Progressive Sentiment Analysis for Code-Switched Text Data

Anonymous ACL submission

Abstract

Multilingual transformer language models have recently attracted much attention from researchers and are used in cross-lingual transfer learning for many NLP tasks such as text classification and named entity recognition. However, similar methods for transfer learning from monolingual text to code-switched text have not been extensively explored mainly due to the following challenges: (1) Code-switched corpus, unlike monolingual corpus, consists of more than one language and existing methods can't be applied efficiently, (2) Code-switched corpus is usually made of resource-rich and low-resource languages and upon using multilingual pre-trained language models, the final model might bias towards resource-rich language. In this paper, we focus on code-switched sentiment analysis where we have a labelled resource-rich language dataset and unlabelled code-switched data. We propose a framework that takes the distinction between resource-rich and low-resource language into account. Instead of training on the entire code-switched corpus at once, we create buckets based on the fraction of words in the resource-rich language and progressively train from resource-rich language dominated samples to low-resource language dominated samples. Extensive experiments across multiple language pairs demonstrate that progressive training helps low-resource language dominated samples.

1 Introduction

Code-switching is the phenomena where the speaker alternates between two or more languages in a conversation. The lack of annotated data and diverse combinations of languages with which this phenomenon can be observed, makes it difficult to progress in NLP tasks on code-switched data. And also, the prevalance of different languages is different, making annotations expensive and difficult.

Intuitively, multilingual language models like mBERT (Devlin et al., 2019) can be used for code-switched text since a single model learns multilingual representations. Although the idea seems straightforward, there are multiple issues. Firstly, mBERT performs differently on different languages depending on their script, prevalence and predominance. mBERT performs well in medium-resource to high-resource languages, but is outperformed by non-contextual subword embeddings in a low-resource setting (Heinzerling and Strube, 2019). Moreover, the performance is highly dependent on the script Pires et al. (2019). Secondly, transformer models have only seen monolingual sentences during the unsupervised pretraining, however code-switched text contains phrases from both the languages in a single sentence, thus making it an entirely new scenario for the transformer models. Thirdly, there is difference in the languages based on the amount of unsupervised corpus that is used for transformer language models pretraining. For e.g., mBERT is trained on the wikipedia corpus. English has $\sim$ 6.3 million articles, whereas Hindi and Tamil have only $\sim$ 140K articles each. This may lead to under-representation of low-resource langauges in the final model. Further, English has been extensively studied by NLP community over the years, making the supervised data and tools more easily accessible. Thus, the model would be able to easily learn patterns present in the resource-rich language segments and motivating us to attempt transfer learning from English supervised datasets to code-switched datasets.

Original	fixing mein saja hone ka gift
Transliterated	fixing मे सजा होने का gift
Translated	gift of punishment for fixing

Figure 1: An example of code-switched text, its transliterated and the translation versions.

