Group 8 : Language Translation Service

Bhushan Mahajan
bhushanm@buffalo.edu
UBID: bhushanm

Yukti Kholiwal
yuktikho@buffalo.edu
UBID: yuktikho

Deep Shahane
deepniti@buffalo.edu
UBID: deepniti

Abstract*— Language translation is an important task that bridges language barriers in our increasingly global society. In this project, we approached the challenge of translating texts between languages while preserving their original meanings and contexts. We utilized a different approach: first, by employing advanced pre-trained models which are Google T5 and M2M-100 and those models are trained and fine tuned on our dataset, which are renowned for their language translation capabilities; and second, by developing our own custom Transformer model with different positional encodings. We enhanced our model's effectiveness through the use of optimization techniques and loss functions. Our results, measured by the METEOR score on the test dataset, allowed us to assess the performance of both the pre-trained models and our custom-built solution. We compared our models scores with pretrained models scores and as well the translated text to check the context. This project not only tested these advanced tools but also provided some insights into the practical aspects of machine translation. We hosted the translation service website using gradio.

Keywords — Language translation, Sequence-to-sequence models, Custom-built Transformer, Pytorch, Optimization techniques, Loss functions, METEOR score

I. Dataset

A. Description of the Dataset:

For our project, we primarily used the Europarl Parallel Corpus. This dataset, originating from the proceedings of the European Parliament, from April 1996 to November 2011 and is saved in format of dataframe as a parallel corpus of english and french sentences. We only focused on the French-English language pair, which featured over 1.8 million rows and totaled 599 MB. We saved the dataset in a hugging face which was easy to access whenever required. This bilingual resource provides a well-aligned sentence-level text, ideal for in-depth linguistic analysis and machine translation tasks. But the dataset is merely focusing on parliament discussions so it might perform better when the input is related to parliament.

B. Data Engineering Techniques Used:

The process for managing and preparing the dataset for machine translation training included several crucial steps:

i. *Data Acquisition:* We downloaded the datasets as zip files from their online repositories, with the Europarl dataset taken from the official Europarl corpus website for French to English texts.

ii. *Reading and Formatting Data:* After downloading the dataset, we used custom Python scripts to read the text files. We designed a function named file_to_lines that reads lines from both the text files(French and English) and transforms them into rows of a dataframe, ensuring the data is orderly and readily accessible for training.

iii. *Tokenization:* We employed SpaCy tokenizers, which use language-specific models for English and French, to tokenize the sentences. This process also involved creating vocabularies from the training dataset that include tokens and special symbols such as <unk> for unknown tokens, <bos> for the beginning of a sentence, <eos> for the end of a sentence, and <pad> for padding to ensure uniformity while training.

iv. *Numerical Encoding and Padding:* The tokens were converted into numerical indices to make them suitable for model input. Padding was applied to the sequences in the dataframe to ensure uniformity in input sizes

v. *Data Splitting and DataLoader Preparation:* We divided the dataset into training (80%), validation (10%), and testing (10%) subsets. PyTorch DataLoaders were then configured with

batch size 32 to allow efficient iteration over these subsets during the training, validation, and testing phases.

II. Model description

A. Model Architecture:

The architecture of our language translation service is primarily built with Transformer models. We also employed pre-trained models like Google T5 and Facebook M2M-100 on our dataset. The Google T5 model functions on a general NLP framework, ideal for a variety of text-to-text tasks including translation, while Facebook M2M-100 is tailored for multilingual machine translation, supporting translation across 100 languages without the need for English language as an intermediary.

