Neural Machine Translation

Saranya Pal
Department of Chemistry
210930
saranyap21@iitk.ac.in

Sourit Saha
Department of Mechanical Enginerring
200998
sourits20@iitk.ac.in

Tuesday 23rd April, 2024

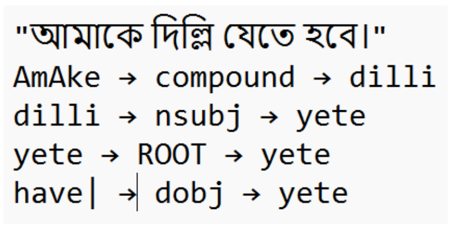
Abstract

The project addresses robust neural machine translation (NMT) for English and Indian languages, focusing on Seq2Seq and Transformer architectures. The dataset used is the Samanantar benchmark dataset available from `https://drive.google.com/drive/folders/1hR-8Mc7qQWsZAC-cw-nUqG8_OCqCdq-b` as translation datasets. The experimentation involves Transformers and optimizing hyperparameters. The training procedure involves Adam optimizer, variable learning rates, and careful epoch selection. The project emphasizes resource constraints, Google Colab and Kaggle, influencing model choices and training times.

1 Challenges Faced

- We attempted to utilize the Dependency Parse tree for Bangla to Hindi translation. Regrettably, our efforts were impeded as we did not receive a response to our email inquiry for the necessary code
- Consequently, we turned to Spacy for dependency parsing. Notably, we found that Spacy yielded similar dependency parsing errors as those anticipated from a Dependency Parse tree, aligning with our findings from the relevant research paper

- **Example 1:**



```
"আমাকে দিল্লি যেতে হবে।"
AmAke → compound → dilli
dilli → nsubj → yete
yete → ROOT → yete
have| →| dobj → yete
```

Figure 1: Dependence Relations of Ex. 1

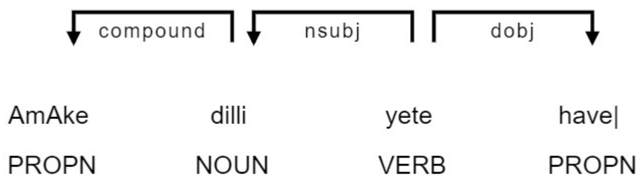
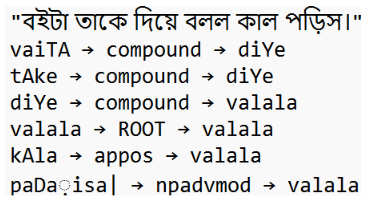


Figure 2: Dependency Parse Tree of Ex. 1

- **Example 2:**



```
"বইটা তাকে দিয়ে বলল কাল পড়িস।"
vaiTA → compound → diYe
tAke → compound → diYe
diYe → compound → valala
valala → ROOT → valala
kAla → appos → valala
paDa়isa। → npadvmod → valala
```

Figure 3: Dependence Relations of Ex. 2

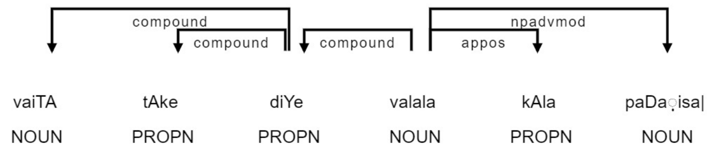


Figure 4: Dependency Parse Tree of Ex. 2

2 Introdcution and Problem Statemnet

The primary objective is to build robust NMT models that can seamlessly translate articles and editorials between Indian languages and English. The various caveats of the problem are:

- The most famous architecture used in NMT problems is the Seq2Seq Models. The encoder must learn a meaningful representation of the source language sentence, and the decoder should generate accurate translations word by word. The encoder and decoder architectures can be anything, ranging from traditional RNN-based models, sequence-to-sequence models with attention mechanisms, or transformer models. The choice should be based on the model's ability to capture contextual information and handle **long-range dependencies**
- The decoding strategy can include options such as greedy decoding, beam search, top-k sampling, etc. The effectiveness of the chosen decoding strategy will significantly impact the quality of translations.
- The model is trained from scratch. This also posed some challenges. The primary challenge was the computational constraints. Developing extensive models for translation tasks frequently necessitates the use of GPUs, thereby imposing limitations on expanding the size of our model.

