Title: Dynamic data augmentation for sentiment analysis on code-mixed text

G068 (s2441181, s2424283, s2449595)

Abstract

There is growing interest in developing sentiment analysis models that can handle code-switched (CS) text to improve their applicability in multilingual settings. In this study, we explore the use of representations derived from pre-trained language models (PLMs) for dynamically augmenting CS datasets to achieve better performance on sentiment analysis (SA). Mixup is a popular data augmentation strategy that has found success across various downstream tasks in NLP, but it has not been evaluated on code-mixed text yet. Our results show that the proposed approach using latent mixup on the XLM-T pretrained language model improves the accuracy and weighted F1-score by 1%. Additionally, the embeddings generated by the final model are more interpretable and represent the relationships between the labels better.

1. Introduction

Code-mixing or code-switching (CS) refers to the tendency of multilingual or bilingual speakers to phase in and out of different languages when trying to communicate (Poplack, 1978). This interweaving of languages can occur over the length of the conversation or even in a single utterance. Up until recently, CS language was only considered to be restricted to informal or casual speech; however, recent findings show that it is becoming a prevalent form of communication in multilingual societies like India (Parshad et al., 2016). In fact, code-switching is expected to present itself in any conversation involving multilingual speakers (201, 2010).

(Solorio et al., 2021) highlights the challenges in building language models for code-mixed text. Code-mixing can involve using words and grammar from multiple languages within a single sentence. This can make it difficult for language models to correctly predict the correct language and associated grammar rules and vocabulary. Code-mixing can result in dissonance in the context between the different languages used in the text. This can cause confusion for the language model, particularly if it is not trained to comprehend the cultural and linguistic subtleties of the languages involved. Code-mixed text can be ambiguous, where the same words can have different meanings depending on the language used. This can make it challenging for the language model to disambiguate and make accurate predictions. To add to this, code-mixed text is relatively scarce as compared to monolingual text, and language models trained on monolingual data may not have enough exposure to code-mixed text to accurately model its linguistic features.

Most of the current NLP models are built for English texts due to the availability of large English annotated datasets. (Farzindar & Inkpen, 2015; Hodorog et al., 2022). Since English is so prevalent on social media, there is a high likelihood that code-mixed text will contain some English words or phrases, even if the other language(s) used is not English. This could make it easier for natural language processing (NLP) models to generalize across different code-mixed languages, as the models could potentially use their knowledge of English to better understand and process the code-mixed text (Choudhury, 2018). (Jose et al., 2020) also show that almost 84% of the code-mixing for downstream tasks like Sentiment Analysis contains English, so having a language model for English-X (any other language) is a viable solution for understanding social media code-mixed text.

Sentiment Analysis (SA) is concerned with extracting emotions or feelings encoded in textual data (Kumawat et al., 2021). SA has found various applications in real-world scenarios; businesses can use SA to understand the likes and dislikes of customers and accordingly make recommendations to them, political campaigners can use social media posts to understand the policy leanings of individuals, etc (Kour & Josan, 2021). Today, social media allows users to express themselves in a more free and unstructured manner. As a result of this, code-switched discourse has become commonplace for multilingual users on platforms like Twitter (Zaharia et al., 2020a). This means that in order to fully incorporate the diverse opinions and preferences of multilingual speakers into any analysis, we must be able to mine sentiment data from code-switched text.

In its early days, SA was dominated by classical approaches that soon were superseded by deep learning approaches that leveraged complex architectures (Dang et al., 2020). Recently the world of NLP has entered a new era of Pretrained Language Models (PLMs) (Qiu et al., 2020). Powerful PLMs that have been pretrained on extremely large corpora of data have now established themselves as the SOTA on SA (Salih et al., 2022). However, applying PLMs to code-mixed data is not a straightforward task. This is because, as mentioned earlier, code-switched text is complex and low-resource (Kour & Josan, 2021; Gupta et al., 2020b). This makes it difficult to train LLMs on big enough corpora of CS text, e.g. Hinglish or Spanglish, to capture the nuances

of the CS lexicon and grammar.

Multilingual Language Models are trained on large corpora of monolingual samples from a number of different languages. This enables them to create a common embedding space for the different languages. These multilingual PLMs have found some success in SA. The recent SemEval-2020 Competition Task 9 showcased that BERT-based ensemble techniques were the most popular and successful option for SA on CS data (Patwa et al., 2020). However, the performance of PLMs on CS data is still far from acceptable. (Li & Murray, 2022; Krishnan et al., 2021).