Additionally, we developed our own custom Transformer model using PyTorch. This model is designed from scratch using pytorch module to allow us some control over architectural elements such as positional encoding both absolute and relative and the use of different optimizers (Adam, AdamW, SGD), which are essential for enhancing efficiency of the model's performance based on our specific dataset and project goals. Below are the Transformer model diagrams illustrating the two positional encoding techniques used:

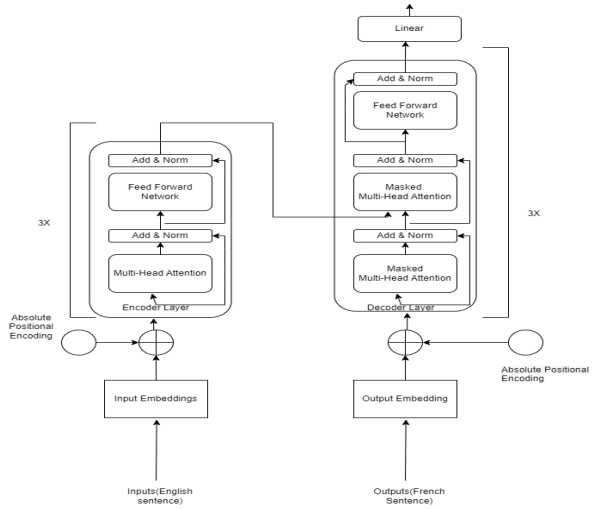


Fig 1: This diagram illustrates a Transformer model using absolute positional encoding, showing the flow from input embeddings of English sentences through multiple encoder and decoder layers, culminating in the output embeddings for French sentences.

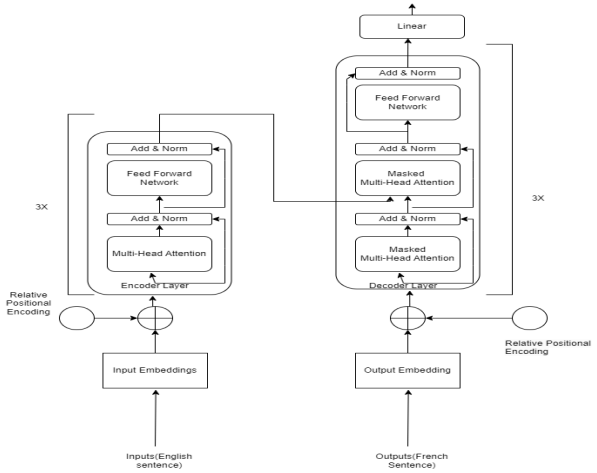


Fig 2: This diagram depicts a Transformer model utilizing relative positional encoding, detailing the encoder and decoder layers that process input embeddings from English to output embeddings in French, emphasizing the position-relative processing of language elements.

Relative Positional Encoding:

Relative Positional Encoding is a method used in Transformer architectures to address the relationships between positions of words in the sequence. It differs from absolute positional encoding, which assigns a unique encoding to each absolute position in the sequence; relative positional encoding focuses on the distances between positions of words. This can be particularly beneficial in tasks where the relative positioning of elements in a sequence is more significant than their absolute positioning. It can understand the context using relative positional encoding.

In our code, the Python class RelativePositionalEncoding implements this concept of relative positioning by generating positional encodings based on relative positions. It defines an embedding size and a maximum sequence length. The method also calculates positional encodings for a range of relative positions from -max_len to +max_len. It uses sinusoidal functions, specifically sine for even indices and cosine for odd indices of the embedding dimensions. This sinusoidal approach is based on the hypothesis that it will allow the model to easily learn to attend by relative positions, since for any

fixed offset k, $\sin(\theta+k)$ and $\cos(\theta+k)$ can be represented as linear transformations of $\sin(\theta)$ and $\cos(\theta)$.

Absolute Positional Encoding:

Absolute Positional Encoding assigns a unique encoding to each position in the sequence, which helps the model to incorporate information about the order of tokens in the input data. This technique is commonly used in many Transformer-based models to enhance their ability to understand sequence order, an aspect not naturally captured by the standard attention mechanisms in Transformers.

