Code Mix Generation Project Report

Team HSV

Team Information

Srihitha Mallepally	**Vedanivas Chowdary**	**Mareddy Sriharshitha**
2021101043	**2021101037**	**2021101067**

Introduction

In multilingual societies, individuals often mix languages in their communication, a phenomenon known as code mixing. This project aims to tackle the task of generating code-mixed text (Hinglish). Code mixing, particularly in online platforms and social media, presents unique challenges due to its informal nature and varied linguistic patterns. The project will explore statistical and neural algorithms to generate coherent and natural-sounding code-mixed Hindi and English text.

Hinglish, a fusion of Hindi and English, is a prominent example. Here's an example of a Hinglish text:

> *Kal mere dost ne mujhe ek amazing movie recommend ki, aur maine uski trailer dekhi. Bohot interesting lag rahi hai, can't wait to watch it!*

In this example, the speaker seamlessly integrates Hindi and English, expressing thoughts using a combination of both languages. This phenomenon is common in informal conversations, especially on social media platforms, where individuals switch between languages based on context, audience, or personal preference.

Objectives

The primary objectives are as follows:

- **Develop advanced language models for code-mixed text generation:** We explored several techniques to produce fluent and grammatically correct code-mixed text, specifically for Hinglish. These are:
  - Neural Language Models: Experiment with these architectures:
    - LSTM seq2seq models: These capture long-term dependencies within sequences.
    - Transformer-based models: Explore multilingual pre-trained models like mBART and mT5, considering their ability to handle multiple languages simultaneously.
- **Comprehensive Evaluation:** Design a robust evaluation strategy using a suitable mix of these metrics:

    - BLEU: Evaluates the similarity of generated text against human-written code-mixed references.
    - CMI: Evaluate the occurrence of code-mixing.
    - Human Evaluation: Native Hinglish speakers will judge the realism and fluency of the generated text.
- **Comparative Analysis:** Comparison of the performance of various models and techniques. Based on the evaluation metrics, analyzing which approaches prove most effective for handling the nuances of Hinglish code-mixing.

2. Datasets Utilized

To train our models, we have leveraged two significant datasets:

LinCE English - Hinglish (ENG - HINGLISH) Dataset

- A comprehensive dataset for linguistic code-switching evaluation, LinCE provides a rich assortment of code-mixed text examples, serving as a critical resource for training our models in recognizing and generating code-switched language patterns.

CMU Hinglish DoG Conversations: Dataset

- Specifically compiled for Hinglish text, this dataset offers a wide variety of English-Hindi code-mixed sentences, drawn from diverse sources. It is instrumental in teaching our models the intricacies of Hinglish, enabling them to produce translations that are not only grammatically correct but also contextually appropriate.

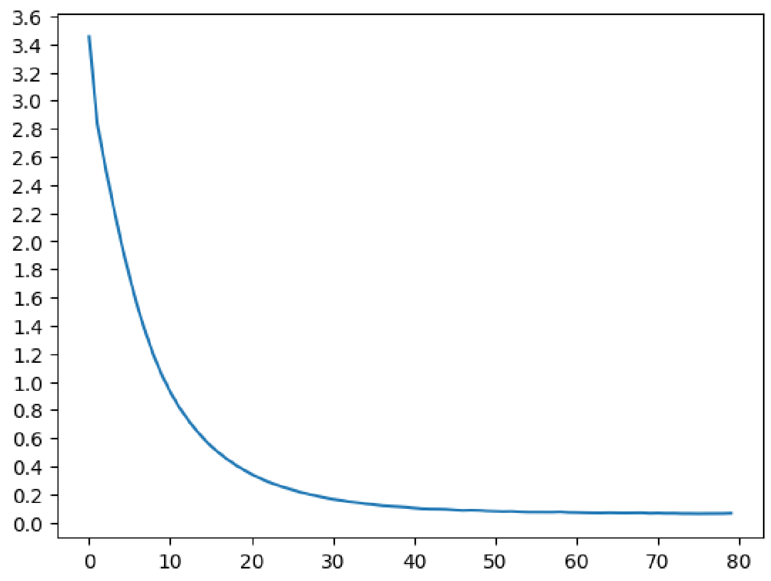
Using these datasets combined, we acknowledge the need for a broader and more diverse corpus to capture the full spectrum of code-mixing phenomena and plan to extend our dataset collection further.

3. Models

1. Initial GRU Model

Our first approach involved utilizing Gated Recurrent Unit (GRU) networks, favored for their simplicity and efficiency in processing sequential data. This model laid the groundwork for our exploration into neural network-based code-mixed translation, offering initial insights into the challenges and potential strategies for generating Hinglish text.