One popular solution to improve performance in PLMs is Data Augmentation (Feng et al., 2021). DA aims to tackle the data scarcity issue inherent in low-resource settings like CS text by creating synthetic samples to expand the training corpora and add more diversity to the available data. This larger context helps PLMs learn the specifics of CS text better.

In our study, we aim to explore the usefulness of representations derived from PLMs for augmenting our CS dataset in order to achieve better performance on SA. We use an augmentation technique known as Mix-Up (Zhang et al., 2018) that, as the name suggests, mixes naturally occurring samples in our dataset to create synthetic CS samples. However, instead of performing this mix-up statically, we perform it dynamically by crossing over latent representations of the sentences created by a PLM during pretraining. This idea was inspired by (Chen et al., 2020) and is a variation of the original Mix-Up that was used in the context of image classification; however, to the best of our knowledge, the same has not been used on CS text before. Hence, we aim to answer the following research questions[1] :-

1. **Does dynamic data augmentation using latent representations from PLMs improve sentiment analysis performance on Code-mixed text?**

The rest of the report is structured as follows:- Section 2 talks about the datasets, the preprocessing pipeline, and the metrics used; Section 3 gives a comprehensive overview of the adopted methodology ("mix-up"); Section 4 describes the experiments that we conducted; Section 5 presents the results and their analysis; Section 6 introduces relevant research that has been done in the area; and finally, Section 7 holds the concluding remarks including directions for future work.

[1]Research questions have evolved from those presented in the interim report and have become more targeted. This is because of a couple of reasons - (1) we felt that the earlier scope was too broad and a more specific objective would allow us to provide better analysis, (2) the static data augmentation strategy provided poor results during initial experimentation, and (3) accounting for resource constraints, specifically time and available computational power.

2. Data set and task

2.1. Datasets Description

Hinglish code-switched tweets are obtained from the SentiMix dataset provided in the SemEval-2020 Task 9 on Sentiment Analysis of Code-Mixed Tweets (Patwa et al., 2020). The entire dataset consists of 20K samples of Hinglish CS tweets with each tweet classified as having “Positive”, “Negative” or “Neutral” sentiment. We do not use the original train, validation, and test splits provided but take the whole dataset and randomly sample to achieve the following split - 80% training, 10% validation, and 10% test respectively. The raw dataset is provided in the CoNLL format, which already breaks the tweets into tokens with language labels for each token. We convert this to a sentence/label construction in order to have homogeneity across our datasets. Dataset statistics for the same are presented in Table 1 below.

Monolingual English tweets are obtained from the Stanford Sentiment 140 dataset which consists of 16 million samples (Go et al., 2009). For the purpose of our exercise, we randomly sample 10,000 tweets which we believe is sufficient to train and understand our models. This dataset is different from the Hinglish dataset in that it only contains "Positive" and "Negative" tweets. We do not believe this is a significant limitation because the English dataset is only used to supplement learning. The dataset statistics are presented in Table 1 below. One tweet was found to be corrupt after preprocessing and was removed from the analysis.

Finally, we also use Hindi monolingual tweets to supplement learning. Romanized Hindi tweets from the Sentiment Analysis in Indian Languages (SAIL) 2017 dataset were used for this [2]. The dataset consisted of 10,079 tweets with all three classes. Dataset statistics are summarised in Table 1 below.

Dataset	Splits		Classes	
Hinglish	Training	12,104	Positive	5,034
	Validation	1,513	Negative	4,459
	Test	1,514	Neutral	5,638
English	Training	9,999	Positive	4,955
			Negative	5,044
			Neutral	-
Hindi	Training	10,077	Positive	3,201
			Negative	2,319
			Neutral	4,557

Table 1. Datasets Summary

2.2. Preprocessing

2.2.1. Cleaning

Text data obtained from scraping Twitter is filled with various forms of noise. We processed the text using the following steps in order to remove some components of the noise without losing important information encoded in the tweets.

[2]http://amitavadas.com/SAIL/data.html

- Punctuation is retained as it can be valuable in conveying the tone.
- Hash symbols are removed but the content of the hashtags is retained as it may contain important information to decode the overall sentiment of the tweet.
- URLs are removed.
- Emojis and emoticons are essential to convey sentiment but in their Unicode formats are noisy for the models to interpret. Hence, they are retained but converted to their text descriptions using the emoji library.

