Code-Mixed Language Translation

Adem Odza, Abhirup Mukherjee, Buvana Seshathri, Samiha Kuncham,

The University of Texas at Dallas, CS 6320 Natural Language Processing, Dr. Xinya Du

GitHub: https://github.com/AdemOdza/Multi-Lingual-Text-Translation

Abstract

In our paper, we investigate and compare the performance of different architectures for the translation of code-mixed language. Code-mixed language translation requires the identification of languages as well as the tokenization of different scripts, which makes it a much more complex problem than typical translation tasks. In order to determine the best type of architecture for this problem, we evaluated different combinations of models and word embeddings by comparing their BLEU scores. We found that Transformers using the mBERT embeddings outperformed the other architectures. While the pre-trained multilingual embeddings performed well, the results suggest that architectures specifically designed and trained for code-mixed language translations would have a better performance. These findings should provide guidance for those looking to perform translation on texts with mixed languages.

1 Introduction

Translation has always been a prominent task within natural language processing, especially with the rise of neural networks and machine learning. While existing translation tools are powerful, the rise of informal and code-mixed language poses a problem for these tools.

Code-mixing is the act of combining words from two different languages in speech, typically done by bilingual speakers or between family members who have varying levels of fluency in their common language. The rise of social media and increasingly informal speech online has led to code-mixing becoming a common occurrence in online discussion. Social media platforms typically have translation features for their site content, but these do not always perform correctly when messages contain multi-lingual content. The mixing of languages leads to many problems with translation, such as tokenization of different scripts, precision/size of the tokens, handling languages with different morphemes, and preserving context between languages.

There are many differences between languages that most translation systems might not account for. Languages like English and Spanish might be similarly structured enough to where there is not a significant difference in performance, but what about more distant languages like English and Hindi? They use the Latin and Devanagari scripts respectively, English uses Subject-Verb-Object order while Hindi uses Subject-Object-Verb, and they have varying amounts of Vowels and Consonants. This poses a significant challenge for tokenization and representation of the words in a common way. There is also the problem of identifying which words are in which language and preserving context between languages.

In our paper, we investigate some of the existing models available and compare the results of different architectures that incorporate these models. We investigated architectures using different word embeddings along with different translation models. These architectures were trained and tested with a mixed English-Hindi language corpus as well as some additional Zero-Shot testing with a mixed English-Spanish corpus. We then evaluated the models based on their BLEU scores.

2 Related Work

Work in machine translation and multilingual natural language processing has improved a lot over the recent years, especially in handling the unique cases that are introduced by codemixed languages.

Modern neural machine translation (NMT) has given us the foundation of the sequencetosequence (seq2seq) paradigm [5], which learns an endtoend mapping from an input sequence to a target sequence using neural networks. Sutskever et al. introduced the canonical seq2seq model that uses multilayer Long ShortTerm Memory (LSTM) en-

coders and decoders to transform variablelength inputs into fixedlength representations, and then decode these representations into target sequences. This paper showed good performance on English–French translation benchmarks compared to traditional statistical methods. These seq2seq models were used as the foundation for neural approaches for translation and other sequence prediction tasks. This was done by showing that recurrent architectures could learn representations for both encoding and decoding language data without human intervention.

Even though we see these advances in neural translation, the codemixed language translation part has many other challenges due to the frequent blending of languages while people speak and the lack of annotated parallel data (to verify against). To address such data limitations, recent work has focused on synthetic data generation. The HINMIX_hien dataset that we used in this project gave us largescale parallel HinglishEnglish data by using bilingual corpus data and synthetic codemixing techniques.

Using this work, Kartik et al. provided a framework that uses synthetic data generation with joint learning that improves translation quality for noisy, code-mixed text. Their method generated large code-mixed data [1]. It also uses joint training to share parameters around clean and noisy examples, which means that it gave an improved performance even in zero-shot transfer to new code-mixed variants.