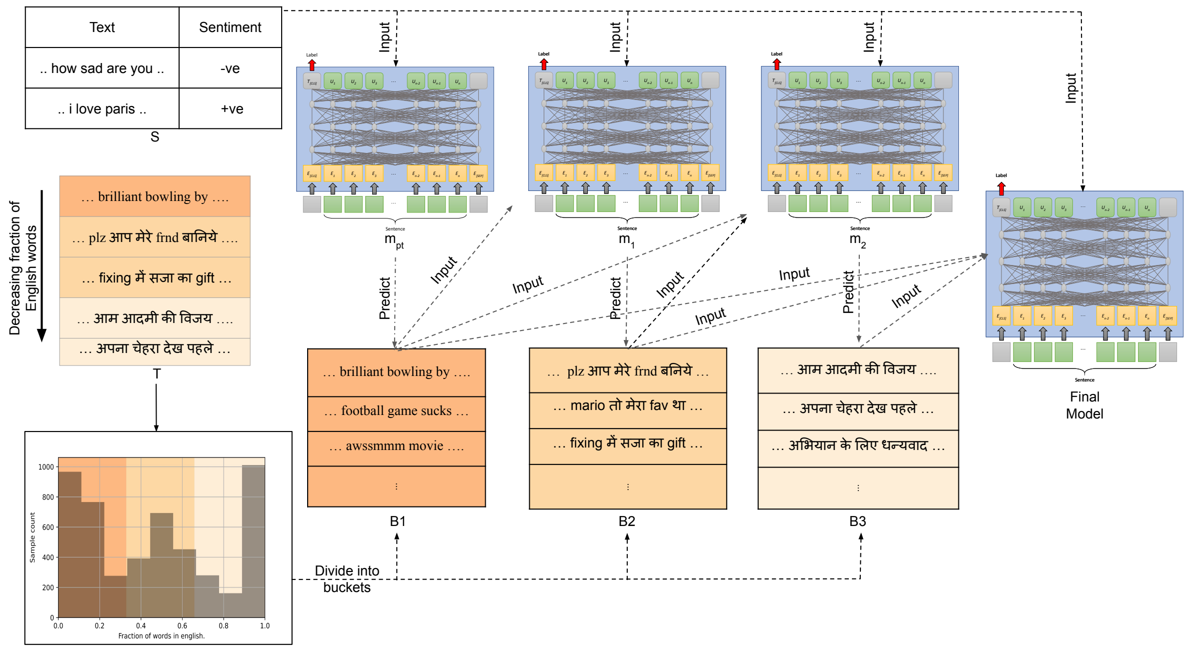

Figure 2: A visualization of the progressive training strategy. The source labelled dataset S in resource rich language should be easily available. Using S, a classifier is trained, say m_{pt}. Unlabelled code-switched dataset T is divided into buckets using the fraction of English words as the metric. The leftmost bucket B1 has samples dominated by resource-rich language and as we move towards right, the samples in the buckets are dominated by low-resource language. m_{pt} is used to generate pseudo-labels for unlabelled texts in bucket B1. We use texts from B1 along with their pseudo-labels and the dataset S to train a second text classifier m_1. Then, m_1 is used to get the pseudo-labels for texts in bucket B2. We keep repeating this until we obtain the final model which is used for predictions.

The main idea behind our paper can be summarised as follows: *When doing zero shot transfer learning from a resource-rich language (LangA) to code switched language (say LangA-LangB, where LangB is a low-resource language compared to LangA), the model is more likely to be wrong when the instances are dominated by LangB. Thus, instead of self-training on the entire corpus at once, we propose to progressively move from LangA-dominated instances to LangB-dominated instances while transfer learning*. Figure 2 illustrates the idea. Model trained on the annotated resource-rich language dataset is used to generate pseudo-labels for code-switched data. Progressive training uses the resource-rich language dataset and (unlabelled) resource-rich language dominated code-switched samples together to generate better quality pseudo-labels for the (unlabelled) low-resource language dominated code-switched samples. Lastly, annotated resource-rich language dataset and pseudo-labelled code-switched data are then used together for the training which increases the performance of the final model.

Our key contributions are summarized as:

- We propose a simple, novel training strategy that demonstrates superior performance. Since our hypothesis is based on the pretraining phase of the multilingual transformer models, it can be combined with any transfer learning method.
- We conduct experiments across multiple language-pair datasets, showing the efficiency of our proposed method.
- We create probing experiments that verify our hypothesis.

2 Related work

Multiple tasks like Language Identification, Named Entity Recognition, Part-of-Speech, Sentiment Analysis, Question Answering and NLI have been studied in the code-switched setting. For sentiment analysis, Vilares et al. (2015) showed that multilingual approaches can outperform pipelines of monolingual models on code-switched data. Lal et al. (2019) use CNN based network for the same. Winata et al. (2019) use hierarchical meta embeddings to combine multilingual word, character and sub-word embeddings for the NER task. Aguilar and Solorio (2020) augment morphological clues to language models and uses them for transfer learning from English to code-switched data with labels. Samanta et al. (2019) uses translation API to create synthetic code-switched text from English datasets and use this for transfer learning from English to code-switched text without labels in the code-switched case. Qin et al. (2020) use synthetically generated code-switched data to enhance zero-shot cross-lingual transfer learning. Recently, Khanuja et al. (2020) released the GLUECoS benchmark to study the performance of multiple models for code-switched tasks across two language pairs En-Es and En-Hi. The benchmark contains 6 tasks, 11 datasets and has 8 models for every task. Multilingual transformers fine tuned with masked-language-model objective on code-switched data can outperform generic multilingual transformers. Results from Khanuja et al. (2020) show that sentiment analysis, question answering and NLI are significantly harder than tasks like NER, POS and LID. In this work, we focus on the sentiment analysis task in the absence of labeled code-switched data using multilingual transformer models, while taking into account the distinction between resource-rich and low-resource languages.

3 Preliminaries

Our problem is a sentiment analysis problem where we have a labelled resource-rich language dataset and unlabelled code-switched data. From here onwards, we refer the labelled resource-rich language dataset as the source dataset and the unlabelled code-switched dataset as target dataset. Since code-switching often occurs in language pairs that include English, we refer to English as the resource-rich language. The source dataset, say S, is in English and has the text-label pairs $\{(x_{s_1}, y_{s_1}), (x_{s_2}, y_{s_2}), ...(x_{s_m}, y_{s_m})\}$ and the target dataset, say T, is in code-switched form and has texts $\{x_{cs_1}, x_{cs_2}, ...x_{cs_n}\}$, where m is significantly greater than n. The objective is to learn a sentiment classifier to detect sentiment of code-switched data by leveraging labelled source dataset and unlabelled target dataset.

4 Methodology

Our methodology can be broken down into three main steps: (1) Source dataset pretraining, which uses the resource-rich language labelled source dataset S for training a text classifier. This classifier is used to generate pseudo-labels for the target dataset T. (2) Bucket creation, which divides the unlabelled data T into buckets based on the fraction of words from resource-rich language. Some buckets would contain samples that are more resource-rich language dominated while others contain samples dominated by low-resource language. (3) Progressive training, where we initially train using S and the samples dominated by resource-rich language and gradually include the low-resource language dominated instances while training. For rest of the paper, pretraining refers to step 1 and training refers to the training in step 3. And, we also use class ratio based instance selection to prevent the model getting biased towards majority label.