2.2.2. Tokenization

Tokenization for XLM-T models was performed using the tokenizer provided as a part of the XLM-T model on Hugging Face[3]. This ensures we achieve the sort of tokens the model performs best with which in the case of XLM-RoBERTa-based models is a subword-level Byte Pair Encoded vocabulary. Byte Pair Encoding was initially introduced as a compression algorithm (Gage, 1994) and adapted to NLP for word segmentation by Sennrich et al. (Sennrich et al., 2016). It helps represent an open word vocabulary with a *compact fixed-size vocabulary* of subwords, eliminating the need for large vocabularies and back-off models. Moreover, using subword instead of character-level representation allows for shorter sequences (Sennrich et al., 2016).

2.3. Evaluations Metrics

2.3.1. Accuracy

Accuracy is the ratio of the number of correct predictions to the number of total predictions.

$$accuracy = \frac{correctpredictions}{totalpredictions}$$

2.3.2. Weighted F1 score

$$F1score = \frac{2 * precision * recall}{precision + recall}$$

here,

$$precision = \frac{tp}{tp + fp}$$

here,

$$recall = \frac{tp}{tp + fn}$$

where, tp = true positives, fp = false positives, fn = false negatives.
The weighted F1 score is computed by finding the average of all the F1 scores for each class, taking into account the support for each class.

[3]https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment

2.3.3. Expected Calibration Error

ECE is used to measure the model's confidence calibration. Often, the predicted probabilities that the model outputs do not reflect the true probability that a particular instance belongs to a particular class. For example, if an instance is assigned the label A by a model with an accuracy of 0.8. It does not necessarily mean that the instance has an 80% chance of belonging to class A. This discrepancy is the model calibration error. The lower the ECE value, the better calibrated a model is.

3. Methodology

The core idea is to use latent representations generated by LLMs to augment the existing training set in order to improve sentiment classification in CS text. We try to evaluate if the latent representations produced by LLMs contain some important linguistic information that could be useful in understanding CS text. These representations are generated by feeding the input data into a pre-trained language model and extracting the embeddings produced by the encoder in the model.

To perform data augmentation, we use a technique called Mixup (Zhang et al., 2018). Mixup is a data augmentation technique that generates new data samples by linearly interpolating between pairs of existing data samples. Vanilla Mixup is a static data augmentation technique that can be applied before training begins. However, it has a limitation: it can only be applied to data points of the same dimensionality. This is problematic for text data, where the input sequences can have varying lengths. To overcome this limitation, (Chen et al., 2020) proposed a mixup variant called latent mixup. In latent Mixup, the latent embeddings are mixed instead of the raw features. This method is dynamic since it requires a model to extract latent embeddings for interpolation. By using latent Mixup, we can create new data samples with varying lengths, which can improve the performance of the model. The samples are mixed according to Equation 1, where $\lambda \sim Beta(\alpha, \alpha)$ and $\alpha > 0$.

In our project, Mixup is used to create synthetic data samples by interpolating between the latent representations of two randomly selected data samples from the training dataset. It is a data augmentation technique where synthetic data points are created $(\tilde{x}, \tilde{y})$ by interpolating the latent representations given out of two randomly sampled data points (x_i, y_i) and (x_j, y_j) from the training dataset D. The mixing ratio is determined by a parameter called λ, which is sampled from a Beta distribution with a parameter α greater than zero. The synthetic data sample is created by taking a weighted average of the two original data samples' latent representations.

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda) x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda) y_j \end{aligned} \tag{1}$$

Figure 1 shows an overview of the architecture used to generate the augmented samples.

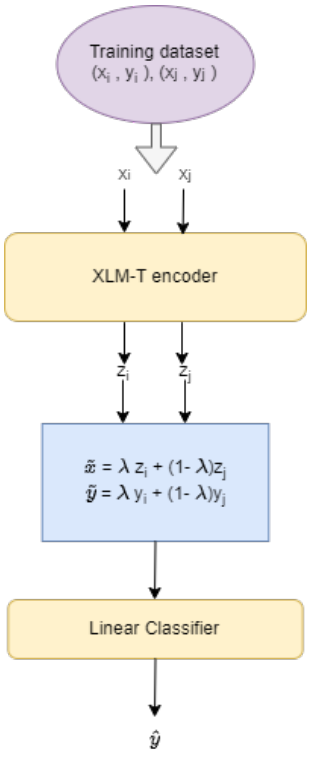

Figure 1. Dynamic data augmentation with latent mixup using XLM-T transformer encoder.