Another major trend in NLP has been the pretraining and finetuning process used by models like BERT (Bidirectional Encoder Representations from Transformers). BERT gave us a deep bidirectional contextual embedding method that captured language representations from large unlabeled data. BERT is also used in many downstream tasks that includes classification, inference, and translation related tasks after doing finetuning.

Even though BERT itself is not a sequencetosequence model, the idea of using a pretrained contextual encoder has affected later translation architectures. It has mainly affected transformerbased models that use pretrained representations or attention mechanisms. These pretrained models improved generalization for codemixed translation where annotated resources are extremely limited.

Previous work has shown that mBERT can be used for codemixed and lowresource translation settings. This is done by making use of mBERT's ability to model mixedlanguage context and crosslingual semantics. mBERT (Multilingual BERT) is an extension of BERT to 100+ languages given to us by jointly pretraining on large multilingual data.

Architecture	Epochs	Training Time (Hrs.)	BLEU Score
Transformer + mBERT	10	23	13.70
Seq2Seq + mBERT	3	1	0.09
Transformer + Learned Embeddings	10	1.25	1.26
Seq2Seq + Learned Embeddings	2	5	1.83

Table 1: The Architectures and their associated BLEU scores.

3 Data

The dataset that was used in this project is the HINMIX dataset. HINMIX is a massive parallel codemixed dataset for Hindi-English code switching.

- This dataset contains 4.2M fully parallel sentences in 6 Hindi-English forms.
- It also consists of a validation set with 280 examples and a test set with 2507 samples.
- HINMIX is a synthetic dataset.
- Hicm includes Hindi sentences with codemix words substituted in English.
- It focuses on Hinglish sentences where Hindi and English are blended in a natural, conversational way.
- This dataset consists of 391,000 unique tokens with both Hindi and English.
- Each Hinglish sentence is paired with a human-annotated English translation.

4 Experiments and Results

4.1 Architecture and Embedding Comparison

We evaluated Transformer and Seq2Seq architectures using both pre-trained multilingual embeddings (mBERT) and learned embeddings. The results can be viewed in Table 1.

Transformer-based models consistently outperformed Seq2Seq models, highlighting the effectiveness of self-attention in capturing long-range dependencies. The combination of **Transformer + mBERT** achieved the highest BLEU score, demonstrating the importance of multilingual contextual embeddings for cross-lingual transfer. Models trained with learned embeddings showed limited performance, indicating insufficient semantic alignment across languages.

Epochs	Number of Samples	BLEU Score
10	500000	1.26
10	1000000	1.30
3	4200000	9.52
5	4200000	11.20
10	4200000	13.70

Table 2: Dataset scale results on Transformer + mBERT

Model	BLEU Score
facebook/nllb-200-distilled-600M	28.40
Helsinki-NLP/opus-mt-mul-en	16.60
RLM-hinglish-translator	15.69
Helsinki-NLP/opus-mt-hi-en	5.34
facebook/m2m100_418M	1.21

Table 3: Pre-Trained Translation Model Scores

4.2 Effect of Dataset Size and Training Epochs

We further analyzed the impact of dataset scale and number of epochs using the Transformer + mBERT model, the results of which can be viewed in Table 2

Increasing the number of training samples resulted in significant improvements in BLEU score. Even with fewer epochs, large datasets led to better performance than small datasets trained for more epochs. Additional epochs provided consistent gains when sufficient data was available. These findings emphasize that **data scale plays a critical role** in effective cross-lingual transfer.

4.3 Comparison with Pre-trained Translation Models

To establish a baseline, we evaluated several pre-trained translation models without fine-tuning. These models are compared in Table 3

Large-scale multilingual models such as NLLB-200 significantly outperformed custom-trained models. Models trained on multilingual or code-mixed data showed better zero-shot generalization. The performance gap highlights the importance of extensive multilingual pretraining.

5 Methodology

We describe the preprocessing methods, model implementations, and model evaluation in this section.

5.1 Data Preprocessing

We had the following approach for tokenizing the data - mBERT tokenizer. The BERT tokenizer uses BPE (byte-pair encoding) subword tokenization. We used the BertTokenizer from HuggingFace transformers [4]. OOV words are handled using subwords. This particular tokenizer had been pretrained on 104 languages. We limited our max sequence length to 128 - anything longer was truncated. [3]

5.2 Model Architectures

We used a simple transformer architecture with self-attention mechanism. This way, we are able to process sequences in parallel and capture long-range dependencies - often surpassing RNNs.

Our transformer architecture (hidden dims = 256) was designed as follows:

1. The encoder consisted of three layers - mutli-head self attention (four heads), FFNN (with a dimension of 1024), layer normalization
2. The decoder had 3 layers that incorporated masked self-attention and cross-attention
3. Sinusoidal positional embedding

A dropout rate (0.01) was applied after the attention and feed-forward layers.

Our seq2seq architecture was designed as follows:

1. Bidirectional LSTM encoder two layers and 256 dimensions (since the bidirectional LSTM concatenates forward and backward hidden states)
2. LSTM decoder
3. BahadanauAttention [2] - This is a differentiable attention model without the unidirectional alignment limitation. When predicting a token, if not all the input tokens are relevant, the model aligns (or attends) only to parts of the input sequence that are deemed relevant to the current prediction. This is then used to update the current state before generating the next token.

5.3 Embeddings

We used the frozen bert-base-multilingual-cased embeddings - this is the pretrained BERT embedding trained on a massive multilingual corpus. We did this to preserve the mutli-lingual embedding space and also reduce our trainable parameters for our models. We also hoped that using pre-trained embeddings would give the model better zero-shot capabilities.

Hyperparameter	Transformer	Seq2Seq
Hidden Dimension	256	256
Layers	2	2
Attention Heads	8	-
Feed-Forward Dim.	1024	-
Dropout	0.1	0.1
Embedding Dim.	768 to 256	768 to 256

Table 4: Hyperparameters on the Transformer and Seq2Seq models

5.4 Evaluation

For evaluation, we used the BLEU (Bilingual Evaluation Understudy) for our primary evaluation metric - we used the sacrebleu library. The BLEU score is a measure of n-gram overlap between the predicted and reference translations. The scores range between 0-100 with models around the 20-30 BLEU points producing coherent translations.

6 Implementation

We implemented the following models: transformer and seq2seq with mBERT embeddings and compared our performance to baseline scores of learned embeddings.

First we loaded up the embeddings from mBERT

```
# Tokenizer + mBERT embeddings

#Multilingual BERT tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-multilingual-cased")

#BERT embeddings
mbert_model = BertModel.from_pretrained("bert-base-multilingual-cased")

# we're only using the mbert word embeddings
mbert_embeddings = mbert_model.embeddings.word_embeddings

# freeze embeddings (don't update while training)
mbert_embeddings.requires_grad_(False)
```

Then we tokenized both the source (hindi-english code mixed) and target (english) sentences using our loaded tokenizer. mBERT applies BPE to handle OOV words by breaking them up into subwords. During training, we also shift the target sequence that creates input output pairs for the decoder. Thus, at each timestep, the decoder receives all previous tokens and learns to predict the next token.

```
def batch(source_texts, target_texts, maxlength=MAX_LEN):
    sourcenc = tokenizer(
        source_texts,
        padding=True,
        truncation=True,
        max_length=maxlength,
        return_tensors="pt"
    )
    targenc = tokenizer(
        target_texts,
        padding=True,
        truncation=True,
        max_length=maxlength,
        return_tensors="pt"
    )
    s = sourcenc["input_ids"].to(DEVICE)
    t = targenc["input_ids"].to(DEVICE)


    sourcepaddingmask = (sourcenc["attention_mask"] == 0).to(DEVICE)
    targetpaddingmask = (targenc["attention_mask"] == 0).to(DEVICE)
    # shift target
    input = t[:, :-1]

    out = t[:, 1:] # Remove first token
    targetpaddingmask = targetpaddingmask[:, :-1]
    return {
        "source_tokens": s,
        "input": input,
        "out": out,
        "sourcepaddingmask": sourcepaddingmask,
        "targetpaddingmask": targetpaddingmask,
    }
```

This is our main training loop using AdamW as the optimizer and Cross Entropy loss. After gradient computation using back-propagation, we also applied a clipping of 1.0 to prevent explosion of the gradients.

```
def train_model(model, dataloader, epochs=10, lr=5e-4, model_name="model"):
    optimizer = optim.AdamW(model.parameters(), lr=lr)
    criterion = nn.CrossEntropyLoss(ignore_index=0)

    for epoch in range(1, epochs + 1):
        for batch in tqdm(dataloader):
            ..
            # Forward pass
            logits = model(src, tgt_in, src_mask, tgt_mask)
            ..
            # Compute loss
            loss = criterion(logits.reshape(-1, logits.size(-1)),
                             tgt_out.reshape(-1))
            ..
            # Backward pass
            optimizer.zero_grad()
            loss.backward()
            torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
            optimizer.step()
```

Then, we calculated the BLEU score using 500 of our test samples.

```
bleu score calculation
def calculate_bleu(model_name, translate_fn, num_samples=500):
    #compute BLEU score on test data
    test_src_lines = open(test_source_path, "r", encoding="utf-8").read().splitlines()[:num_samples]
    test_tgt_lines = open(test_target_path, "r", encoding="utf-8").read().splitlines()[:num_samples]

    predictions = []
    references = []

    for src_text, ref_text in tqdm(islice(zip(test_src_lines, test_tgt_lines), total=len(test_src_lines), desc=model_name):
        pred_text = translate_fn(src_text)
        predictions.append(pred_text)
        references.append(ref_text)


    bleu = BLEU()
    score = bleu.corpus_score(predictions, [references])
    return score.score, predictions, references
```

7 Conclusion

We investigated cross-lingual code-mixed Hindi-English translation in this project experimenting with different embedding strategies and architectures. Our results show that Transformer-based architecture performs better compared to Seq2Seq-based architecture. The main reason behind this would be the self-attention mechanism of Trans-

formers, which supports the capturing long-range dependencies.

Using multilingual pre-trained embeddings (i.e., mBERT) produced significantly better results than using learned embeddings and which shows us the importance of cross-lingual semantic alignment between the two languages. Our experiments also demonstrated the importance of large datasets on performance in low-resource environments, with dataset size having a larger impact than the number of training epochs.

Our final model achieved a BLEU score of 13.70, however larger pre-trained models such as NLLB-200 outperforms our model.These findings indicate that models with extensive pretraining outperform all others, even under zero-shot evaluation. Our future research will focus on developing efficient means of fine-tuning large pre-trained models, zero-shot testing on unseen code-mixed data, and cross-lingual transfer to other language pairs.

The results support the conclusion that Transformers, combined with multilingual pre-training, form the foundation for building strong models for cross-lingual code-mixed translation.

Bibliography

[1] K. Kartik, S. Soni, A. Kunchukuttan, T. Chakraborty, and M. S. Akhtar, "Synthetic Data Generation and Joint Learning for Robust Code-Mixed Translation," arXiv.org, 2024. https://arxiv.org/abs/2403.16771.

[2] A. Zhang, Z. Lipton, M. Li, and A. Smola, "The Bahdanau Attention Mechanism - Dive into Deep Learning 1.0.3," D2l.ai, 2014. https://d2l.ai/chapter_attention-mechanisms-and-transformers/bahdanau-attention.html

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," ArXiv, Oct. 11, 2018. https://arxiv.org/abs/1810.04805

[4] Google, "google-bert/bert-base-multilingual-cased · Hugging Face," huggingface.co, Mar. 11, 2024. https://huggingface.co/google-bert/bert-base-multilingual-cased

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," arXiv.org, 2014. https://arxiv.org/abs/1409.3215