4.1 Source Dataset Pretraining

Resource-rich languages have abundant resources which includes labeled data. Intuitively, sentences in T that are similar to positive sentiment sentences in S would also be having positive sentiment (and same for the negative sentiment). Therefore, we can treat the predictions made on T by multilingual model trained on S as their respective pseudo-labels. This would assign noisy pseudo-labels to unlabeled dataset T. The source dataset pretraining step is a text classification task. Let the model obtained after pretraining on dataset S be called m_{pt}. This model is used to generate the initial pseudo-labels and to select the instances to be used for progressive training.

4.2 Bucket Creation

Since progressive training aims to gradually progress from training on resource-rich language dominated samples to low-resource language dominated samples, we divide the dataset T into buckets based on fraction of words in resource-rich language. This creates buckets that have more resource-rich language dominated instances and also buckets that have more low-resource language dominated instances as well. In figure 2, we can observe that the instances in the leftmost bucket are dominated by the English, whereas the instances in the rightmost bucket are dominated by Hindi. More specifically, we define:

$$f_{eng}(x_i) = \frac{n_{eng}(x_i)}{n_{words}(x_i)}$$

where $n_{eng}(x_i)$ and $n\_words(x_i)$ denotes the number of English words and total number of words in the text x_i. Then, we sort the texts in dataset T in decreasing order of $f_{eng}(x_i)$ and create k buckets $(B_1, ..., B_k)$ with equal number of texts in each

Algorithm 1: Pseudocode for our progressive training framework.

```
Input: Source dataset $S$, target dataset $T$
Parameter: Selection fraction $\delta$, number of buckets $k$
Output: Predictions on target dataset $T$
// Backbone model
Model $m_{bb} \epsilon (mBERT, MuRIL, IndicBERT)$
Model $m_{pt} \leftarrow m_{bb}$ trained on dataset $S$
// Bucketing step
$T' \leftarrow ((f_{eng}(x_i), x_i, m_{pt}(x_i))$ for $x_i$ in $T)$
$T' \leftarrow reverse\_sorted(T')$
$(B_1, B_2, ..., B_k) \leftarrow$ divide $T'$ into $k$ equal buckets
// Class ratio based instance
   selection
$X_{class_q} \leftarrow$ Samples in $T'$ predicted to be in class $q$
$X_{st} \leftarrow \cup_{q=0}^{q=1}$ ($\delta$ most confident samples in $X_{class_q}$)
$X_{st_r} \leftarrow X_{st} \cap B_r$
// Progressive training step
Model $m_0 \leftarrow m_{pt}$
for i $= 1$ to $k$ do
    $T_i \leftarrow \cup_{r=1}^{r=i}((x, m_{r-1}(x))$ for $x$ in $X_{st_r})$
    Model $m_i \leftarrow m_{bb}$ trained on $S \cup T_i$
Return $m_k(T)$
```

bucket. Thus, bucket B_1 contains the instances mostly dominated by English language and as we move towards buckets with higher index, instances would be dominated by the low-resource language.

4.3 Progressive Training

As the model m_{pt} is obtained by fine-tuning on a resource-rich language dataset S, it is more likely to perform better on resource-rich language dominated instances. Therefore, we choose to start progressive training from resource-rich language dominated samples. However, note that the pseudo-labels generated for dataset T are noisy, thus we sample high confident resource-rich language dominated samples to obtain better quality pseudo-labels for the rest of the instances.

Firstly, we use m_{pt} to obtain all the high confidence samples from dataset T to be used for progressive training and their respective pseudo-labels. Among the samples to be used for progressive training, we select the samples from B_1 and use them along with S to train a second classifier which is further used to generate pseudo-labels for the rest of the samples to be used for progressive training. Then we select samples from B_2 and use them along with samples from previous iterations (i.e. samples selected from B_1 and S) to get a third classifier. We continue this process until we reach the last bucket and use the model obtained at the last iteration to make the final predictions.

More formally, we use m_{pt} to select the most confident δ fraction of samples from the dataset T, considering probability as the proxy for the confidence. Let X_{st} denote the δ fraction of samples with the highest probability of the majority class to be used for progressive training. Let $X_{st_i} = X_{st} \cap B_i$, where X_{st_i} is the subset of samples from bucket B_i that would be used for the progressive training. To train across k buckets, we use k iterations. Let m_j denote the model obtained after training for iteration j and m_0 refers to model m_{pt}. Iteration j is trained using texts $((\cup_{i=1}^{j} X_{st_i}) \cup S)$. The true labels for texts in S are available and for texts X_{st_i}, labels obtained using model m_{i-1} are considered as their respective labels. The model obtained at the last iteration i.e. m_k is used for final predictions.