In the code we have one PositionalEncoding class, which is absolute positional encodings and those are generated using a similar sinusoidal pattern as the relative version, but they are directly tied to the absolute positions of the words. It computes a positional encoding for each position up to a maximum length of sequence. The sinusoidal functions are applied, where positions are multiplied by a decay factor computed from the embedding dimensionality, helping to vary the wavelength of the sinusoidal wave across dimensions.

The resultant positional encodings are then added to the token embeddings in the Transformer. The use of dropout in the positional encoding layer helps prevent overfitting by randomly zeroing vector elements during training.

B. List of Models Tried and Reason for the Best Choice:

i. *Google T5:* The Google T5 model is utilized for its versatility and efficiency in managing complex sequence-to-sequence transformations, making it ideal for tasks requiring an in-depth comprehension of text and the generation of human-like text. We optimized its configuration for our training with tailored hyperparameters: a learning rate of 2e-5, batch sizes of 4 for both training and evaluation per device, a two-step gradient accumulation, and a total of 10 training epochs. Additionally, we applied a weight decay of 0.01 for regularization, and conducted evaluations and saves at specific steps to ensure a balance between monitoring performance and optimizing resource use.

ii. *Facebook M2M-100:* The Facebook M2M-100 model was selected for its strong capability in direct multilingual translation, crucial for our project's goal to translate between French and English without English as an intermediary. We fine-tuned this model with particular hyperparameters to enhance its effectiveness: a learning rate of 3e-5, training and evaluation batch sizes of 4, a gradient accumulation strategy over two steps, and a training duration of 10 epochs with a weight decay of 0.01 to improve generalization. Evaluations and model saves were systematically carried out every 1000 steps, as recorded in our logging directory, to maintain an optimal balance between performance evaluation and computational efficiency.

iii. *Custom Transformer:* Additionally, we developed a custom Transformer model to investigate how different positional encoding(relative or absolute) impacts translation quality. Each variant of this model underwent training over 10 epochs with a batch size of 32. Specifically, for the custom transformer used in the final two selected models(Best from 18 models we tested), training was executed on 1 million rows(data) to thoroughly evaluate and refine the system for optimal performance of translation task.

C. Description and Understanding of the Algorithm:

The Transformer model architecture, which underlies both Google T5 and our custom model, operates on a mechanism of attention, specifically the self-attention mechanism, that allows the model to weigh the importance of different words in a sentence, irrespective of their positional distance from the target word. This is crucial for understanding context and meaning in translation tasks.

The model is split into two main components:

i. *Encoder:* Processes the input text and represents it as a set of vectors containing the contextual information of each word relative to all other words in the sentence.

ii. *Decoder:* Generates the translated output incrementally, focusing on different parts of the input sequence as guided by the attention mechanism, and leveraging the encoded vectors.

The effective use of both absolute and relative positional encodings in our custom Transformer allows

the model to not only capture the sequence of the words but also understand the context based on their positions relative to each other. This dual approach in positional encoding provides a more nuanced comprehension of the input sequence, which is particularly beneficial in languages with flexible syntactic structures. By combining the robust pre-trained models with a customized approach, our project ensures an advanced and nuanced handling of language translation, setting the stage for effective real-world application in diverse linguistic scenarios.

III. Loss functions

A. Loss Functions Implemented:

i. *Cross Entropy Loss:* The primary loss function used for our model is cross entropy loss, which measures the difference between the actual labels and the predicted probabilities by the model, which are crucial for classification tasks. We have implemented various optimizers: Adam, AdamW, and SGD, each with the same learning rates of 0.001. We have ignored the padding while calculating the loss in each loss function.

ii. *Cross Entropy with Label Smoothing:* Cross Entropy with Label Smoothing adjusts target probabilities by reducing penalties for incorrect predictions and preventing overconfidence to make the model more robust. This modification increases the model's capability to handle the noisy input data. Even for this loss function, We implemented this technique across various optimizers like Adam, AdamW, and SGD, maintaining consistent learning rates. This approach ensures the model that benefits from reduced prediction confidence and enhanced robustness, improving the models performance.

iii. *Negative Log Likelihood (NLL) Loss:* Negative Log Likelihood (NLL) Loss is employed with log-softmax activation to evaluate model performance effectively. It is similar to cross entropy but differs in mathematics. This loss function focuses on the probability of getting correct classes, improving training by directing attention to key outputs. NLL Loss is also used with optimizers like Adam, AdamW, and SGD. Our settings include a consistent learning rate of 0.001, enhancing the model's precision and focus during training.