Training Loss:

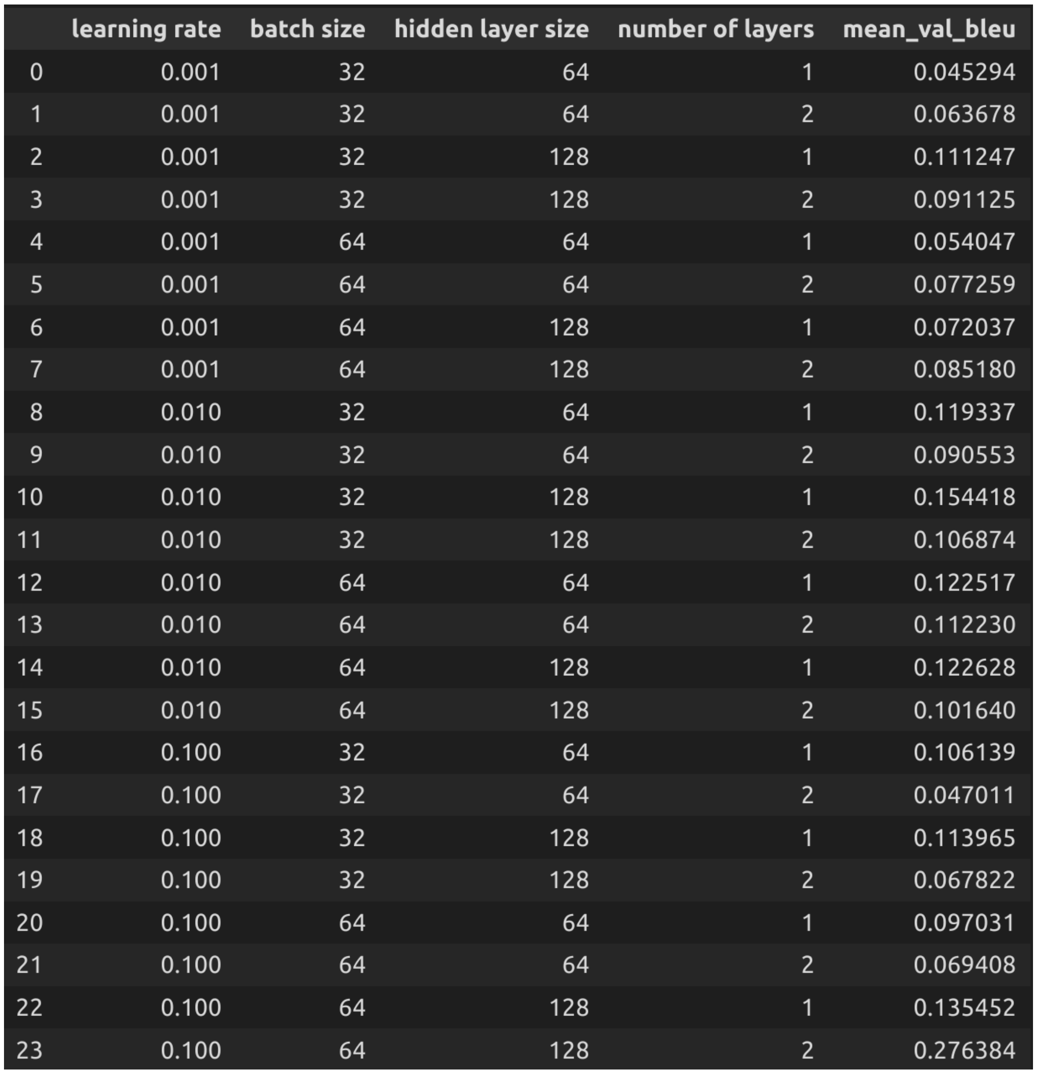


Outputs for visualisation:

- On some sentences of test set:

```
> i never knew it was based on a novel
= muje patha hi nahi ki voh novel ke based the
< nahi lagta hai woh kuch nahi hai

> the wolf of wallstreet
= the wolf of wallstreet
< jarur track the post side track ka score play kartha

> i m a movie buff in general
= main movie ke bare mein bahut janta hun
< mein maantha hoon mein ek dream mein ek scene shikaar unmaad

> yeah i am sure the government was shocked
= haan mujhe pata he government shock hua
< haan lakin tom ko kuch bothered nahi huva

> ok done
= ok theek hai
< ok tho great aur yeh main lamba time dekh kar raha

> did he get convicted
= kya wo convicted hai
< voh phone call tho mera favorite scene hein

> okay d
= okday
< acha tho mein dekh ke baare mein kuch dekha hi

> no its from
= nahin its from
< acha tho yeh sahi ka pasand hein

> still there
= abhee tak vaheen
< mere paas abhi tak task ne isey nahi saktha
```

Hyperparameter Tuning



	learning rate	batch size	hidden layer size	number of layers	mean_val_bleu
0	0.001	32	64	1	0.045294
1	0.001	32	64	2	0.063678
2	0.001	32	128	1	0.111247
3	0.001	32	128	2	0.091125
4	0.001	64	64	1	0.054047
5	0.001	64	64	2	0.077259
6	0.001	64	128	1	0.072037
7	0.001	64	128	2	0.085180
8	0.010	32	64	1	0.119337
9	0.010	32	64	2	0.090553
10	0.010	32	128	1	0.154418
11	0.010	32	128	2	0.106874
12	0.010	64	64	1	0.122517
13	0.010	64	64	2	0.112230
14	0.010	64	128	1	0.122628
15	0.010	64	128	2	0.101640
16	0.100	32	64	1	0.106139
17	0.100	32	64	2	0.047011
18	0.100	32	128	1	0.113965
19	0.100	32	128	2	0.067822
20	0.100	64	64	1	0.097031
21	0.100	64	64	2	0.069408
22	0.100	64	128	1	0.135452
23	0.100	64	128	2	0.276384