4.4 Class ratio based instance selection

Datasets frequently have a significant amount of class imbalance, thus when selecting the samples for progressive training, we often end up selecting a very small amount or no samples from the minority class which leads to very poor performance. Hence, instead of selecting δ fraction of samples from the entire dataset T, we select δ fraction of samples per class. Specifically, let X_+ and X_- denote the set of samples for which the pseudo-labels are positive and negative sentiment respectively. For progressive training, we choose δ fraction of most confident samples from X_+ and δ fraction of most confident samples from X_-.

The pseudo-code for algorithm is shown in Algorithm 1.

5 Experiments

We describe the details relevant to the experiments in this section and also elaborate on the probing tasks.

5.1 Datasets

For source dataset pretraining, we use the English Twitter dataset from SemEval 2017 Task 4 (Rosenthal et al., 2017). We upsample the minority class to create a balanced dataset. We use three code-switched datasets for our experiments : Hindi-English (Patra et al., 2018), Spanish-English (Vilares et al., 2016), and Tamil-English (Chakravarthi et al.). Hindi-English, Spanish-English are collected from Twitter and the Tamil-English is collected from YouTube comments.

Most of the sentences in the datasets are written in the Roman transcript. We use the same preprocessing as done in GLUECoS for the first two datasets. For the other two datasets, we use the AI4Bharat Transliteration python library [1] to get the transliterations. The statistics of the dataset can be found in Table 1. Two out of the three datasets have a class imbalance, the maximum being in the case of Tamil-English where the positive class is ∼5x of the negative class.

5.2 Model training

In all the experiments we use multilingual-bert-base-cased (mBERT) for tokenization and training. The supervised English dataset has a 80-20 train-validation split. For pretraining with supervised English dataset, we use 4 epochs and choose the best model using the validation set. For training, we use 4 epochs and use the model at the end for the final evaluation. The batch size is 64, sequence length is 128 and learning rate is 5e-5. Every iteration takes approximately ∼1-2 seconds and ∼12 GB of memory on a GPU. For all the experiments the value of δ is set to 0.5 following Wang et al. (2021). We observe that in most datasets, the number of spikes in the distribution plot of $f_{eng}(x_i)$ is either 1 or 2. For example, we observe there are only two spikes for the Hindi-English dataset in Figure 7 in Appendix. Therefore, we set k=2. More details can be found in table 5 in Appendix.

In the rest of the paper, we refer to the model pretrained on the resource-rich language source dataset as model m_{pt}, the model trained on source dataset along with bucket B_1 as m_1, and the model trained on source dataset along with the buckets B_1 and B_2 as m_2. m_2 is the model used for final predictions.

5.3 Evaluation

As the datasets are significantly skewed between the two classes, we choose to report micro, macro and weighted f1 scores as done in Mekala and Shang (2020). For code-switched datasets, we use all the sentences without labels during the self-training. The final score is obtained using the predictions made by model m_2 on all the sentences and their true labels. For each dataset, we run the experiment with 5 seeds and report the mean and standard deviation.

[1]https://pypi.org/project/ai4bharat-transliteration/

Table 1: Statistics of datasets. SemEval2017 is a supervised English dataset and the rest are code-switched datasets.

Dataset	Total	Positive	Negative
SemEval2017	27608	19799	7809
Spanish-English	914	489	425
Hindi-English	6190	3589	2601
Tamil-English	10097	8484	1613

5.4 Baselines

We consider three baselines described below:

- Deep Embedding for Clustering (**DEC**) initially trains the model on the source dataset S, obtains pseudo-labels on unlabeled data in T and further trains on all the samples from unlabelled data along with their pseudo-labels using the soft labeling objective (Xie et al., 2016) as done in LOTClass (Meng et al., 2020).
- No Progressive Training (**No-PT**) initially trains the model on the source dataset S. As done in (Wang et al., 2021), it selects δ fraction of the code-switched data with pseudo-labels and trains a classifier on selected samples and the source dataset S without any progressive training.
- Unsupervised Self-Training (**Unsup-ST**) (Gupta et al., 2021) starts with a pretrained sentiment analysis model and then self-trains using code-switched dataset. We use the default version which doesn't require human annotations. We use the model m_{pt} to initiate the self-training for fair comparison.

We also compare with two ablation versions of our method, denoted by ***- Source*** and ***- Ratio***. The first method uses only the code-switched dataset with its corresponding pseudo-labels without the source dataset S for training. The second method chooses the most confident samples for training without taking the class ratio into account.

We also report the performance in the supervised setting, denoted by ***Supervised***. For each dataset, train the model only on dataset T but use true labels to do the same. This helps in getting a possible upper bound.

5.5 Performance comparison

The results for all the three datasets are reported in Table 2. In almost all the cases, we observe a performance improvement using our method as compared to the baselines, maximum improvement being upto $\sim 1.2\%$ in the case of Spanish-English. In

Table 2: Model performance on the Spanish-English, Hindi-English & Tamil-English dataset using mBERT. For Tamil-English dataset, the ***- Ratio*** method increases the F1 score of positive class (which is the majority class) by $\sim 2\%$ but F1 score of negative class drops by $\sim 9\%$. Thus, we observe a performance improvement in weighted F1 score and micro F1 score but a decrease in macro F1 score.