B. Chosen Loss Function:

The best model for relative positional encoding is with the combination Cross Entropy Loss and Adam optimizer, where we got a test loss of 2.8791 and the greatest METEOR score of 0.109. This suggests a good balance between generalization to new data and translation accuracy.

The best results for absolute positional encoding were obtained with the AdamW optimizer and Cross Entropy Loss with Label Smoothing loss. This resulted in a test loss of 7.6610 and a METEOR score of 0.0844. Reducing the model's confidence can be critical in some cases for avoiding overfitting of the model, particularly when dealing with difficult language translation assignments where precise output matching is less practical, and here we are more interested in context. This is where the label smoothing component comes in.

C. Innovation on the Loss Function:

Here, we explored the dynamic adaptation of label smoothing parameters based on validation performance, particularly with help of using the AdamW optimizer. This represents a different way of approaching a problem.

IV. Optimization algorithms

A. Optimizers Implemented:

i. *Adam Optimizer:* ToAdam is used to improve the training process which implements an adaptive optimization technique. Based on the gradients it computes it dynamically adjusts the parameters, with a constant learning rate of 0.001. It does it with the help of following strategies such as controlling the accumulation of gradients and applying gradient clipping. Additionally, it effectively handles relevant data by ignoring padding indices when calculating Cross Entropy Loss. These features make it exceptionally suitable for complex sequence-to-sequence translation tasks, allowing for precise learning of intricate data relationships.

ii. *AdamW Optimizer:* In this optimizer for complex model training it includes an enhancement that

optimizes parameter growth. This enhancement comes in the form of weight decay, which helps prevent overfitting. It applies a structured approach to initialize the weight and continues with updates for a loop over a span of 10 epochs for model generalization. It begins with a learning rate of 0.001. The efficiency of this optimizer is underscored by its precise management of different loss functions and positional encodings, which play a crucial role in maintaining the resilience of the model.

iii. *SGD (Stochastic Gradient Descent):* Implementing SGD with a learning rate of 0.001. This optimizer's implementation across ten epochs involves an examination of different positional encodings and loss functions. SGD facilitates the steady improvement of the model by updating its parameters in a controlled fashion. This approach establishes a reliable baseline when comparing it to more advanced optimizers, underscoring its utility in supporting consistent model development.

B. *Chosen Optimization Algorithm:*

Since the Adam optimizer can handle sparse gradients well and is flexible enough to work with a wide range of datasets, it was selected as the best model with relative positional encoding. Notable for explicitly adding weight decay, AdamW worked best for the optimal model utilizing absolute positional encoding, improving training stability and performance over the long run.

C. *Innovations on the Optimization Algorithm:*

The dynamic adaptation of label smoothing parameters, especially when using the AdamW optimizer, marks a significant improvement, as it allows the model to modify its learning approach based on immediate feedback from its validation performance. This innovation is crucial for continually refining the model's accuracy and responsiveness to changing data patterns.

V. Metrics and Experimental Results

In our project, we aimed to develop a robust machine translation system. We evaluated the performance of our models with the help of different metrics, like COMET, BLeu with a focus on the METEOR score for its alignment with human judgment quality assessments in translation tasks.