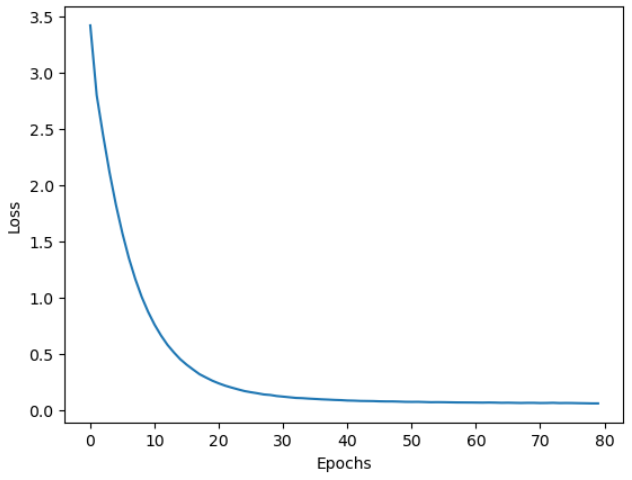
From the hyperparameters tuning we got that the best performing model has parameters

- Learning rate = 0.100
- Batch size = 64
- Hidden layer size = 128
- Num of layers = 2

2. Refinement with GRU and Attention Mechanisms

Further refinement of our approach led us back to GRU models, this time incorporating attention mechanisms. The addition of attention was designed to improve the model's capacity for focusing on relevant aspects of the input text, enhancing the accuracy and contextual relevance of the translated Hinglish sentences. This iteration represented a significant step forward in our project, combining the efficiency of GRU models with the sophisticated targeting capabilities provided by attention mechanisms.

Training Loss:

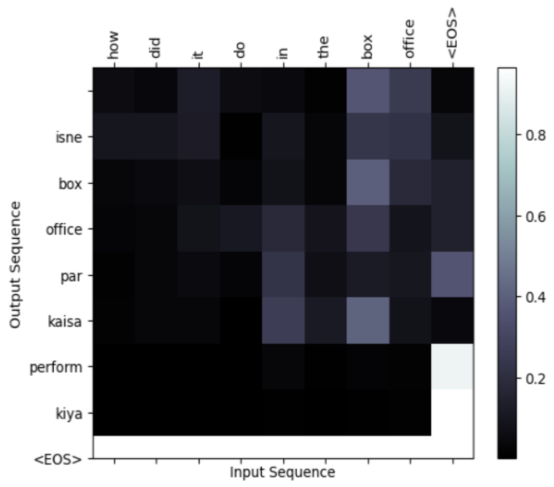


Outputs for visualization:

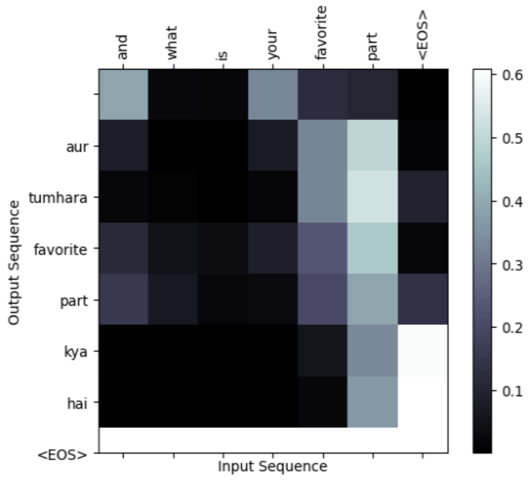
- On some sentences of test set

```
> i think you watched this movie also
= mujhe lagtha hai ke aap is movie bhi dekha hain
< mujhe lagta hai ye movie tumne dekhi hai <EOS>

> cool story factor about him tryingto be fired
= usake baare mein shaant kahaanee kaarak nikaal diya jaega
< kya wo uske baare me koi big role batao <EOS>

> she is
= vah hai
< wo bohuth ysterious hai <EOS>

> and what is your favorite part
= aur tumhara favorite part kon sa he
< dusra favorite part konsa hein <EOS>

> yes the director of it was morten tyldum
= haan isake nirdeshak mortan taildam the
< haan sahi hein voh director morten tyldum hein <EOS>

> yea i think maleficent cheated on him or something
= haan i think maleficent ne usay dhoka diya ya kuch
< mujhe wo ek lagti hai achhi lagi <EOS>
```