Methods	**Spanish-English**			**Hindi-English**			**Tamil-English**		
	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1
Supervised	71.1	71.5	71.6	74.7	74.9	75.1	62.7	84.3	82.8
DEC	75.7 ± 0.4	75.3 ± 0.3	75.5 ± 0.5	66.0 ± 0.5	67.9 ± 0.7	67.1 ± 0.4	51.7 ± 0.9	77.8 ± 0.8	75.5 ± 0.6
No-PT	76.2 ± 0.6	76.3 ± 0.6	76.3 ± 0.6	67.1 ± 0.8	69.2 ± 0.3	68.4 ± 0.6	53.3 ± 0.4	78.6 ± 0.6	76.7 ± 0.2
Unsup-ST	73.9 ± 1.7	74.3 ± 1.6	74.1 ± 1.7	66.6 ± 1.5	66.7 ± 1.4	66.8 ± 1.3	49.8 ± 2.1	69.2 ± 5.3	69.8 ± 8.4
Ours	$\mathbf{77.4 \pm 0.8}$	$\mathbf{77.5 \pm 0.8}$	$\mathbf{77.5 \pm 0.8}$	$\mathbf{67.8 \pm 0.8}$	$\mathbf{69.9 \pm 0.5}$	$\mathbf{69.1 \pm 0.7}$	$\mathbf{53.1 \pm 0.4}$	80.5 ± 0.2	77.5 ± 0.1
- Source	74.6 ± 1.4	74.6 ± 1.4	74.6 ± 1.5	67.2 ± 0.5	68.7 ± 0.4	68.3 ± 0.4	74.6 ± 1.4	74.6 ± 1.4	74.6 ± 1.5
- Ratio	77.1 ± 0.7	77.3 ± 0.6	77.2 ± 0.6	64.7 ± 0.5	68.4 ± 0.2	66.5 ± 0.4	49.9 ± 0.3	$\mathbf{83.3 \pm 0.1}$	$\mathbf{77.7 \pm 0.1}$

most cases, the final performance is within $\sim 10\%$ of the supervised setting. We believe our improvements are significant since the baselines are close to the supervised model in terms of the performance and yet our progressive training strategy makes a significant improvement. We report the statistical significance test result between our method and other baselines in table 6. In all the cases, we observe the p-value to be less than 0.001 . The progressively trained model for Spanish-English does better than its corresponding supervised setting, outperforming it by $\sim 6\%$. We hypothesize, this is because of having a large number of instances in the source dataset S, the progressively trained model has access to more information and successfully leveraged it to improve the performance on target code-switched dataset. The comparison between our method and its ablated version ***- Source*** demonstrates the importance of source dataset while training the classifier. We can note that our proposed method is efficiently transferring the relevant information from the source dataset to the code-switched dataset, thereby improving the performance. On comparing our method with ***- Ratio***, we observe that using class ratio based instance selection improves the performance in two out of three cases. For the Tamil-English dataset, we observe that the weighted & micro F1 score are higher for ***- Ratio*** method but the macro F1 score is poor. This is because the F1 score of the positive class increases by $\sim 2\%$ but F1 score of negative class drops by $\sim 9\%$ when using ***- Ratio*** method instead of ours. Since the datatset is skewed in the favor of the positive class, this lead to a higher weighted and micro F1 score.

In Figure 3, we plot the performance obtained by No-PT and our method on both buckets. Since our method aims at improving the performance of low-resource language dominated instances, we expect our model m_2 to perform better on bucket B_2 and we observe the same. As shown in Figure 3, in most of the cases, our method performs better than the baseline on bucket B_2. For bucket B_1, we observe a minor improvement in the case of Spanish-English, whereas it stays similar for other datasets. Detailed qualitative analysis is present in section A.4 in Appendix.

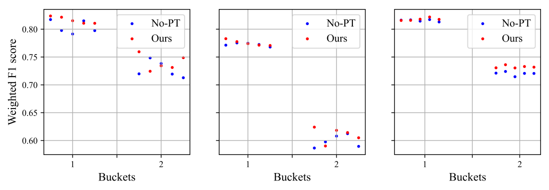

Figure 3: Model performance comparison across buckets (left-to-right: Spanish-English, Hindi-English, Tamil-English). Bucketwise F1 score for datasets. B_1 contains English dominated instances and B_2 contains low resource language dominated instances. Values are reported across 5 runs. Points on the same vertical line are from the same run i.e. both No-PT and our model were initialized with same initial weights.

5.6 Probing task : Out-Of-Distribution (OOD) detection

As previously mentioned, our proposed framework is based on two main hypotheses: (1.) A transformer model trained on resource-rich language dataset is more likely to be correct/robust on resource-rich language dominated samples compared to the low-resource language dominated samples, (2.) The models obtained using the progressive training framework is more likely to be correct/robust on the low-resource dominated samples compared to the models self-trained on the entire code-switched corpus at once. To confirm our hypotheses, we perform a probing task where we compute the fraction of the samples that are OOD.

