. 2019. Sentiment analysis in social media and its application: Systematic literature review. *Procedia Computer Science*, 161:707–714.

Murray B Emeneau. 1967. The south dravidian languages. *Journal of the American Oriental Society*, 87(4):365–413.

Atefeh Farzindar and Diana Inkpen. 2015. Natural language processing for social media. *Synthesis Lectures on Human Language Technologies*, 8(2):1–166.

Björn Gambäck and Amitava Das. 2014. On measuring the complexity of code-mixing. In *Proceedings of the 11th International Conference on Natural Language Processing, Goa, India*, pages 1–7.

Devansh Gautam, Kshitij Gupta, and Manish Shrivastava. 2021. Translate and classify: Improving sequence level classification for english-hindi code-mixed data. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 15–25.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Judy Woon Yee Ho et al. 2007. Code-mixing: Linguistic form and socio-cultural meaning. *The International Journal of Language Society and Culture*, 21(7):1–8.

Andrei Hodorog, Ioan Petri, and Yacine Rezgui. 2022. Machine learning and natural language processing of social media data for event detection in smart cities. *Sustainable Cities and Society*, 85:104026.

Navya Jose, Bharathi Raja Chakravarthi, Shardul Suryawanshi, Elizabeth Sherly, and John P McCrae. 2020. A survey of current datasets for code-switching research. In *2020 6th international conference on advanced computing and communication systems (ICACCS)*, pages 136–141. IEEE.

Suin Kim, Ingmar Weber, Li Wei, and Alice Oh. 2014. Sociolinguistic analysis of twitter in multilingual societies. In *Proceedings of the 25th ACM conference on Hypertext and social media*, pages 243–248.

Jitin Krishnan, Antonios Anastasopoulos, Hemant Purohit, and Huzefa Rangwala. 2021. Multilingual code-switching for zero-shot cross-lingual intent prediction and slot filling. *arXiv preprint arXiv:2103.07792*.

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).

Jiaxiang Liu, Xuyi Chen, Shikun Feng, Shuohuan Wang, Xuan Ouyang, Yu Sun, Zhengjie Huang, and Weiyue Su. 2020a. Kk2018 at semeval-2020 task 9: Adversarial training for code-mixing sentiment classification. *arXiv preprint arXiv:2009.03673*.

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

Carol Myers-Scotton. 1992. Comparing codeswitching and borrowing. *Journal of Multilingual & Multicultural Development*, 13(1-2):19–39.

Monika Obrocka, Charles Copley, Themba Gqaza, and Eli Grant. 2019. Prevalence of code mixing in semi-formal patient communication in low resource languages of south africa. *arXiv preprint arXiv:1911.05636*.

Alvaro Ortigosa, José M Martín, and Rosa M Carro. 2014. Sentiment analysis in facebook and its application to e-learning. *Computers in human behavior*, 31:527–541.

Xuan Ouyang, Shuohuan Wang, Chao Pang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-m: Enhanced multilingual representation by aligning cross-lingual semantics with monolingual corpora. *arXiv preprint arXiv:2012.15674*.

Oladapo Oyebode, Chinenye Ndulue, Dinesh Mulchandani, Banuchitra Suruliraj, Ashfaq Adib, Fidelia Anulika Orji, Evangelos Milios, Stan Matwin, and Rita Orji. 2022. Covid-19 pandemic: Identifying key issues using social media and natural language processing. *Journal of Healthcare Informatics Research*, 6(2):174–207.

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

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas Pykl, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the fourteenth workshop on semantic evaluation*, pages 774–790.

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.

Libo Qin, Minheng Ni, Yue Zhang, and Wanxiang Che. 2020. Cosda-ml: Multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp. *arXiv preprint arXiv:2006.06402*.

Pattabhi RK Rao and Sobha Lalitha Devi. 2016. Cmeeil: Code mix entity extraction in indian languages from social media text@ fire 2016-an overview. *FIRE (Working Notes)*, 289.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do hindi-english speakers do on twitter? In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1131–1141.

Maurizio Serva and Filippo Petroni. 2008. Indo-european languages tree by levenshtein distance. *EPL (Europhysics Letters)*, 81(6):68005.

Vivek Srivastava and Mayank Singh. 2020. Phinc: A parallel hinglish social media code-mixed corpus for machine translation. *arXiv preprint arXiv:2004.09447*.

Vivek Srivastava and Mayank Singh. 2021. Hinge: A dataset for generation and evaluation of code-mixed hinglish text. *arXiv preprint arXiv:2107.03760*.

S Thara and Prabaharan Poornachandran. 2018. Code-mixing: A brief survey. In *2018 International conference on advances in computing, communications and informatics (ICACCI)*, pages 2382–2388. IEEE.