This latent mixup method is low in computational cost and also requires no additional space for storing the newly generated data as samples are created on the fly.

Since code-mixed text can have varying language patterns, mixing the latent representations of different code-mixed text samples could potentially lead to the creation of more diverse training samples, which can help the model generalize better to unseen data during inference.

4. Experiments and Results

To establish a baseline, we fine-tuned and tested some well-known SOTA pretrained models such as BERT-base (Devlin et al., 2018), mBERT, XLMR (Conneau et al., 2019), XLM-T(Barbieri et al., 2021) and IndicBERT (Kakwani et al., 2020) on the SemEval-2020 Task 9 Sentiment Analysis Dataset. All input sequences were padded or trimmed to a length of 128 and appropriate attention masks were created to differentiate the real tokens from the padded tokens. Start and end tokens were also added to each input sequence. For finetuning the transformer encoders, we unfroze all the weights in the encoder and then fed the output transformer embeddings to a linear classifier that outputs a vector of scores for each class (positive, negative, neutral). The class with the maximum score was chosen as the predicted class. For all our experiments, we used a learning rate of 1e-5, a batch size of 20, and trained all our experiments for 30 epochs with early stopping enabled and its patience parameter set to 10. We also observed a much smoother decrease in loss during training while using a cosine annealing learning rate scheduler. Table 2 consolidates the performance of our baseline models.

	Accuracy	Weighted F1	ECE
BERT	0.6	0.6	0.239
mBERT	0.61	0.61	0.301
XLMR	0.63	0.62	0.218
Indic-BERT	0.57	0.56	0.228
XLM-T	0.64	0.63	0.229

Table 2. Accuracy, Weighted F1 and ECE scores of baseline models.

All the multilingual pretrained models performed better than BERT which is a monolingual model. This hints that these models are able to recognise the presence of two different languages (English and Hindi) present in the code-mixed text better even though they had not been pretrained on romanized Hindi text. Among the multilingual pretrained models, mBERT performed the worst. This is because mBERT is pre-trained using only the Masked Language Modelling objective, which primarily captures monolingual information. This may give XLM-R and XLM-T an advantage for tasks that require better cross-lingual understanding, such as code-mixed text classification. In fact, (Qin et al., 2020) also demonstrated that limited overlap was observed between sentences of different languages in the shared embedding space of mBERT. XLM-T performed the best out of all the models. This was expected as it was pre-trained on large amounts of Twitter data. This makes XLM-T better suited for understanding noisy and informal texts, such as social media posts or chat messages. Surprisingly Indic-BERT performed worse than all of them, even though it had been pretrained on Indian languages. This could be because the Hinglish sentences in the SemEval dataset contain more English words and linguistic patterns than Hindi, a common trend observed in social media texts. Indic-BERT was also trained on fewer parameters than BERT which could have contributed to its poorer performance.

In the next experiment, we applied latent mixup to the best-performing model chosen from our baselines [4]. To apply latent mixup we first created a subset of 6k randomly chosen samples from the Hinglish dataset. For each example in this subset, we randomly sampled another from the subset and applied latent mixup using the default metrics defined in the original paper (Zhang et al., 2018). We first experimented with picking samples from the same class and then experimented with picking samples randomly across all classes. The motivation for this was to see which experiment was better at generating mixed samples. Table 3 shows the results for both these experiments.

We expected the 'latent-mixup-within-class' experiment to fare better as mixing intermediate representations from two different classes could result in noisy representations that

[4]Due to the training process being computationally expensive we only report results on the best-performing baseline.

could be harder to decipher for the model. However, we observed the opposite in our results. 'latent-mixup-across-class' performed better than 'latent-mixup-within-class' by 1% in both accuracy and weighted F1. We suspect this is because choosing samples from the same class may not introduce as much diversity. This could be because the data points being mixed are likely to be more similar to each other. This can cause the model to overfit to the training data and not generalize well to new data, which can also lead to a higher ECE metric. We do note that there is an increase in the ECE metric in both experiments over the baseline, which means that the additional samples actually resulted in the poorer calibration of the model. In other words, the model's confidence in its predictions is decreasing, and it may be overestimating or underestimating the probability of a particular class.