More specifically, we ask two questions: a) Is the fraction of OOD samples same for both the buckets for model m_{pt}? b) Is there a change in OOD fraction for bucket B_2 if we use model m_1 instead of model m_{pt}? The first question helps in verifying the first part of the hypothesis and the second question helps in verifying the second part of the hypothesis.

Since the source dataset S is in English and the target dataset T is code-switched, the entire dataset T might be considered as out-of-distribution. However, transformer models are considered robust and can generalise well to OOD data (Hendrycks et al., 2020). Determining if a sample is OOD is difficult until we know more about the difference in the datasets. However, model probability can be used as a proxy. We use the method based on model's softmax probability output similar to Hendrycks and Gimpel (2018) to do OOD detection. Higher the probability of the predicted class, more is the confidence of the model, thus less likely the sample is out of distribution.

For a given model trained on a dataset, a threshold p_α is determined using the development set (or the unseen set of samples) to detect OOD samples. p_α is the probability value such that only α fraction of samples from the development set (or the unseen set of samples) have probability of the predicted class less than p_α. For example, if $\alpha = 10\%$, 90% of samples in the development set have probability of predicted class greater than p_α. If a new sample from another dataset (or bucket) has probability of predicted class less than p_α, we would consider it to be OOD. Using p_α, we can determine the fraction of samples from the new set that are OOD. Since, there is no method to know the exact value of α to be used, we report OOD using three values of α : 0.01, 0.05 and 0.10. For model m_{pt}, we use the development split from the dataset S to determine the value of p_α, and for model m_1, we use the set of samples from bucket B_1 that are not used in self-training (i.e. $B_1 - X_{st_1}$) to determine p_α. Based on the value of α, we conduct two experiments and answer our two questions.

Is the fraction of OOD samples same for both the buckets for model m_{pt}? In the first experiment, we consider the model trained on the source dataset and try to find the fraction of OOD samples in both the buckets. Since the first bucket contains more resource-rich language dominated samples, we expect a lesser fraction of samples to be out-of-distribution compared to the second bucket. We can observe this is true for almost all the datasets in the Figure 4. However, the difference in the out-of-distribution fraction for the buckets is different across different datasets. This shows that instances dominated by resource-rich language are less likely to be out-of-distribution for the classifier trained on S compared to instances dominated by low-resource language, thus providing empirical evidence in support of the first part of our hypothesis.

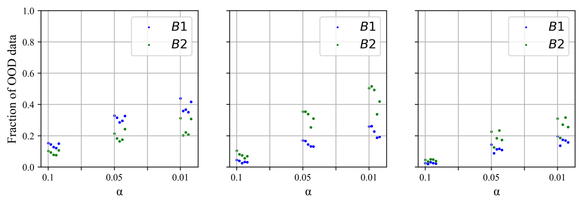

Figure 4: Figure showing bucketwise OOD for different datasets (left-to-right: Spanish-English, Hindi-English, Tamil-English) using the model m_{pt}. In most cases, samples in B_2 are more OOD compared to samples in B_1 across different values of α. Values are reported across 5 runs, points on the same vertical line are from the same run i.e. once a model m_{pt} has been trained, the same model is used to evaluate the fraction of OOD data in both the buckets.

Is there a change in OOD fraction for bucket B_2 if we use model m_1 instead of model m_{pt}? In the second experiment, we compare the fraction of OOD data in bucket B_2 for the models m_{pt} and m_1. In Figure 5, we observe a lesser fractions of samples in bucket B2 are OOD for model m_1 compared to model m_{pt}. This is expected since the model m_1 has seen samples with low-resource language words while training, thus providing em-

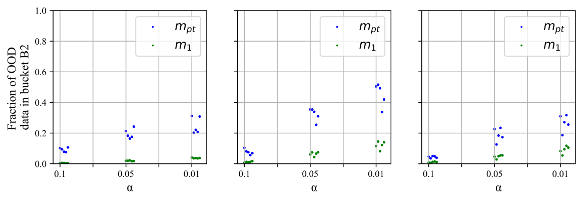

Figure 5: Figure showing model wise OOD for bucket B_2 across multiple datasets (left-to-right: Spanish-English, Hindi-English, Tamil-English). We compare two models, m_{pt} and m_1. In all the cases, we observe that samples in B_2 are more OOD for model m_{pt} compared to m_1 across different values of α. Values are reported for 5 runs and points on the same vertical line are from the same run i.e. both m_{pt} and m_1 were initialized with same initial weights.

Table 3: Performance using multiple multilingual models. First three rows denote performance without using progressive training and the last row denotes the performance when the model with best performance is used with progressive training.