A. *Predicted Results Overview:*

We experimented with multiple optimizers and loss functions, implementing a range of models and setups and using both absolute and relative positional encodings. The models were pre-processed to meet our training requirements and trained on a variety of datasets obtained from Europarl. After a thorough training and assessment process, we determined which models were the best at each positional encoding strategy:

Relative Positional Encoding: With a test loss of 2.8791 and a METEOR score of 0.1093, the top model employed the Adam optimizer with Cross-Entropy Loss.

Absolute Positional Encoding: With a test loss of 7.6610 and a METEOR score of 0.0844, the AdamW optimizer employing Cross-Entropy Loss with Label Smoothing produced the best model.

B. *Detailed Results from Experiments:*

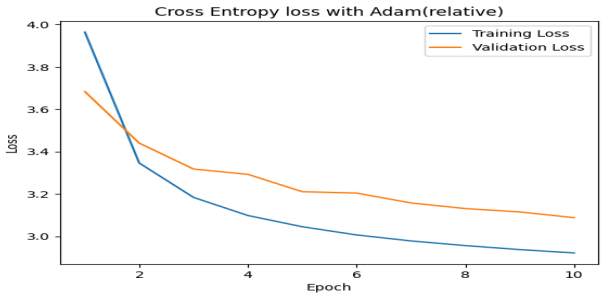


Fig 3: Graph showing training and validation loss for a model using relative positional encoding, with losses decreasing significantly over 10 epochs, indicating improved model accuracy

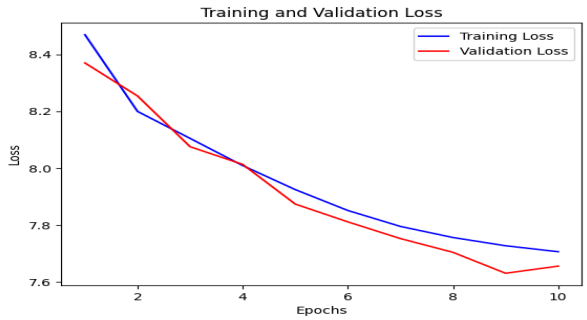


Fig 4 : Loss trends for training and validation for a model using absolute positional encoding, illustrating consistent convergence over 10 epochs, highlighting effective learning and generalization capabilities.

i. Relative Positional Encoding Results:

Optimizer	Loss Function	Training Loss	Validation Loss	Test Loss	Meteor Score
Adam	**CEL**	**3.24**	**3.47**	**2.87**	**0.1092**
	CELWLS	7.72	7.71	4.57	0.1034
	NLLLoss	3.23	3.45	3.02	0.0722
Adamw	CEL	3.24	3.49	10.65	0.0014
	CELWLS	7.71	7.71	7.63	0.0769
	NLLLoss	3.28	3.70	3.31	0.083
SGD	CEL	6.69	6.66	6.65	0.0529
	CELWLS	7.24	7.20	7.20	0.0448
	NLLLoss	6.68	6.64	6.65	0.0323

ii. Absolute Positional Encoding Results:

Optimizer	Loss Function	Training Loss	Validation Loss	Test Loss	Meteor Score
Adam	CEL	3.38	3.68	3.32	0.0818
	CELWLS	7.78	7.81	4.82	0.0758
	NLLLoss	3.40	3.84	3.49	0.0821
Adamw	CEL	3.36	3.52	3.14	0.0833
	CELWLS	**7.75**	**7.72**	**7.66**	**0.0843**
	NLLLoss	3.34	3.83	3.47	0.1059
SGD	CEL	6.93	6.88	6.87	0.0504
	CELWLS	7.40	7.36	7.36	0.0475
	NLLLoss	7.01	6.94	6.94	0.0680

In the Tables above: CEL is Cross-Entropy Loss, CELWLS is Cross-Entropy Loss with Label Smoothing and NLLLoss is Negative Log Likelihood Loss

C. Metrics Used for Evaluation:

METEOR Score: Our primary metric, chosen for its ability to assess translation quality not only based on exact word matches but also on synonymy and sentence structure.