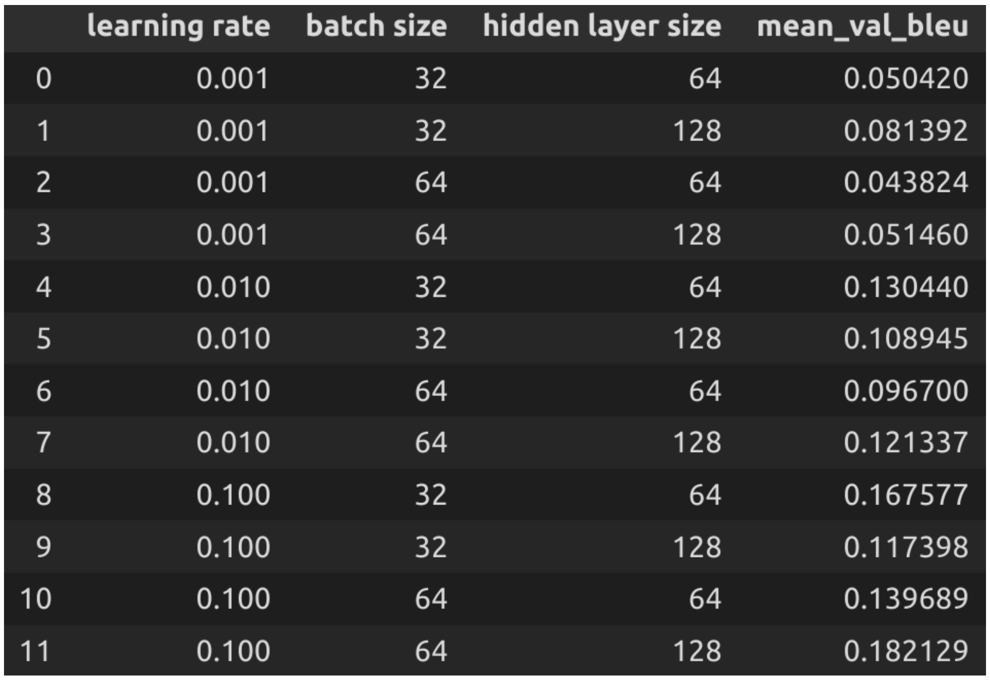
```
input = how did it do in the box office
output = isne box office par kaisa perform kiya <EOS>
```



```
input = and what is your favorite part
output = aur tumhara favorite part kya hai <EOS>
```



Hyperparameter tuning

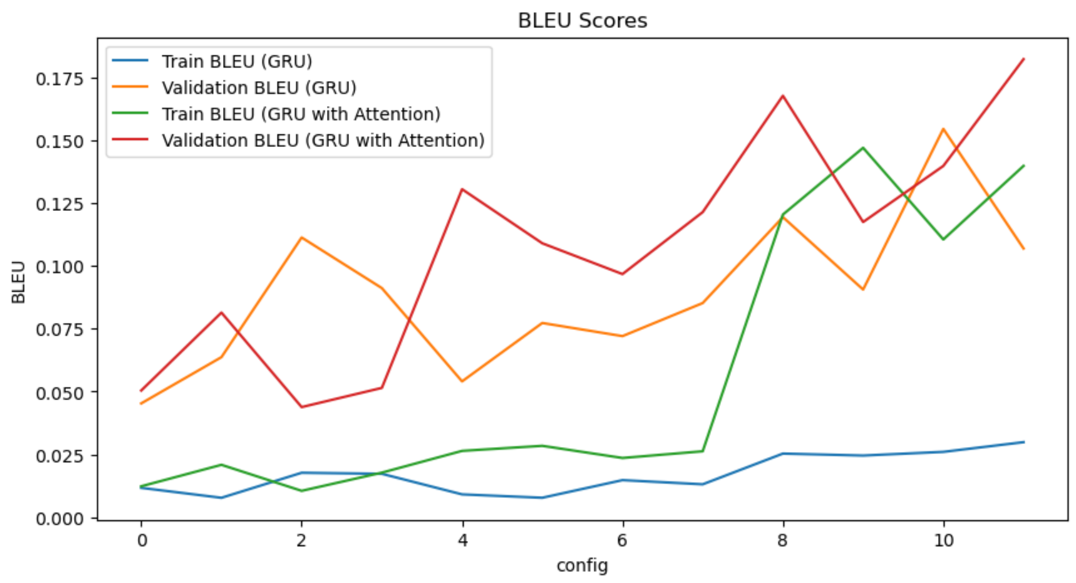


	learning rate	batch size	hidden layer size	mean_val_bleu
0	0.001	32	64	0.050420
1	0.001	32	128	0.081392
2	0.001	64	64	0.043824
3	0.001	64	128	0.051460
4	0.010	32	64	0.130440
5	0.010	32	128	0.108945
6	0.010	64	64	0.096700
7	0.010	64	128	0.121337
8	0.100	32	64	0.167577
9	0.100	32	128	0.117398
10	0.100	64	64	0.139689
11	0.100	64	128	0.182129

From the hyperparameters tuning we got that the best performing model has parameters

- Learning rate = 0.100
- Batch size = 64
- Hidden layer size = 128

Comparision of BLEU Scores on GRU:



On test set: (from 1-gram to 8-gram)

GRU Without Attention: [0.08942, 0.015402, 0.00515, 0.00236, 0.000922, 0.0, 0.0, 0.0]

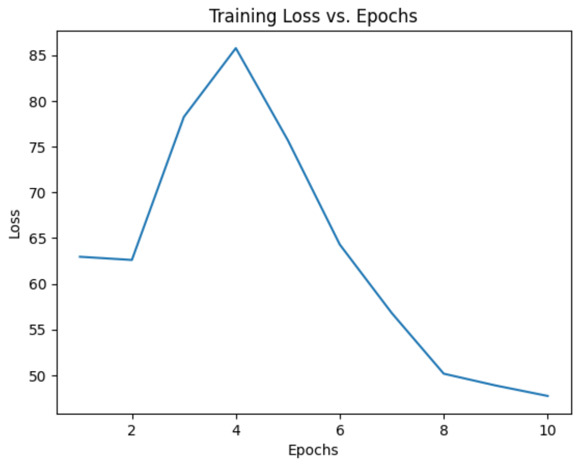
GRU With Attention: [0.120989, 0.031799, 0.009756, 0.004685, 0.002282, 0.000845, 0.0, 0.0]