	Accuracy	**Weighted F1**	**ECE**
XLM-T-m (within class)	0.63	0.63	0.351
XLM-T-m (across class)	0.64	0.64	0.324

Table 3. Results with XLM-T latent mixup (XLM-T-m) within and across class

In order to help our pretrained models understand the CS dataset better, we added some monolingual datasets. We added the English and Hindi monolingual datasets which were described in section 2.1. Table 4 shows the results of latent mixup across classes on mBERT, XLM-R, XLM-T with added monolingual data during finetuning.

	Accuracy	**Weighted F1**	**ECE**
mBERT-m-across + mono data	0.61	0.61	0.338
XLM-R-m-across + mono data	0.63	0.63	0.282
XLM-T-m-across + mono data	0.64	0.64	0.299

Table 4. Results with latent mixup across class with additional monolingual data

The addition of the monolingual datasets did help decrease the ECE score of XLM-T-m-across, even though there was barely any change in the Weighted F1. So the additional monolingual data did help in improving the confidence of the model's predictions even though it did not improve the predictions themselves. Although the ECE score is still slightly higher than the baseline XLM-T model. We also observed no improvements in the accuracy or weighted F1 scores for any other models. Perhaps the monolingual data distribution is quite different from our Hinglish dataset distribution.

5. Discussion

Comparing the performance of XLM-T baseline vs XLM-T with latent mixup across classes + fine-tuned on added monolingual data (final model):

label	baseline	final	text
neg	neu	neg	Official Lota kuch b kr ly uska muqadar bath-room he ha
neg	neu	neg	I'm TRYING chutti nahi approve ho rahi (

Table 5. Example sentences being labelled correctly by the final model

label	baseline	final	text
neu	neu	neg	Apna koi future nhi bata payega :back-hand_index_pointing_right:: wink-ing_face_with_tongue:
neu	neu	pos.	2nd semester ka result 3rd ka Ab ye 4th ka Tests ki duaa karwai wo be EIDI be

Table 6. Example sentences being labeled incorrectly by the final model

The baseline model was incorrectly predicting a lot of positive and negative samples as neutral, but the final model is correctly classifying these samples. Examples are presented in Table 5. This improvement indicates that the final model has a better understanding of the nuances of code-mixed writing and is better able to differentiate between different sentiment labels.

Due to a better understanding of the subtleties of the code-mixed text, the final model is better equipped to accurately classify samples that have a slightly positive or slightly negative sentiment. This suggests that the final model has a better ability to identify and interpret more complex language features.

The labels are one-hot encoded. For example, a positive label will be of the form $[1, 0, 0]$. During augmentation, we interpolate the labels of two samples to create the new label. When two samples across classes are mixed, a new label will be created that is somewhere between the two, resulting in samples that are closer to the neutral labels. This means that all such samples that are being created through data augmentation have labels that are closer to neutral than the original samples.

When the model is trained on these augmented samples, it is better able to distinguish between slightly negative or slightly positive samples and neutral samples. This is because the model has been exposed to a wider range of samples, including those that fall closer to the neutral boundary. As a result, the model is able to learn to identify and inter-

pret the more subtle features of the code-mixed text that distinguish different sentiment labels.

By improving the model's ability to correctly classify these nuanced samples, the overall performance of the model has been improved. This is particularly important in the sentiment analysis task where the boundary between positive and negative sentiment can be blurry, and where slight variations in language can significantly impact the sentiment classification.

However, despite these improvements, there is a trade-off. The final model is more prone to incorrectly predicting neutral samples as either positive or negative, whereas the baseline model was correctly predicting these as neutral. We can see examples of this in Table 6. This suggests that the final model may be overly sensitive to certain language features, resulting in incorrect predictions in some cases. While the final model has improved in some respects, this trade-off highlights the importance of understanding the limitations of the model's performance and the need for continued improvement and refinement.

	text
Hindi with English words	respected sir ji hamare is purshani ka tod nikaliye sir ji please Azamgarh se jalandhar city punjab t â€¦
Hindi and English both together	I am very sad ki mainpuri up me krishna nagar me fauji ko thapd mara gaya jo
English with Hindi words	I will do you one better - namak parantha with :mango:

Table 7. Example sentences showing variation in language-switching patterns

	text
Hindi structure	Apke aeria me jo network best ho whi :grinning_face_with_smiling_eyes:
both Hindi and English structure	They are really behaving like children Mujhe batting nahi mili main kal se khelne nahi aaoonga
English structure	another chotu babie a whole cutie ILYSM !:sparkling_heart: Your mother is literally so beautiful too omg !! Im always here if you need me

Table 8. Example sentences showing variation in grammar structure

Pretrained language models may be struggling with code-mixed data due to the following reasons -

- **Variation in language-switching patterns:** As we can see in Table 7 in code-mixed data, the writer can switch between different languages within a sentence or even within a single word. This can result in a lack of consistency in language patterns, which can make it difficult for the model to capture the correct context and meaning of the text.