Test Loss: Used to measure how well the model predicts the actual translation outputs during the testing phase, providing a direct indicator of model performance under non-training conditions.

D. Understanding METEOR Score:

METEOR (Metric for Evaluation of Translation with Explicit Ordering) is an advanced evaluation metric used for assessing the quality of machine translations. Unlike BLEU, which primarily relies on exact match n-grams between the machine output and reference translation, METEOR goes further by incorporating synonyms and considering the sentence structure. It calculates scores based on the harmonic mean of unigram precision and recall, with a higher weighting on recall and adjustments for sentence structure and synonyms. This makes METEOR particularly valuable in scenarios where nuanced translation quality is critical, as it aligns more closely with human judgment than other metrics.

E. Why Choose METEOR?

Higher Correlation with Human Judgment: METEOR scores have been shown to correlate more strongly with human assessments compared to other metrics.

Adaptability: It accounts for synonymy and varied phrasings, essential for translating languages with rich linguistic structures and vocabularies.

F. *Comparison of Experiments*

Our experiments spanned various configurations:

Before scaling to larger datasets (20,000 and eventually 100,000 rows) to fine-tune and validate our models, we first experimented with smaller subsets of data (5,000 rows) to change specific transformer model parameters.

Various decoding algorithms were examined, such as Beam Search and Greedy Decoding. By taking into account various candidate sequences, Beam Search produced better results in terms of retaining coherent long-form translations.

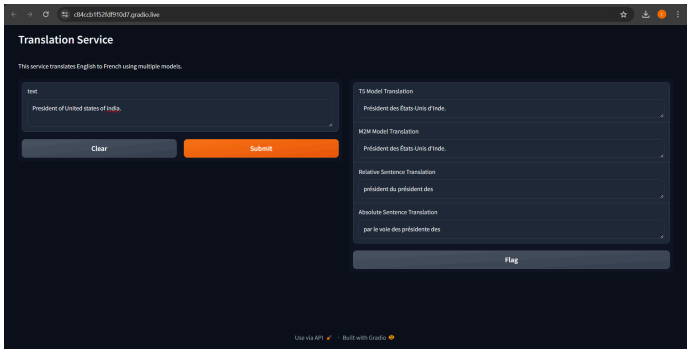
The models were hosted on Gradio, allowing real-time interaction and demonstration of their translation capabilities to stakeholders and potential users.

To further contextualize our results, we compared the performance of our best absolute and relative positional encoding models against two pre-trained models that we chose: M2M_100 and Google T5. The comparison was based on their METEOR scores:

Models	**Meteor Score**
Google T5	0.5654
M2M_100	0.0512
Best Absolute Model	0.0844
Best Relational Model	0.1093

The results highlight that our relative positional encoding model with Adam and Cross-Entropy Loss surpasses the performance of the advanced M2M_100 model and shows a competitive edge close to the Google T5 in terms of translation quality, particularly emphasizing the effectiveness of our approach in handling complex translation tasks.

Implementation:



We hosted our project using gradio. After passing an input in a box the user will get 4 outputs that are from google t5, m2m_100, best absolute model, best relational model. We uploaded Google T5 and M2M_100 on hugging face and calling it from there for better accessibility.

VI. Contributions and Github

Contributions Percentage:
Bhushan: 33.33%
Deep: 33.33%
Yukti: 33.33%

Github link:
https://github.com/Bhushan4829/deep_learning_project

VII. References

1. https://pytorch.org/
2. https://github.com/google-research/text-to-text-transfer-transformer
3. https://huggingface.co/docs/transformers/en/model_doc/m2m_100
4. https://stackoverflow.com/
5. https://www.analyticsvidhya.com/
6. https://medium.com/
7. https://huggingface.co/spaces/evaluate-metric/meteor
8. https://arxiv.org/abs/1706.03762