3 Data Sources And Description

The dataset used is the Samanantar benchmark dataset available from `https://drive.google.com/drive/folders/1hR-8Mc7qQWsZAC-cw-nUqG8_OCqCdq-b` as translation datasets.

3.1 Corpus Statistics

The data used for the translations are taken from the `wat2021-devtest` and `wat2020-devtest`.

1. For Bengali-English translations, the following are done:
   - **Train Dataset**: From `benchamrks/wat2020-devtest/en-bn/`, the `dev.bn`, `test.bn` and from `benchmarks/wat2021-devtest/`, the `test.bn` and the respective `.en` files are concatenated to create the `train_df_bn_en.csv` and `train_df_en_bn.csv` datasets that are used to train the transformer.
   - **Test Dataset**: From `benchmarks/wat2021-devtest/`, the `dev.bn` and the respective `.en` files are used to create the `test_df_bn_en.csv` and `test_df_en_bn.csv` datasets that are used to test the model outputs. This is done as the Pre Trained model results are already available from `Assignment 3`.
2. For Bengali-Hindi translations,
   - Mltilingual data was not available except `benchmarks/wat2021-devtest/`. Only using either `dev.bn` or `test.bn` was not possible as the number of sentences are not sufficient to train transformer (only 1000 and 2390 respectively).
   - So we concatenated the `test.bn` and `dev.bn` and the corresponding `.hi` files to create the `train_df_bn_hi.csv` and `train_df_hi_bn.csv` datasets.

- The test dataset was created similarly to the English-Bengali translation.

The following table shows the statistics of the corpus:

Table 1: Dataset Statistics

Dataset	**Number of Sentences**	**Source Language**	**Avg. Length of sentence**	**Target Language**	**Avg. Length of sentence**
train_df_bn_en	9044	Bengali	14.465	English	18.512
test_df_bn_en	1000	Bengali	13.935	English	16.322
train_df_bn_hi	3390	Bengali	13.777	Hindi	17.687
test_df_bn_hi	1000	Bengali	13.935	Hindi	17.739

3.2 Vocabulary Statistics

The vocabulary statistics across the languages helps to understand and decide the sizes of the vocabulary that needs to be considered for the translation pipeline. The train and test datasets are considered. The statistics are shown in Table below:

Table 2: Vocabulary Lengths and Limitations by Language Pair

Translation	**Language**	**Limitation**	**Vocab. Length**
Bengali-English	English	None	8435
		MF = 2	5165
English-Bengali	Bengali	None	11588
		MF = 2	5472
Bengali-Hindi	Hindi	None	3645
		MF = 2	2498
Hindi-Bengali	Bengali	None	3976
		MF = 2	2609

4 Model Description

Transformers were used with changes in the inference algorithm.

4.1 Inference

We have only used greedy search and top k-sampling. In the initial stages, we thought that not using top k-sampling was the reason behind not getting good results (mostly getting < UNK > outputs). But, top k-sampling performed severely worse, because it was outputting < EOS > tags with a higher probability (temperature = 1.5, k = 5). All models henceforth were trained and inferred using greedy search only. Notably, the inference codes are not well optimized, so, they might take longer than expected to complete inference.

4.2 Loss Function

The loss function used in all the models was Cross Entropy Loss.

4.3 Final Model

The final model used was a transformer. The model is the standard transformer as described in [1]. The implementation is inspired by [2]. The first layer is the Embedding layer, which will convert the list of input indices into corresponding input embeddings. Then positional encodings are also added. Using that the model is run, and the model returns probabilities over each word, which are then decoded.