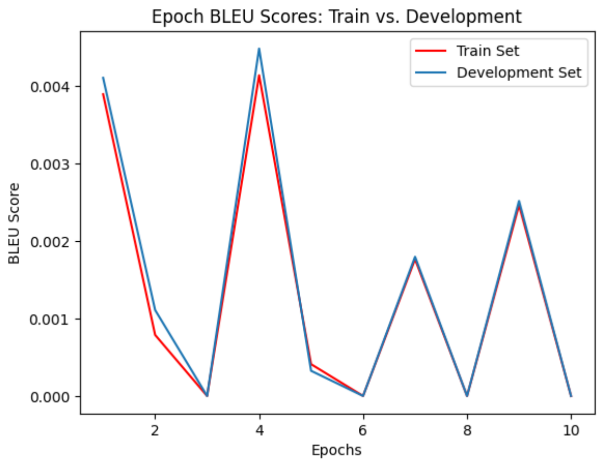
Conclusion:

GRU with attention outperforms GRU without attention on the test set for BLEU score because attention mechanisms enable the model to focus on relevant parts of the input sequence, aiding in capturing long-range dependencies and improving translation quality.

3. Advancement to LSTM Model

Building upon our initial findings, we transitioned to Long Short-Term Memory (LSTM) networks, known for their enhanced ability to capture long-term dependencies in text data. The shift to LSTM aimed to address some of the limitations observed with the GRU model, particularly in handling complex sentence structures and nuanced language features inherent in code-mixing.





learning_rate	dropout	hidden_size	embedding_size	attention	bleu_score
0.100	0.5	128	128	True	0.061230
0.100	0.5	64	64	False	0.068300
0.100	0.5	256	256	False	0.077900
0.100	0.5	128	128	False	0.079600
0.100	0.5	64	64	True	0.117270
0.100	0.5	256	256	True	0.131562
0.010	0.5	256	256	False	0.198800
0.010	0.5	128	128	False	0.213600
0.010	0.5	128	128	True	0.251508
0.010	0.5	64	64	False	0.270600
0.001	0.5	64	64	False	0.277900
0.001	0.5	128	128	False	0.300100
0.010	0.5	256	256	True	0.300541
0.010	0.5	64	64	True	0.307635
0.001	0.5	64	64	True	0.322172
0.001	0.5	128	128	True	0.347912
0.001	0.5	256	256	False	0.351900
0.001	0.5	256	256	True	0.401365

4. Transformers

The Transformer in NLP is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. Transfer Learning is a technique where a deep learning model trained on large dataset (pre-trained model) is used to perform similar tasks on another dataset (fine tuning).

For our project we used Huggingface's Transformer for our code-mixed generation task as the library provides tons of pre-trained models that we can either directly use or fine-tune on our tasks.

mT5

We'll be using publicly available mT5 pre-trained checkpoints, which is essentially a multi-lingual version of T5 (Text To Text Transfer Transformer).

- T5 is also a pre-trained language model based on un-labeled data, the main distinction is it re-formulates all text-based NLP problem like translation, into a sequence-to-sequence setting that can be solved by an encoder-decoder architecture.
- mT5 uses a sentence piece tokenizer and scaled up to 250,000 word pieces as vocabulary size compared to the 32,000 of T5.
- mT5 was only pre-trained on mC4 excluding any supervised training. Therefore, this model doesn't include a task prefix like the original T5 model and must be fine-tuned before it is usable on a downstream task.

We fine-tuned the model on our combined dataset specified above.

Number of parameters: 300176768

```
model_checkpoint: str = "google/mt5-small"

tokenizer = AutoTokenizer.from_pretrained(config.model_checkpoint, cache_dir=config.cache_dir)
```

Finetuning

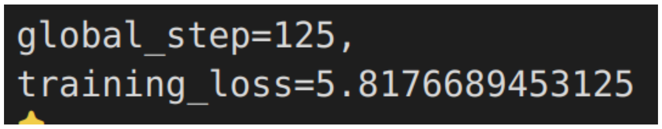
Fine-tuning a sequence-to-sequence (seq2seq) model has a great advantage. Seq2seq By fine-tuning with a seq2seq approach, the model's ability to capture contextual information from the input English data and generate code-mixed Hindi-English text is better.

One of the main advantages of seq2seq models is because they have flexibility in output length, enabling them to handle diverse linguistic complexities inherent in code-mixed text generation.

This results in optimized performance and high-quality code-mixed output.