Haoran Xu, Seth Ebner, Mahsa Yarmohammadi, Aaron Steven White, Benjamin Van Durme, and Kenton Murray. 2021. Gradual fine-tuning for low-resource domain adaptation. *arXiv preprint arXiv:2103.02205*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

A Hyperparameter Tuning for CMI

Figure 5 shows the CMI of the synthetic data generated by the phrase level replacement algorithm using different temperature values τ. Upon closer observation, the natural data contains more Hindi tokens than English tokens, so a τ value to the right of the peak should be chosen if we use English as the source language in the generation algorithm because a higher temperature means more tokens will be selected to be translated into the embedded language. When $\tau = 0.4$, the CMI of the SCM is the closest to the CMI of the NCM.

B Lexical Replacement Algorithm

Algorithm 1 shows the lexical replacement technique in generating SCM.

C Syntactic Replacement Algorithm

Algorithm 2 shows the syntactic replacement technique in generating SCM.

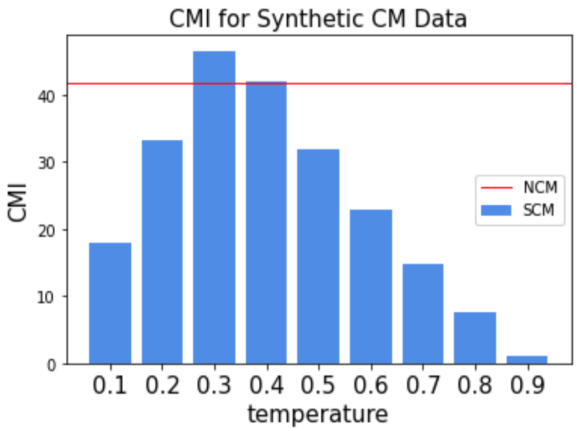

Figure 5: Temperature Tuning for CMI

Algorithm 1: Lexical Replacement

Data: $\mathcal{S}$
Result: $\mathcal{S}_{cm}$
for *s in* $\mathcal{S}$ **do**
  $s' \leftarrow$ '';
  $cr \leftarrow 0$;
  while *cr < len(s)* **do**
    if *random()* $< \tau$ **then**
      $L \leftarrow rand(1, 2, 3)$;
      $phrase \leftarrow$ '';
      $r \leftarrow 0$;
      for $cr < len(s)$ *and* $r < L$ **do**
        phrase += s[curr];
        curr += 1;
        r += 1;
      end
      s' += translate(phrase);
    else
      s' += s[cr];
      cr += 1;
    end
  end
  $\mathcal{S}_{cm}$ += s';
end

D Phrase-Level Translation

We experiment with two different word-level translation methods in this section. First, we create a simple one-to-many weighted English-Hindi dictionary using Awesome-Align (Dou and Neubig, 2021)[4]. Take the Hinglish code-mixing, for example, we generate word-level alignments using an English-Hindi parallel corpus. For each English

[4] https://pypi.org/project/awesome-align/

Algorithm 2: Syntactic Replacement

Data: $\mathcal{S}$
Result: $\mathcal{S}_p$
for *s in* $\mathcal{S}$ **do**
  $s' \leftarrow$ '';
  for *word in s* **do**
    if *POS(word)* == *p* **then**
      word' $\leftarrow$ translate(word);
      s' += word';
    else
      s' += word;
    end
  end
  $\mathcal{S}_p$ += s';
end

token, the aligned Hindi word is collected to create a weighted list. For the tokens to be replaced in the source sentence, an aligned Hindi word is randomly selected from the weighted word-level dictionary. The second translation method uses mBART (Liu et al., 2020b) to translate the English phrases into Hindi.

E N-gram SCM Generation

We train n-gram language models on the limited NCM data with the assumption that a model trained on positively labeled sentences will be a "positive model" that generates positive code-mixing sentences. The generation capability is limited by the small training corpus size, and there is a trade-off between n in an n-gram language model and the diversity of the sentence generated. Therefore, for each of the positive, neutral, and negative subsets of the training corpus, we train trigram, 4-gram, 5-gram, and 6-gram language models with back-off and add-lambda smoothing. We combine the generated sentences to take advantage of both the quality and diversity of the generated data.

#	Experiment	F1	↑ (%)
1	Baseline (only NCM)	0.5505	0
23	Ngram (combined)	0.5540	0.63

Table 5: NGram Language Model SCM Generation

F Low Resource Levels

Table 6 shows the weighted F1-score of the $\mathcal{M}$-$\mathcal{E}_{<GIB>}$ SCM across situations with different amount of available labeled NCM training data and the relative improvement over each perspective baseline without data augmentation.

#	|NCM|	Baseline	+gibSCM	↑ (%)
24	14000	0.6543	**0.6679**	2.08
25	12000	0.6370	0.6613	3.81
26	10000	0.6234	0.6477	3.89
27	8000	0.5929	0.6322	6.63
28	6000	0.5782	0.6152	6.40
29	4000	0.5638	0.6068	7.63
5	3000	0.5505	0.5853	6.32
30	2000	0.5426	0.5662	4.35
31	1000	0.3578	0.5427	33.08
32	500	0.3130	0.5259	48.98
33	100	0.3062	0.4743	**54.90**

Table 6: Effect of SCM Across Low Resource Levels