5 Model Training

5.1 Data Preprocessing

1. The `.csv` files mentioned in section 3 is imported onto Google Colab.
2. The English text is also converted to lowercase. This is because none of the Indian languages have an idea of case, so it is immaterial to the translator.
3. Using custom-built functions and classes built upon PyTorch utilities DataLoader and Dataset, the csv files are converted into datasets, which are then loaded as an iterator. These iterators are fed into the model and the corresponding outputs are taken.

   i **Vocabulary Class**: The Vocabulary class is designed for managing the vocabulary of a natural language processing (NLP) model. It is particularly useful for handling tokenization, building vocabulary dictionaries, and converting text data into numericalized representations. The tokenization is done by custom functions built upon `spaCy` and `indic-nlp`.

   ii **Train_Dataset Class**: The Train Dataset class is designed to serve as a dataset for training a natural language processing (NLP) model. It extends the functionality of PyTorch's Dataset class and is equipped to handle source and target text data, build vocabularies, and perform text numericalization. A similar class is made for the Validation/Test Datasets.

   iii **get_train_loader function**: The get train loader function facilitates the creation of a PyTorch DataLoader for training a neural network model. The shuffling and padding with zeros is also carried out in this function. Similar functions are made for Validation/ Test Datasets.

   iv **master_loader function**: The master loader function serves as a higher-level interface for creating training and validation loaders, along with other essential components for a sequence-to-sequence neural network model. Only this function is called in the main notebook, thus serving as a packaged and short function.

5.2 Training Procedure

- The optimizer in all the trained models was Adam. It would have been interesting to try out other optimizers
- The major problem in learning rates popped up in the Transformer training, when there was a overshoot noticed for these values of learning Rate. Hence, it was set to 0.0001 to 0.0003 in the transformer.
- **Epochs:** We tried with epoch variations of 25, from 25 to 150. Higher the epoch along with batch size, better the results.
- **Training Time:** Below mentions the total training time for the entire code. This direclty reflects on the training time of the models used.
  - For transformers, training times were roughly around 10 seconds for English-Bengali translations and around 3 seconds for Hindi-Bengali translations for each epoch.

5.3 Hyperparameters Tuning

The following hyperparameters were tried for the Transformer models:-

- `EMB SIZE` = 64, 128
- `NHEAD` = 8
- `FFN HID DIM` = 64, 128
- `BATCH SIZE` = 64, 128, 256
- `NUM ENCODER LAYERS` = 3, 4
- `NUM DECODER LAYERS` = 3, 4
- `NUM EPOCHS` = 50, 75, 100, 150

The best set of hyperparameters, that gave the best score was, using a Transformer model was using batch size 256, embedding dimensions 128, hidden dimension 128, number of layers 4 and number of epochs set to 150.

6 Model Evaluation

6.1 Results

The results of the submissions made are listed below. For some unknown reason, Bengali-English translations always performs better than English-Bengali translations; this will be observed in the results of pre trained models as well. The reason of very high scores for Bengali-Hindi and vice versa translations is that the model was already trained on the test cases as discussed in section 3.1.

Table 3: Translation Performance Metrics

Translation	BLEU score	ROUGE score
Bengali-English	0.03765	0.14280
English-Bengali	0.02392	0.05867
Bengali-Hindi	0.16174	0.36209
Hindi-Bengali	0.12761	0.32345

6.2 Comparision with Pre Trained models

We have compared the scores of our transformer model with scores of existing pre trained models as done in Assignment 3. The below table shows the comparisons:

Table 4: Performance Metrics for Translation Models

Translation	BLEU/ROUGE	Transformer	Nllb-200	IndicTrans
Bengali-English	BLEU	0.03765	0.17807	0.22722
	ROUGE	0.14280	0.15486	0.49672
English-Bengali	BLEU	0.02392	0.05635	0.07784
	ROUGE	0.05867	0.06261	0.30413
Bengali-Hindi	BLEU	0.16174	0.17855	0.16733
	ROUGE	0.36209	0.43114	0.42765
Hindi-Bengali	BLEU	0.12761	0.05862	0.06457
	ROUGE	0.32345	0.26968	0.27764