Figure 2. T-SNE representation of embeddings of 100 random test samples generated using the final model

Figure 3. T-SNE representation of embeddings of 100 random test samples generated using the baseline XLM-T model

- **variation in grammatical structure:** Hinglish text can follow the grammatical structure of either Hindi or English or both in the same sentence as we can see in Table 8. This variation in grammatical structure can make it difficult for pre-trained models to accurately capture the correct meaning and context of the text.
- **lack of diversity in the training data:** Because the amount of code-mixed data available for training is scarce and code-mixed text has a lot of scope for variation generally, the model's ability to generalize to unseen code-mixed data is limited.

We visualize the embeddings generated by the baseline and the final model for 100 randomly selected test samples with Fig. 2 and Fig. 3. Comparing the two we can see that the final model's embeddings represent the semantics of labels better, i.e. the neutral samples lie between the positive and negative samples. Overall, the results indicate that the proposed data augmentation method and fine-tuning strategy led to a sentiment analysis model that captures the nuances of code-mixed text better.

6. Related work

As described in the introduction, PLMs have been used in NLP research on SA in the context of code-mixed text. We mention a few

important contributions here. (Sultan et al., 2020) propose a fine-tuning approach for sentiment analysis of code-mixed data using XLM-RoBERTa, a pre-trained transformer model. This approach involves feeding the model with both word-level and character-level information, which is useful for handling languages with different scripts or characters. (Braaksma et al., 2020) define a two-step fine-tuning strategy for sentiment analysis of code-mixed data. In the first step, they fine-tune XLM-RoBERTa for the language modeling objective, which involves predicting the next word in a sequence. In the second step, they fine-tune the model for the task of sentiment analysis. This approach improves the performance of XLM-RoBERTa on code-mixed data.

(Ver & Soares, 2020) introduce an ensemble of three models - logistic regression, random forest, and BERT - for sentiment analysis of code-mixed data. They also propose a technique that involves representing the input text as a combination of one-hot encoded vectors and multilingual embeddings. (Wang, 2020) uses a pre-training approach called "masked language modeling," which involves randomly masking some words in the input text and asking the model to predict the missing words. (Zaharia et al., 2020b) use a similar approach but also include an additional pre-training task called "translation language modeling," which involves translating Hinglish text to either Hindi or English and asking the model to predict the next word in the translated text. Both papers show that their pre-training and fine-tuning approach improves the performance of BERT on sentiment analysis of Hinglish text.

NLP has seen a surge in research focused on data augmentation techniques because of the growing interest in low-resource domains, novel downstream tasks and the immense data requirements of LMs (Feng et al., 2021). We summarise some data augmentation techniques, especially in the context of synthetic code-mixed (SCM) text generation.

Traditional rule-based approaches involved the use of predefined transformations without adding any model based complexities. (Pratapa et al., 2018) proposed an approach that aligns the parse trees of English and Hindi sentences using an equivalence constraint. This alignment is used to replace words in the source sentence with their equivalent code-mixed counterparts. (Wei & Zou, 2019) use simple token-level random insertion, swap, and deletion operations to generate code-mixed text. (Gupta et al., 2020a) employ an approach called the Matrix-embedded Language Framework (MLF) that generates code-mixed text without relying on parallel data.

More sophisticated approaches use some form of learning to identify properties or relationships in the data. (Xie et al., 2020) propose the Unsupervised Data Augmentation (UDA) that paraphrases unlabeled data using back translation and word replacement and then uses the original input and the augmented version for consistency training. (Li & Murray, 2019) suggest two algorithms for creating synthetic code-mixed (SCM) samples: Lexical replacement which randomly replaces words in the source language and Syntactic replacement which accounts for syntactic information in the replacement strategy using a POS tagger to first tag the words in the source sentence. (Lui & Baldwin, 2019) use a self attention mechanism to identify appropriate words in the source language to replace in order to generate CS text.