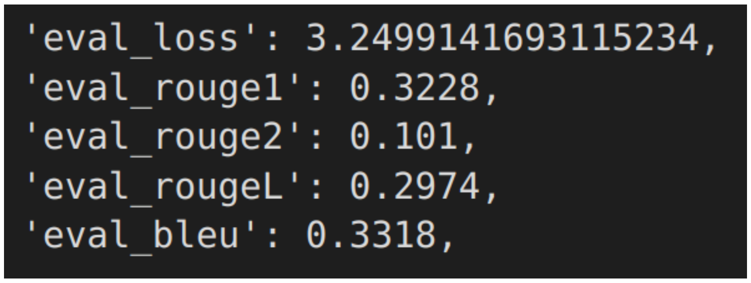
These are the results we got for fine tuning

On Train set:

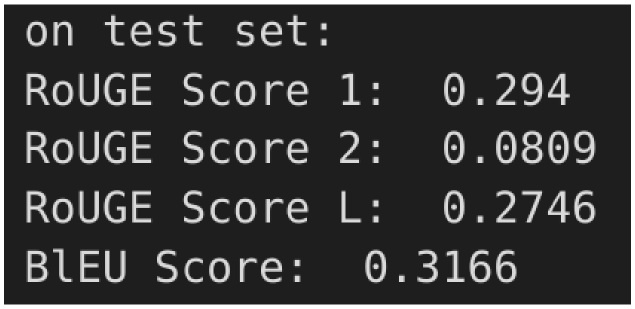


```
global_step=125,
training_loss=5.8176689453125
```

On validation set:



```
'eval_loss': 3.2499141693115234,
'eval_rouge1': 0.3228,
'eval_rouge2': 0.101,
'eval_rougeL': 0.2974,
'eval_bleu': 0.3318,
```



```
on test set:
RoUGE Score 1:  0.294
RoUGE Score 2:  0.0809
RoUGE Score L:  0.2746
BlEU Score:  0.3166
```

mBART

mBART (Multilingual BART) excels in handling code-mixed language tasks like Hindi and English by leveraging its pre-training on a diverse multilingual corpus. Its bidirectional architecture captures contextual dependencies across languages enabling accurate understanding and generation of code-mixed text, while auto-regressive generation ensures fluent outputs. Its bidirectional nature captures contextual dependencies across languages. By fine-tuning mBART on code-mixed data, it becomes adept at tasks like translation or named entity recognition, making it a powerful choice for our project.

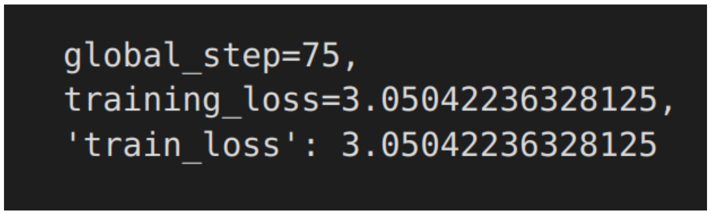
number of parameters: 610879488

model_checkpoint: str = "facebook/mbart-large-50-many-to-many-mmt"

tokenizer = MBart50TokenizerFast.from_pretrained(config.model_checkpoint, cache_dir=config.cache_dir)

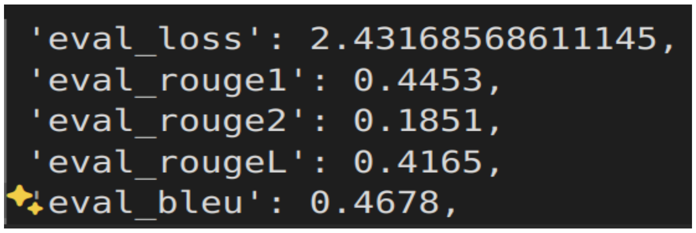
Finetuning

Fine-tuning models like mBART for code-mixed English-Hindi text offers similar advantages. These models can understand the mix of languages and produce text that fits the context well. By training them specifically on code-mixed data, they learn how to handle language switching and create accurate output. Leveraging pre-trained mBART models speeds up this process, requiring fewer examples to learn effectively. Ultimately, this approach helps develop useful applications for multilingual environments and advances research in handling mixed languages.

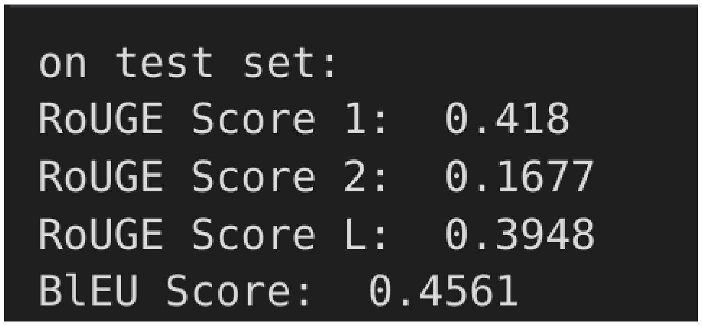


```
global_step=75,
training_loss=3.05042236328125,
'train_loss': 3.05042236328125
```

On val set:



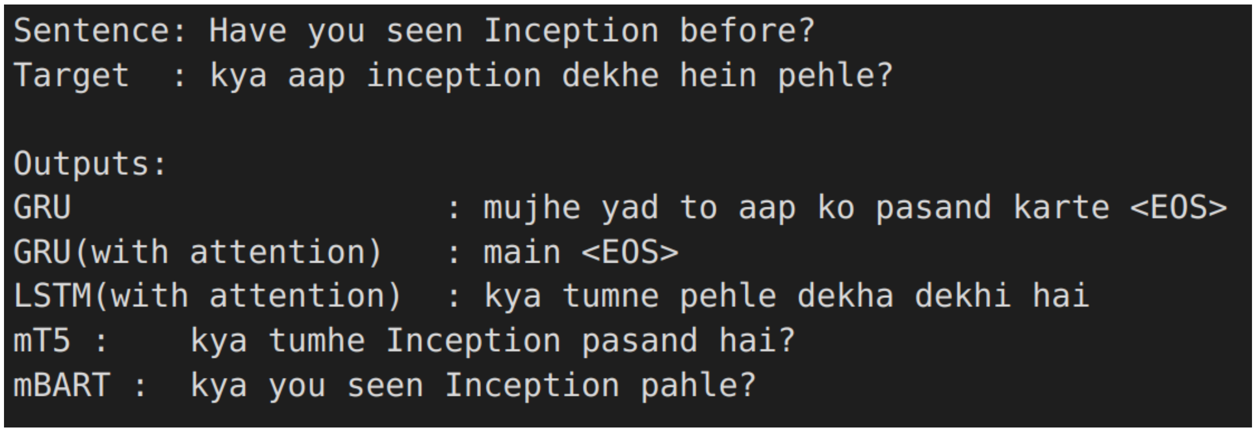
```
'eval_loss': 2.43168568611145,
'eval_rouge1': 0.4453,
'eval_rouge2': 0.1851,
'eval_rougeL': 0.4165,
'eval_bleu': 0.4678,
```



```
on test set:
RoUGE Score 1:  0.418
RoUGE Score 2:  0.1677
RoUGE Score L:  0.3948
BlEU Score:  0.4561
```

Best models for transformers and the prediction results are available at link

Comparision of models



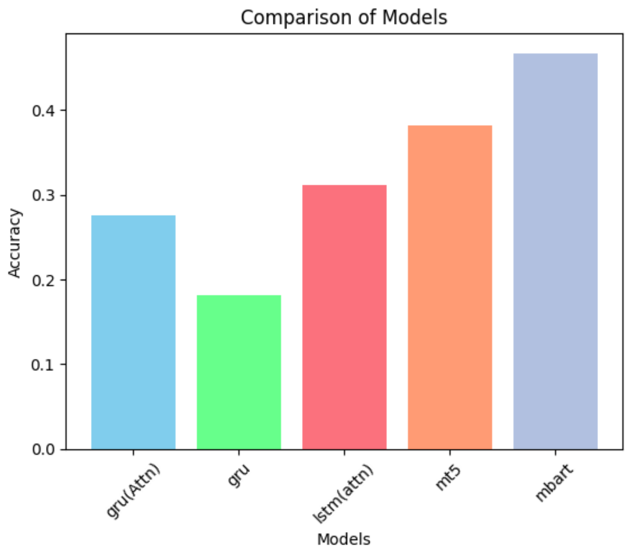
```
Sentence: Have you seen Inception before?
Target  : kya aap inception dekhe hein pehle?

Outputs:
GRU                     : mujhe yad to aap ko pasand karte <EOS>
GRU(with attention)     : main <EOS>
LSTM(with attention)    : kya tumne pehle dekha dekhi hai
mT5 :    kya tumhe Inception pasand hai?
mBART :  kya you seen Inception pahle?
```

As we can see from the above sentence from test set

The performance of advanced models such as mT5 and mBART significantly surpasses that of traditional architectures like GRU and LSTM, even when augmented with attention mechanisms.

The superiority of mT5 and mBART is because of extensive pre-training on large-scale multilingual corpora, hence it has superior conceptual understanding and linguistic representation capabilities.

Despite the enhancements achieved with attention mechanisms, GRU-based models continue to lag behind the performance benchmarks set by mT5 and mBART. Hence pretrained models are the preferred choice for our code-mixed generation



So, our understanding of how the models performed is:

GRU < GRU (attention) < LSTM (attention) << mT5 < mBART

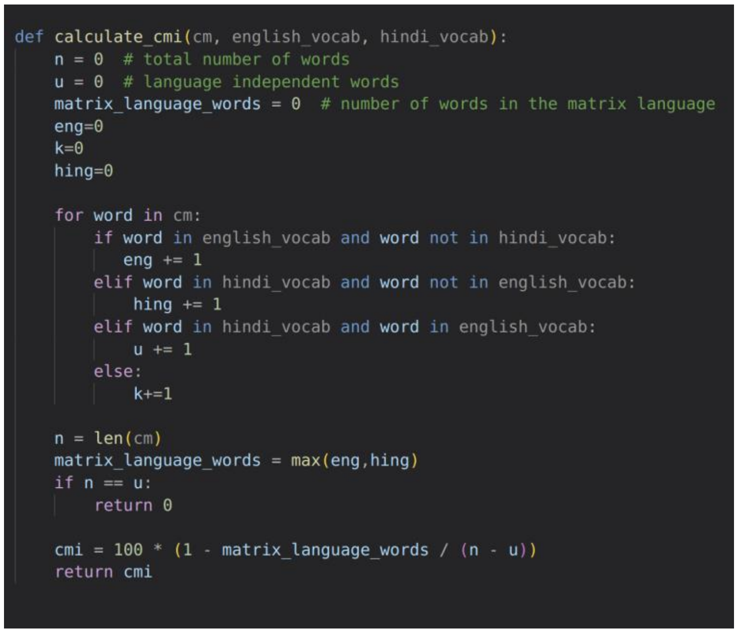
BLEU Scores

BLEU (Bilingual Evaluation Understudy) scores were utilized to evaluate the quality of machine translation output across all models. BLEU compares n-gram overlap between machine translation output and reference translations, providing a score ranging from 0 to 1, with 1 indicating a perfect match. The metric's language-independent nature made it a suitable choice for evaluating machine translation systems and tracking system progress over time.