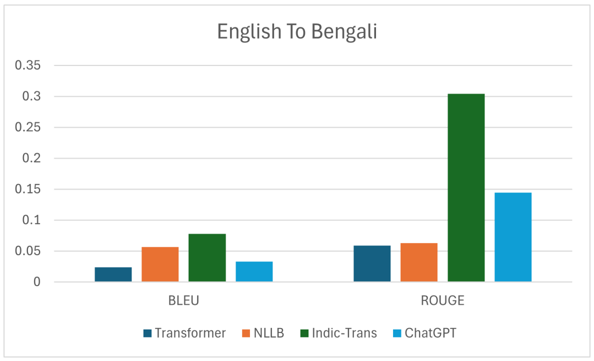


Figure 5: English To Bengali

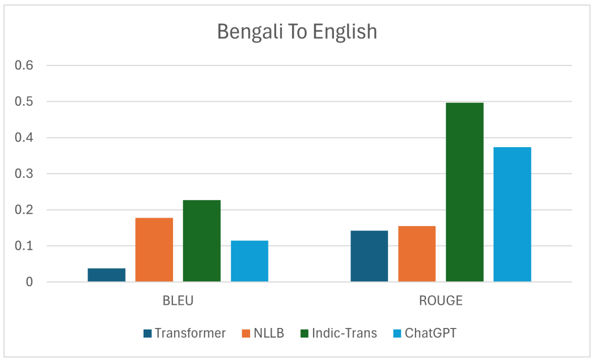


Figure 6: Bengali To English

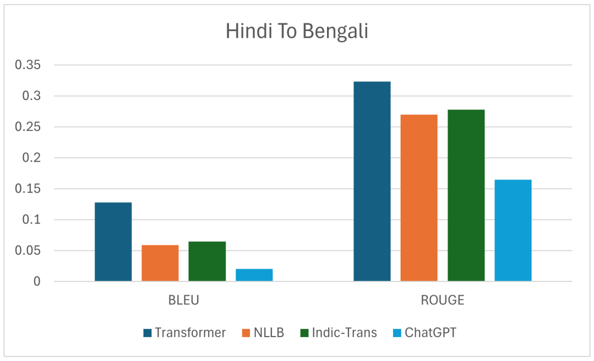


Figure 7: Hindi To Bengali

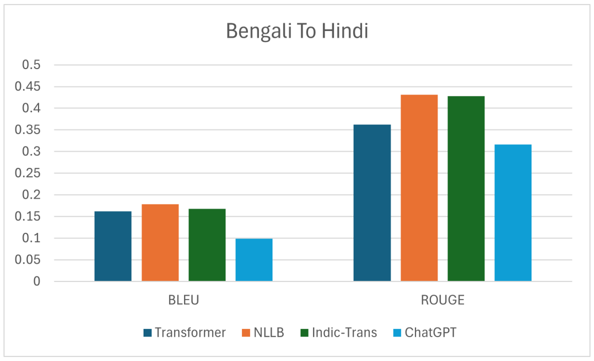


Figure 8: Bengali To Hindi

Note: ChatGPT translations are on 50 sentences only.

7 Error Analysis

1. The models did not come out perfectly mostly due to resource constraints. The datasets used for translations are not sufficient enough to generate perfect models.
2. It was impossible to train large models given Google Colab's limits. Creating larger vocabularies and including all words in all lists for the English and other language vocabulary would have helped.

8 Conclusion

The report discusses different models that have been experimented with in the Transformer architecture. Hyperparameter tuning, training procedures, and results are presented, with a notable emphasis on resource constraints due to the use of platforms like Google Colab. The results showcase the performance of different models in terms of BLEU and ROGUE scores. The Transformer architecture stands out with the decent scores as compared to other pre-trained models, demonstrating its effectiveness in machine translation tasks.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[2] P. Tutorials, "Sequence-to-sequence modeling with nn.transformer and torchtext." Available: `https://pytorch.org/tutorials/beginner/translation_transformer.html`, 2023.

[3] A. Persson, "Aladdin Persson's YouTube channel," 2023.