7. Conclusion

In this study, we implemented and evaluated a dynamic data augmentation method for sentiment analysis of code-mixed Twitter data. XLM-T (finetuned on Hindi, English, and Hinglish) with dynamic data augmentation using Hinglish performed the best. It improved accuracy and weighted F1-score by 1%. Moreover, the visualizations of the embeddings produced by the baseline and the final model revealed that the final model generated embeddings that better represent semantics, with neutral samples located between positive and negative samples.

One possible future direction is to explore sample selection strategies based on metrics such as the model's confidence when predicting the labels for the samples, instead of random sampling for mixup. Additionally, we suggest exploring the use of non-linear combinations of samples for generating new samples, as this may further enhance the performance of the model. Overall, our study highlights the potential of dynamic data augmentation in improving the accuracy and interpretability of sentiment analysis models for code-mixed text and opens up avenues for further research in this area.

References

The Handbook of Language Contact. John Wiley & Sons, Ltd, 2010. ISBN 9781444318159. doi: https://doi.org/10.1002/9781444318159.fmatter. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781444318159.fmatter.

Barbieri, Francesco, Espinosa Anke, Luis, and Camacho-Collados, Jose. Xlm-t: A multilingual language model toolkit for twitter. *arXiv e-prints*, pp. arXiv–2104, 2021.

Braaksma, Aaron, Bansal, Namit, and de Rijke, Maarten. Multilingual code-switching sentiment analysis using pre-trained contextual embeddings. *arXiv preprint arXiv:2010.02511*, 2020.

Chen, Jiaao, Yang, Zichao, and Yang, Diyi. MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. 4 2020. URL http://arxiv.org/abs/2004.12239.

Choudhury, Monojit. Code-mixing and social media: A digital sociolinguistic agenda for global english. *English Today*, 34 (1):3–8, 2018.

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

Dang, Nhan Cach, Moreno-García, María N., and De la Prieta, Fernando. Sentiment analysis based on deep learning: A comparative study. *Electronics (Switzerland)*, 9(3), 3 2020. ISSN 20799292. doi: 10.3390/electronics9030483.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Farzindar, A and Inkpen, D. A survey of computational approaches to code-switching. *Journal of Artificial Intelligence Research*, 53:613–656, 2015.

Feng, Steven Y., Gangal, Varun, Wei, Jason, Chandar, Sarath, Vosoughi, Soroush, Mitamura, Teruko, and Hovy, Eduard. A survey of data augmentation approaches for nlp, 2021.

Gage, Philip. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38, 1994.

Go, Alec, Bhayani, Richa, and Huang, Lei. Twitter sentiment classification using distant supervision. *Processing*, pp. 1–6, 2009. URL http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf.

Gupta, Aditya, Winata, Genta Indra, Madan, Vishal, and Chakraborty, Tanmoy. Matrix-embedded language framework for code-switching text generation. *arXiv preprint arXiv:2010.02546*, 2020a.

Gupta, Deepak, Ekbal, Asif, and Bhattacharyya, Pushpak. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2267–2280, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.206. URL https://aclanthology.org/2020.findings-emnlp.206.

Hodorog, Andreea, Trandabat, Diana, Sava, MI, and Maruster, Lucian. Predicting code-mixing in romanian-english social media conversations. *Computer Speech & Language*, 73:101312, 2022.

Jose, Roshan, Patra, Soujanya, Choudhury, Monojit, and Mukherjee, Animesh. A survey on code-mixing datasets: Insights and recommendations. *arXiv preprint arXiv:2012.11904*, 2020.

Kakwani, Divyanshu, Kunchukuttan, Anoop, Golla, Satish, N.C., Gokul, Bhattacharyya, Avik, Khapra, Mitesh M., and Kumar, Pratyush. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*, 2020.

Kour, Ramandeep and Josan, Gurpreet Singh. Sentiment Analysis of Codemixed Text: A Survey. *IJCST*, 12(2), 6 2021. ISSN 0976-8491. URL https://www.ijcst.com/vol12/issue2/3-ramandeep-kour.pdf.

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

Kumawat, Spraha, Yadav, Inna, Pahal, Nisha, and Goel, Deepti. Sentiment analysis using language models: A study. In *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*, pp. 984–988. Institute of Electrical and Electronics Engineers Inc., 1 2021. ISBN 9780738131603. doi: 10.1109/Confluence51648.2021.9377043.