Model	**Spanish-English**			**Hindi-English**			**Tamil-English**		
	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1
mBERT	76.2 ± 0.6	76.3 ± 0.6	76.3 ± 0.6	67.1 ± 0.8	69.2 ± 0.3	68.4 ± 0.6	53.3 ± 0.4	78.6 ± 0.6	76.7 ± 0.2
MuRIL	-	-	-	$\mathbf{77.0 \pm 0.4}$	$\mathbf{77.7 \pm 0.3}$	$\mathbf{77.7 \pm 0.4}$	54.2 ± 0.2	64.2 ± 0.4	68.8 ± 0.3
IndicBERT	-	-	-	73.5 ± 0.5	74.5 ± 0.3	74.3 ± 0.4	$\mathbf{54.6 \pm 0.1}$	68.0 ± 0.6	71.3 ± 0.4
Ours + Best	$\mathbf{77.4 \pm 0.8}$	$\mathbf{77.5 \pm 0.8}$	$\mathbf{77.5 \pm 0.8}$	$\mathbf{77.0 \pm 0.4}$	77.6 ± 0.4	77.6 ± 0.4	53.1 ± 0.4	$\mathbf{80.5 \pm 0.2}$	$\mathbf{77.5 \pm 0.1}$

Table 4: Model performance on the three datasets for different number of buckets (k) using mBERT.

Buckets	**Spanish-English**			**Hindi-English**			**Tamil-English**		
	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1
k=2	$\mathbf{77.4 \pm 0.8}$	$\mathbf{77.5 \pm 0.8}$	$\mathbf{77.5 \pm 0.8}$	$\mathbf{67.8 \pm 0.8}$	$\mathbf{69.9 \pm 0.5}$	$\mathbf{69.1 \pm 0.7}$	$\mathbf{53.1 \pm 0.4}$	80.5 ± 0.2	$\mathbf{77.5 \pm 0.1}$
k=3	76.1 ± 0.7	76.2 ± 0.7	76.2 ± 0.7	67.2 ± 0.6	69.4 ± 0.5	68.6 ± 0.6	53.1 ± 0.2	79.9 ± 0.4	77.2 ± 0.1
k=4	76.6 ± 1.6	76.7 ± 1.7	76.7 ± 1.6	67.6 ± 0.4	69.7 ± 0.3	68.9 ± 0.4	52.9 ± 0.4	$\mathbf{80.6 \pm 0.7}$	$\mathbf{77.5 \pm 0.3}$

pirical evidence in the support of our proposed training strategy. Although, the samples from B_2 would still have noisy labels, we expect them to be more accurate when predicted by m_1 than m_{pt}

5.7 Comparison with other multilingual models

Recently, multiple multilingual transformer models focusing on Indian languages have been proposed. We experiment with MuRIL (Khanuja et al., 2021) and IndicBERT (Kakwani et al., 2020). Firstly, we obtain the performance of three language models: mBERT, MuRIL, and IndicBERT without progressive training on all datasets and we use progressive training on top of the best performing model corresponding to each dataset and verify whether it further improves the performance. The F1 scores are reported in Table 3. We observe that performance either increases or stays very competitive in all the cases, thus showing our method is capable of improving performance even when used with the best multilingual model for the task.

5.8 Hyper-parameter sensitivity analysis

There are two hyper-parameters in our experiments: the number of buckets (k) and the ratio of samples selected for self-training (δ). We vary k from 2 to 4 to study the effect of the number of buckets on the performance and the F1-scores are reported in Table 4. We observe that the values with k=2 perform either better or competitive with other values. As mentioned earlier, we believe this is because of the number of spikes in the distribution plot of f_{eng} being 1 or 2 across the datasets. In presence of more number of spikes, higher value of k should give better performance.

For studying the effect of hyper-parameter δ, we plot macro, micro, and weighted F1 scores across multiple values of δ in figure 6. With low δ, there wouldn't be enough sentences for self-training to help whereas with high δ, the samples would be too noisy. Thus, a value in the middle i.e. 0.4-0.6 should be reasonable choice.

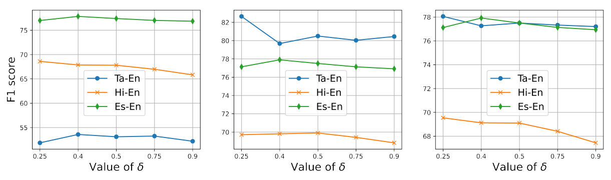

Figure 6: F1 score using different values of parameter δ for the three datasets (Spanish-English, Hindi-English and Tamil-English). The plots represent macro, micro and weighted F1 score (left-to-right).

6 Conclusion, limitations and future work

In this paper, we propose progressive training framework that takes distinction between low-resource and resource-rich language into account while doing zero-shot transfer learning for code-switched texts. We show that our framework improves performance across multiple datasets. Further, we also create probing tasks to provide empirical evidence in support of our hypothesis.

A key potential limitation of the current framework is that depending on the size of S and the capacity of the model, the model might forget information relevant for the low-resource language. In future, we would like to perform a systematic study of the dependency on size of S. And also, we want to extend the framework to other tasks like question-answering and natural language inference.

7 Ethical consideration

This paper proposes a progressive training framework to transfer knowledge from resource-rich language data to low-resource code-switched data. We work on sentiment clsasification task which is a standard NLP problem. Based on our experiments, we don't see any major ethical concerns with our work.

