Code Mix Generation- Interim Submission

Project Components

We refer to the following papers in our project:
https://arxiv.org/pdf/2105.08807.pdf
https://aclanthology.org/P18-1143.pdf
As proposed in these papers, we plan to fine-tune a pre-trained multilingual model such as mBERT or mT5. This is done using synthetic code-mixed data followed by real CM data using the curriculum training as described in the papers. Various methods for generating code-mixed data are described at https://arxiv.org/pdf/2105.08807.pdf we will implement and compare some of them.

As such, the project can be divided into 6 parts.

1. Modify the OPUS dataset using cumulative n-grams.
2. Train Word2Vec on the dataset.
3. Create a synthetic dataset using n-gram translation.
4. Get pre-trained mT5 and finetune on generated data.
5. Fine-tune on real data.
6. Experiments with generating synthetic datasets.

Progress So Far

1. Modifying OPUS dataset using cumulative n-grams.

Sample sentence from the modified dataset:

> ['i_know', 'जानता', 'कैसे_जानता', 'how_do', 'do_i', 'मैं', 'how', 'जानता_हूँ', 'हूँ', 'know', 'i', 'कैसे', 'मैं_कैसे', 'do']

Ideas to explore:

Explore the role of tokenization in sentences of the OPUS dataset on the performance of the word2vec model in computing the similarity of words

2. Train Word2Vec

Examples of words with similar W2V representations (cosine similarity):

> <और, and>: similarity=0.9570194482803345

> <of, की>: similarity=0.7434549927711487

('और',) and ('fire',) is 0.2974220812320709

3. Creating a Synthetic dataset

using text similarity that we got from training Word2Vec.

with x probability, we replace the English words with their highest similar Hindi word. Thus generating a Hinglish dataset.

4. Fine-tuning pre-trained mT5 (partially complete)

Since we are working parallelly we first tried to finetune mt5 using

PHNIC dataset.

We clean the dataset and split it into train test and valid data.

we then fine-tune using the mT5 model, which is the multilingual version of T5.

The T5 model may carry out numerous tasks in a single model, including translation, linguistic acceptability (CoLA), semantic similarity (STSB), summarization, correction, etc. The original T5 model is based on a study on transfer learning in NLP.

For instance, the format of the input (source text) should be "summarize: TEXT" when you want to summarize text using a pre-trained T5 model. "Translate to hinglish" should be used when translating from English to hinglish.

mT5 is pre-trained only in an unsupervised manner with multiple languages, and it’s not trained for specific downstream tasks. hence we will fine tune it to train it to give us hinglish texts.

we are using **MT5ForConditionalGeneration, MT5Tokenizer from transformers,Seq2SeqTrainer, Seq2SeqTrainingArguments.**

To generate inputs for fine-tuning, we tokenize text into ids.

using **MT5Tokenizer.**

finetuning:

We are using the sequence-to-sequence (Seq2Seq) approach. Seq2Seq models are neural network models that map input sequences to output sequences. In code-mixed text generation, the input sequence is a prompt or context, and the output sequence is the code-mixed text generated by the model. Fine-tuning the MT5 model involves updating the weights of the pre-trained model using the code-mixed text corpus.

TODO / Work Plan

4. Fine-tuning pre-trained mT5

5. Fine tune on real data

1. **Hyperparameter tuning**: Experiment with different hyperparameters such as the learning rate, batch size, and the number of training epochs to find the best combination that results in the highest model performance on the validation set.

6. Experiments

1. **Evaluation:** Evaluate the performance of the fine-tuned MT5 model on the test set using metrics such as BLEU score, perplexity, and accuracy.
2. Effect of using Mbert instead of MT5.
3. Effect of using POS tags etc as another feature while training.