Li, Shuyue Stella and Murray, Kenton. Language agnostic code-mixing data augmentation by predicting linguistic patterns, 2022.

Li, Xiaofei and Murray, Kenton. Synthetic code-mixed data generation using neural language models. *arXiv preprint arXiv:1909.00222*, 2019.

Lui, Marco and Baldwin, Timothy. Pre-training via paraphrasing. 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)*, pp. 1261–1271, 2019.

Parshad, Rana D, Bhowmick, Suman, Chand, Vineeta, Kumari, Nitu, and Sinha, Neha. What is India speaking? Exploring the "Hinglish" invasion. *Physica A: Statistical Mechanics and its Applications*, 449:375–389, 2016. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2016.01.015. URL https://www.sciencedirect.com/science/article/pii/S0378437116000236.

Patwa, Parth, Aguilar, Gustavo, Kar, Sudipta, Pandey, Suraj, PYKL, Srinivas, Gambäck, Björn, Chakraborty, Tanmoy, Solorio, Thamar, and Das, Amitava. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets, 2020.

Poplack, Shana. Dialect acquisition among puerto rican bilinguals. *Language in Society*, 7(1):89–103, 1978. doi: 10.1017/S0047404500005340.

Pratapa, Arti Singh, Singh, Mayank, and Bhattacharyya, Pushpak. Unsupervised hindi to english transliteration using a recurrent neural network with attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 627–633, 2018.

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

Qiu, Xipeng, Sun, Tianxiang, Xu, Yige, Shao, Yunfan, Dai, Ning, and Huang, Xuanjing. Pre-trained Models for Natural Language Processing: A Survey. 3 2020. doi: 10.1007/s11431-020-1647-3. URL http://arxiv.org/abs/2003.08271http://dx.doi.org/10.1007/s11431-020-1647-3.

Salih, Balci, Demirci, Gozde Merve, Hilmi, Demirhan, and Salih, Sarp. Sentiment Analysis Using State of the Art Machine Learning Techniques. In Cezary, Biele, Kacprzyk, Janusz, Wiesław, Kopeć, W., Owsiński Jan, Andrzej, Romanowski, and Marcin, Sikorski (eds.), *Digital Interaction and Machine Intelligence*, pp. 34–42, Cham, 2022. Springer International Publishing. ISBN 978-3-031-11432-8.

Sennrich, Rico, Haddow, Barry, and Birch, Alexandra. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Solorio, Thamar, Liu, Yang, Xie, Pengtao, Mukherjee, Arjun, Vyas, Yash, Mukherjee, Subhabrata, Das, Dipankar, Haffari, Gholamreza, and Nakov, Preslav. Code-switching in multilingual natural language processing: A comprehensive survey. *arXiv preprint arXiv:2105.05205*, 2021.

Sultan, Md Faisal, Chakraborty, Tanmoy, and Kumar, Rajiv. Improving sentiment analysis in code-mixed text using character-level and word-level embeddings. *Expert Systems with Applications*, 155:113406, 2020.

Ver, Pranav and Soares, Murilo. Multilingual sentiment analysis for code-mixed languages. *arXiv preprint arXiv:2008.09202*, 2020.

Wang, Yiqiao. Hinglish sentiment analysis using pretrained models. *arXiv preprint arXiv:2011.10920*, 2020.

Wei, Jason and Zou, Kai. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. 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)*, pp. 6382–6387, 2019.

Xie, Qizhe, Dai, Zihang, Hovy, Eduard, Luong, Minh-Thang, and Le, Quoc V. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2020.

Zaharia, George-Eduard, Vlad, George-Alexandru, Cercel, Dumitru-Clementin, Rebedea, Traian, and Chiru, Costin. UPB at SemEval-2020 task 9: Identifying sentiment in code-mixed social media texts using transformers and multi-task learning. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pp. 1322–1330, Barcelona (online), December 2020a. International Committee for Computational Linguistics. doi: 10.18653/v1/2020.semeval-1.179. URL https://aclanthology.org/2020.semeval-1.179.

Zaharia, Titus, Pai, Aditya, Satpute, Pratik, and Sundararajan, Mukund. Hinglish sentiment analysis using bert. In *Proceedings of the 28th International Conference on Computational Linguistics: Posters*, pp. 2434–2441, 2020b.

Zhang, Hongyi, Cisse, Moustapha, Dauphin, Yann N., and Lopez-Paz, David. mixup: Beyond empirical risk minimization, 2018.