References

Gustavo Aguilar and Thamar Solorio. 2020. From English to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8033–8044, Online. Association for Computational Linguistics.

Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P McCrae. Dravidiancodemix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text. *Language Resources and Evaluation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Akshat Gupta, Sargam Menghani, Sai Krishna Rallabandi, and Alan W Black. 2021. Unsupervised self-training for sentiment analysis of code-switched data. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 103–112, Online. Association for Computational Linguistics.

Benjamin Heinzerling and Michael Strube. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 273–291, Florence, Italy. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2018. A baseline for detecting misclassified and out-of-distribution examples in neural networks.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online. Association for Computational Linguistics.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, Shruti Gupta, Subhash Chandra Bose Gali, Vish Subramanian, and Partha P. Talukdar. 2021. Muril: Multilingual representations for indian languages. *CoRR*, abs/2103.10730.

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.

Yash Kumar Lal, Vaibhav Kumar, Mrinal Dhar, Manish Shrivastava, and Philipp Koehn. 2019. De-mixing sentiment from code-mixed text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 371–377, Florence, Italy. Association for Computational Linguistics.

Dheeraj Mekala and Jingbo Shang. 2020. Contextualized weak supervision for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 323–333, Online. Association for Computational Linguistics.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020. Text classification using label names only: A language model self-training approach.

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

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert?

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*, abs/2006.06402.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada. Association for Computational Linguistics.

Bidisha Samanta, Niloy Ganguly, and Soumen Chakrabarti. 2019. Improved sentiment detection via label transfer from monolingual to synthetic code-switched text. *arXiv preprint arXiv:1906.05725*.

David Vilares, Miguel A. Alonso, and Carlos Gómez-Rodríguez. 2015. Sentiment analysis on monolingual, multilingual and code-switching Twitter corpora. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–8, Lisboa, Portugal. Association for Computational Linguistics.

David Vilares, Miguel A. Alonso, and Carlos Gómez-Rodríguez. 2016. EN-ES-CS: An English-Spanish code-switching Twitter corpus for multilingual sentiment analysis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4149–4153, Portorož, Slovenia. European Language Resources Association (ELRA).

Zihan Wang, Dheeraj Mekala, and Jingbo Shang. 2021. X-class: Text classification with extremely weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3043–3053, Online. Association for Computational Linguistics.

Genta Indra Winata, Zhaojiang Lin, Jamin Shin, Zihan Liu, and Pascale Fung. 2019. Hierarchical meta-embeddings for code-switching named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3541–3547, Hong Kong, China. Association for Computational Linguistics.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis.

A Appendix

A.1 Statistics related to the dataset

Table 5: Average value and standard deviation of f_{eng} for both the buckets.

Dataset	B1	B2
Spanish-English	0.79 ± 0.08	0.44 ± 0.14
Hindi-English	0.79 ± 0.21	0.14 ± 0.13
Tamil-English	0.51 ± 0.16	0.13 ± 0.09

A.2 Statistical Significance Results

Table 6: We perform paired t-test between our method and baselines. The p-value obtained by performing the test between our methods and baselines for all three datasets is reported in the table.

Dataset	No-PT	DCE	*- Source*	*- Ratio*
Spanish-English	$5.11e^{-13}$	$2.09e^{-6}$	$8.28e^{-4}$	$8.82e^{-8}$
Hindi-English	$2.76e^{-6}$	$4.37e^{-43}$	$3.49e^{-4}$	$6.85e^{-12}$
Tamil-English	$1.87e^{-41}$	$3.72e^{-16}$	$3.71e^{-3}$	$3.40e^{-7}$

A.3 Distribution plot of f_{eng} words

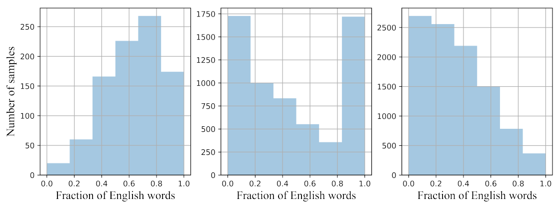

Figure 7: Distribution of $f_{eng}(x)$ vs number of samples for the Spanish-English, Hindi-English and Tamil-English datasets (left-to-right). For Hindi-English, we can observe two spikes in the graph showing some samples are heavily dominated by English and some samples are heavily dominated by Hindi. For the other two datasets, we observe the progression to be more gradual.

A.4 Qualitative analysis

As discussed previously, on the low-resource language dominated bucket, our model is correct more often than the ***No-PT*** baseline. We focus on samples from bucket B2 for qualitative analysis. For the sample, *"fixing me saja hone ka gift"*, the Hindi word *"saja"* refers to punishment which is negative in sentiment whereas the word *"gift"* is positive in sentiment. Thus, the contextual information in the Hindi combined with that of the English is necessary to make correct prediction. For the sample *"Mera bharat mahan, padhega India tabhi badhega India"*, the model has to identify Hindi words "mahan" & "badhega" to make the correct predictions.