CMI scores

The Code-Mixing Index (CMI) is a measure used to quantify the occurrence of code-mixing within a linguistic dataset. CMI calculates the proportion of code-mixed instances relative to the total number of language instances present in the dataset.

Relying on just the BLEU score for assessing translations can misrepresent the quality of translations, as models could generate monolingual outputs and still have a basic BLEU score due to n-gram overlap. Hence, we included CMI for evaluation of models to help in better assessing the quality of the outputs of code-mixed data



```python
def calculate_cmi(cm, english_vocab, hindi_vocab):
    n = 0  # total number of words
    u = 0  # language independent words
    matrix_language_words = 0  # number of words in the matrix language
    eng=0
    k=0
    hing=0

    for word in cm:
        if word in english_vocab and word not in hindi_vocab:
           eng += 1
        elif word in hindi_vocab and word not in english_vocab:
            hing += 1
        elif word in hindi_vocab and word in english_vocab:
            u += 1
        else:
            k+=1

    n = len(cm)
    matrix_language_words = max(eng,hing)
    if n == u:
        return 0

    cmi = 100 * (1 - matrix_language_words / (n - u))
    return cmi
```

For mBART

```
56.013745704467354 35.799790513242556
```

For mt5

```
35.57929334428923 35.799790513242556
```

The 1st score is for our model generated text, 2nd value is of the original hinglish text from data.

MBART has more code mixing than hinglish

Challenges while training

LSTM

Implementing a teacher-forcing mechanism in LSTM required exploring various architectures, making observations, and experimenting with different approaches. The training time was notably longer compared to the other models, spanning roughly 10 to 11 hours without GPU.

GRU

Similar to LSTM, implementing a teacher-forcing mechanism in GRU involved extensive experimentation with different architectures. The training time was also relatively longer compared to models with attention mechanisms.

mT5

Fine-tuning mT5 with the entire dataset posed a significant challenge due to memory constraints. The model required excessive RAM, surpassing the limits of platforms like Google Colab. As a workaround, the amount of training data had to be reduced. Adjusting the number of epochs did not alleviate the memory constraints.

mBART

Similar to mT5, training mBART encountered memory issues, necessitating a reduction in the training dataset size. Despite adjusting the number of epochs, memory usage remained a persistent challenge.

Challenges post training

- Post-training, running LSTM for more than 50 epochs was necessary to observe any discernible results. However, the quality of results was suboptimal, characterized by repetitive word usage.
- Challenges arise when relying solely on the BLEU score to assess translations. While BLEU is a widely used metric, it can misrepresent translation quality. Models might produce monolingual outputs that happen to have a basic BLEU score because of n-gram overlap, without genuinely capturing the essence or fluency of the translated text. This limitation underscores the need for a more nuanced evaluation approach that considers factors beyond simple token overlap.
- By randomly sampling a subset of our dataset, we identified two classes of translation errors:
    - significant semantic distortions in the target compared to the source (Error)
    - minor discrepancies that did not alter the overall semantic content (No Error).
    - English Sentence: I think I know the football player it was based on.
      Hinglish Sentence: Muje lagtha ki yeh football player ke baare mein hein.
    - There are substantial amounts of semantic distortion in the target. These findings underscore the importance of high-quality datasets for effective model training, as errors within the data can hinder training and performance.

Conclusions

1. Model Ability:

- The transformer models (mT5 and mBART) outperformed traditional RNN-based models (GRU, LSTM) in generating code-switched sentences, as evidenced by higher BLEU scores.

- The attention mechanisms significantly contributed to understanding contextual nuances and aligning parts of sentences across languages.

2. Data Quality Impact:

- The lack of high-quality, well-annotated code-switched training data limited the models' potential to learn complex code-switching patterns effectively.

- Improving data quality and annotations could potentially lead to better model performance and more natural code-switched sentence generation.

3. Future Directions:

- Data Enhancement: Collecting and annotating high-quality code-switched data or using techniques like data augmentation to improve existing datasets could enhance training outcomes.

- Model Exploration: Further exploration of newer architectures or custom modifications to existing models could yield improvements in handling the intricacies of code-switching.

- Evaluation Metrics: Alongside BLEU and CMI, incorporating other evaluation metrics that specifically measure the quality and authenticity of code-switching could provide a more holistic view of model performance.

References

- https://www.nltk.org/_modules/nltk/translate/bleu_score.html
- https://towardsdatascience.com/a-comprehensive-guide-to-neural-machine-translation-using-seq2sequence-modelling-using-pytorch-41c9b84ba350
- https://huggingface.co/docs/transformers/en/model_doc/mt5
- https://github.com/huggingface/transformers
- https://huggingface.co/docs/transformers/en/model_doc/mbart
- https://aclanthology.org/L16-1292.pdf

Presentation Link - https://www.canva.com/design/DAGEr3FthNI/1_94LnULZKL3H_3qPZIvCg/edit?utm_content=DAGEr3FthNI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton