GROUP 16 : Language Translation Service

Vaishnavi Koyyada
50468340
vkoyyada@buffalo.edu

Thilak Reddy Dharam
50469154
thilakre@buffalo.edu

Teja Naveen Chintapalli
50468972
tejanave@buffalo.edu

Abstract

In the era of globalization, the ability to break language barriers is more crucial than ever. This report presents the development of a state-of-the-art neural machine translation service, designed to translate text between Romanian to English and French to English. Leveraging the rich linguistic resources and the comprehensive proceedings of the Europarl corpus Datasets, this service employs advanced natural language processing techniques. The project delineates the deployment of sophisticated sequence-to-sequence models, including RNNs with GRUs, attention mechanisms, and cutting-edge transformer models. This detailed exploration not only charts the methodical data collection and preprocessing efforts but also the iterative optimization and fine-tuning of these models. The culmination of this work is a robust translation service, the efficacy of which is meticulously evaluated and documented, showcasing the profound skills acquired in Python programming and sequence-to-sequence model architecture throughout this endeavor.

1 Dataset

1.1 Description of the Dataset

The Europarl Dataset, derived from the proceedings of the European Parliament, stands as a cornerstone in the realm of computational linguistics and machine translation. It is particularly integral to projects that aim to navigate the complexities of language translation, such as our service focusing on Romanian to English and French to English translations. This dataset encompasses an array of 21 European languages, prominently featuring Romanic languages like French and Romanian, and the ubiquitous languages French, English. The dataset's unique selling point is its meticulously sentence-aligned text corpus, an essential attribute for the development of high-fidelity translation models.

• The French-English Parallel Corpus features 2,007,723 sentences, with the French component containing 51,388,643 words and the English counterpart comprising 50,196,035 words.
• The Romanian-English Parallel Corpus includes 399,375 sentences, with 9,628,010 Romanian words and 9,710,331 English words.

1.2 Data Engineering

The data engineering process for the Europarl Language Translation project is an intricate and multi-staged procedure designed to prepare the Romanian-English and French-English datasets for effective machine translation. At the initial stage, we have read the raw text files and have opened and read the contents of the file in UTF-8 encoding. Then we processed the sentences in these files by following the steps below:

• Each sentence is split and stripped of leading and trailing spaces.
• Non-ASCII characters are normalized and removed, leveraging the normalize function from the Unicode data module.
• All words are converted to lowercase, and punctuation is removed to ensure uniformity.
• Non-alphabetic characters are filtered out to retain only words.
• The processed sentences are then compiled into a list.

These preprocessed sentences of both languages are then saved into separate pickle files. Then we load the preprocessed data from these pickle files by creating a data frame with these sentences and then a combined vocabulary list is created by zipping both the language vocabularies together. We then map each word in the sentences to a unique index and vice versa. This mapping is essential for converting words to numerical representations suitable for model input. Specific prefixes in English sentences are identified to ensure that the dataset includes a diverse range of starting phrases, enhancing the model's ability to handle different sentence structures. Then, the sentences are selected and paired based on certain criteria, such as length and starting prefixes.

We then process these filtered set of sentence pairs. Sentences are then converted into tensors. This involves converting each word to its corresponding index and padding the sentences to a uniform length. These tensors are then used to create custom datasets for training and validation. Data Loader objects are created for both training and validation datasets. These are used to feed data into the neural network in batches during the training and validation phases.

2 Model Description

In the development of our project, we have undertaken a thorough examination of four distinct models. Our objective was to assess and compare these models to identify the most effective solution for our project's needs. The models we have evaluated include:

2.1 Encoder - Decoder Model

The Encoder-Decoder Model has two main Components, the Encoder and the Decoder in Figure 1. The Encoder's primary function is to transform the input sequence into a context vector. It features a hidden layer in the Gated Recurrent Unit (GRU) and additionally, it incorporates a dropout layer characterized by a dropout probability.

The Decoder complements the Encoder. Its primary role is to convert the context vector from the Encoder into the output sequence. It includes an Embedding Layer that transforms each target sequence token into vectors, a GRU Layer which works same as that of Encoder's and a Linear Layer that aligns GRU's output with the output vocabulary's size, and a Dropout Layer like the Encoder's. During its forward pass, the Decoder begins with the Encoder's outputs and state, initiating with a unique start token. It then generates output tokens step-by-step, by updating its state. The model can also employ its 'teacher forcing' by using the actual target tokens as inputs during training or can even rely on its own predictions.

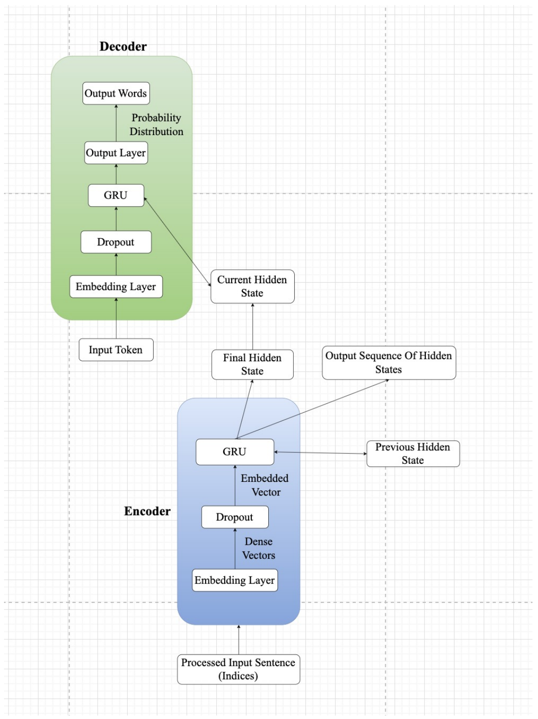


Figure 1: Encoder Decoder Model

Finally, it applies a SoftMax function to produce a probability distribution over the output vocabulary. The Training and Validation Losses of this model with and without fine tuning is shown in Figures 2 and 3.

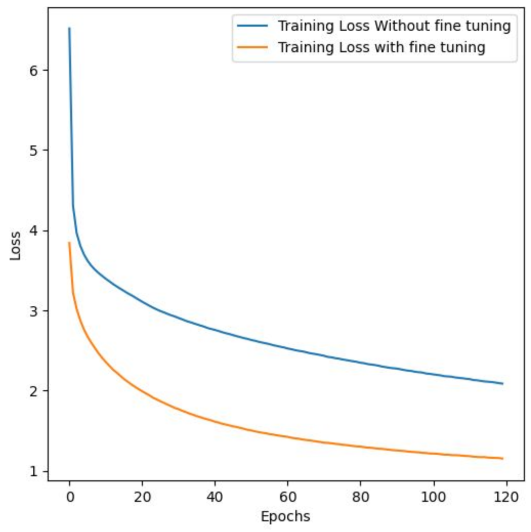


Figure 2: Training Loss Encoder-Decoder Model

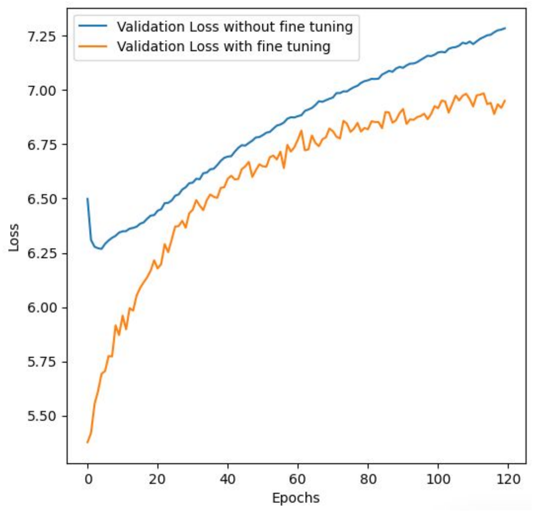


Figure 3: Validation Loss Encoder-Decoder Model

2.2 Encoder - Decoder with Multi Head Attention Mechanism

The Encoder Decoder with Attention Mechanism represents an advancement over the before model, by integrating an attention module to enhance its processing capabilities. The architecture of the model is as depicted in Figure 4. This model comprises three core components: Sequence Encoder, Multi Head Attention, and Decoder with Attention. The Sequence Encoder starts with an embedding layer that transforms input tokens into vectors of uniform size, thereby standardizing the input representation. Then we have a Gated Recurrent Unit (GRU) layer which processes these embeddings in sequence and a dropout layer that improves the model's ability to generalize.

The Multi Head Attention mechanism works by splitting the attention process across multiple 'heads', enabling the model to attend to different segments of the input sequence in parallel. Each head in the Multi-Head Attention layer linearly projects the queries, keys, and values, facilitating a comprehensive understanding of the input context. The attention scores generated by each head are scaled, normalized, and then aggregated, resulting in context vector that encapsulates information from the input sequence.

This decoder initiates the process by preparing the input, which is then combined with the context vector from the Multi-Head Attention. A GRU layer within the decoder processes this input. The decoder operates iteratively by generating the output sequence that is one token at a time. At each step, the model employs a strategy where it chooses between using the actual next input token or the predicted token from the previous step with the teacher forcing ratio. The output from the GRU is subsequently passed through a linear layer, aligning it with the size of the output vocabulary. Finally, the SoftMax layer is applied to produce a probability distribution over the output tokens, determining the most probable subsequent token in the sequence. The Training and Validation Losses of this model are shown in Figures 5 and 6.

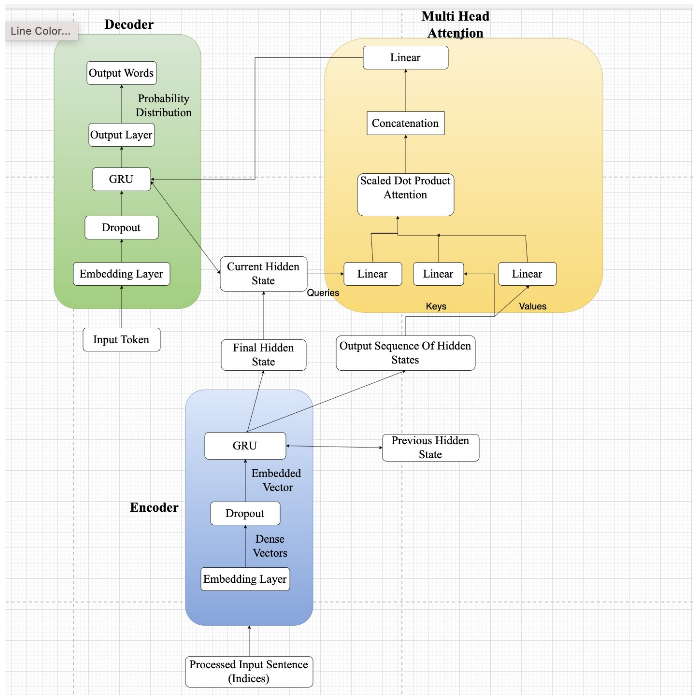


Figure 4: Encoder-Decoder with MultiHead Attention Mechanism Model Architecture

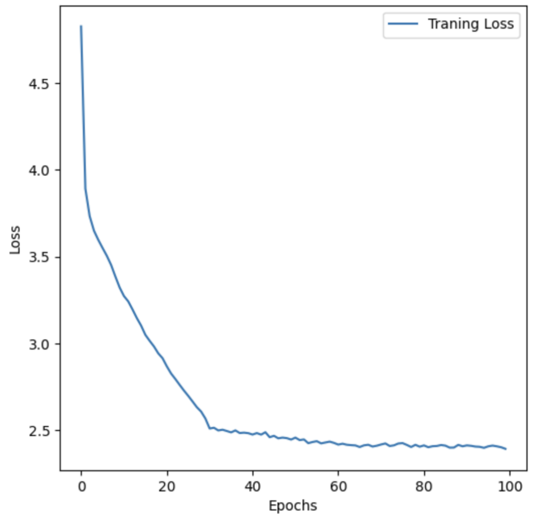


Figure 5: Training Loss Encoder-Decoder with Multi Head Attention Mechanism Model

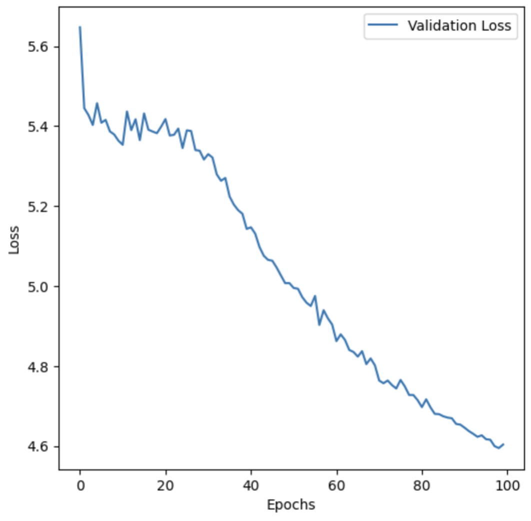


Figure 6: Validation Loss Encoder-Decoder with Multi Head Attention Mechanism Model

2.3 T5 Model

T5, or the Text-to-Text Transfer Transformer, is a versatile and powerful model developed by Google Research. Its foundational architecture is based on the Transformer model but with a unique approach to handling NLP tasks. As we can see for Figure 7, T5 has encoder and decoder blocks. Each block in the encoder and decoder is made up of layers that include self-attention mechanisms and feed-forward neural networks. It also includes positional encodings to maintain the order of words in the input text. The model also features layer normalization and residual connections in each block, which are integral for stabilizing the training process and aiding in the flow of gradients during backpropagation.

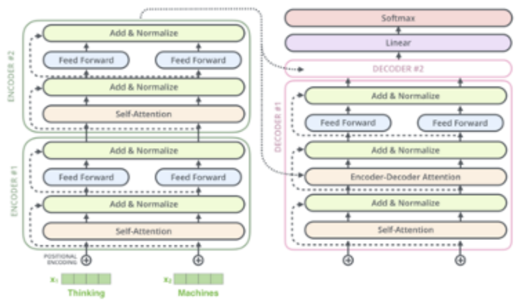


Figure 7: T5 Model Architecture

T5 standardizes all NLP tasks into a uniform text-to-text format. This model simplifies the NLP pipeline. During pre-training, T5 engages in tasks like filling in missing words, which helps it grasp the nuances of language. Its ability to generalize across various tasks without needing specific architecture modifications for each task is a standout feature. T5's text-to-text approach, combined with its scalable architecture and exemplary performance. The Training and Validation Losses of this model are shown in Figures 8 and 9.

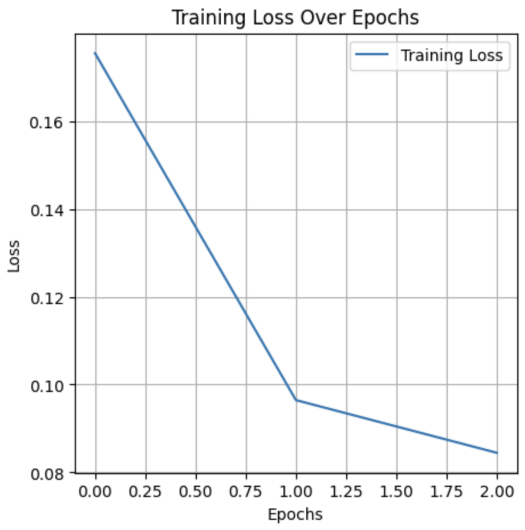


Figure 8: Training Loss for T5 Model

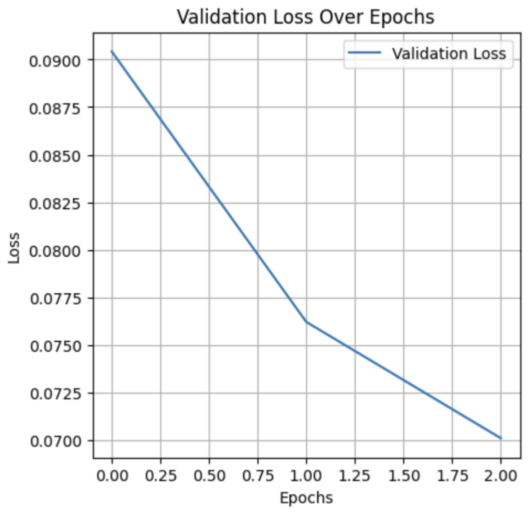


Figure 9: Validation Loss for T5 Model

2.4 Marian MT Model

Marian MT is an advanced Neural Machine Translation framework, known for its efficiency and effectiveness in language translation tasks.The Architecture is as in Figure 10. The Transformer architecture within Marian MT is characterized by its encoder-decoder structure. The encoder processes the input text by employing self-attention mechanisms, which enable the model to weigh the relevance of each word in the context of the entire sentence. The decoder then generates the translated text, also utilizing self-attention, along with an additional attention mechanism that focuses on the encoder's output. This architecture allows for parallel processing of data, drastically improving the speed and efficiency of translation tasks.The Training and Validation Losses of this model are shown in Figures 11 and 12.

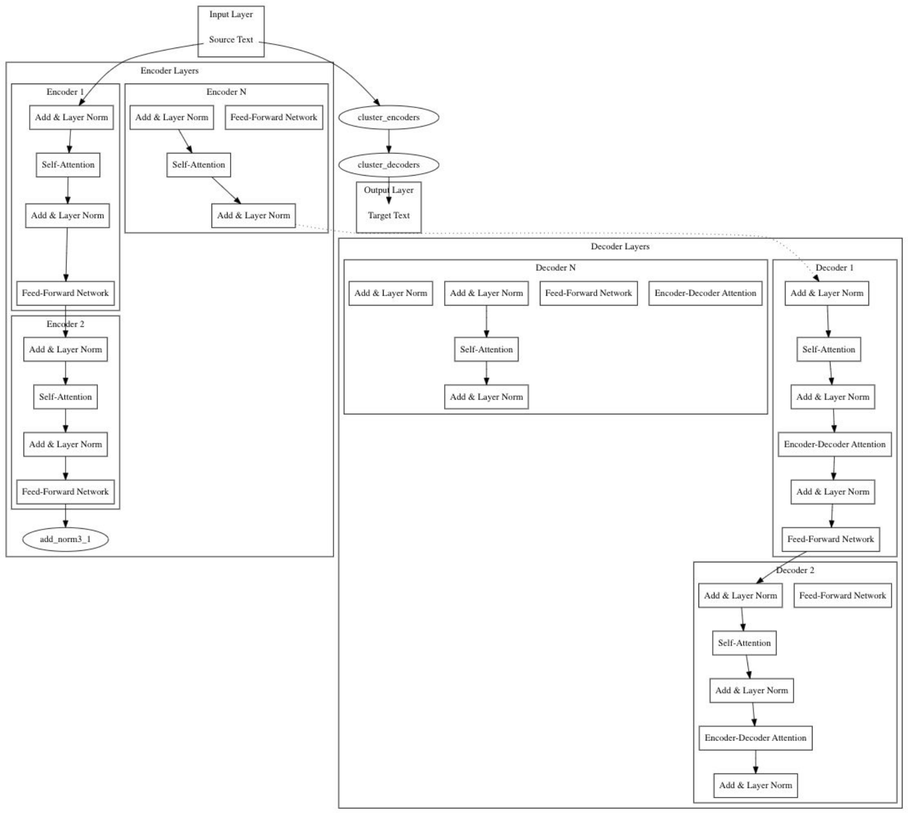


Figure 10: Marian MT Model Architecture

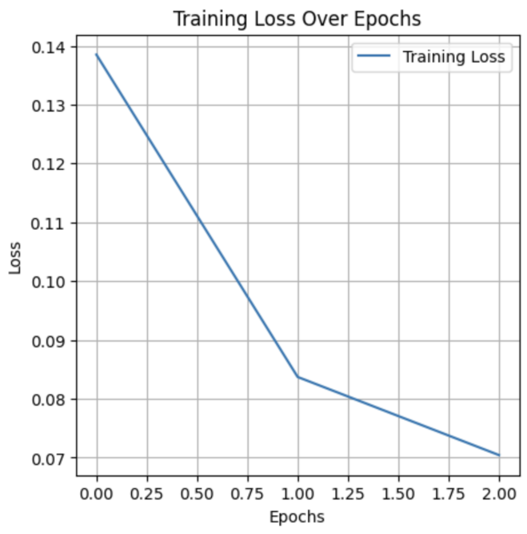


Figure 11: Training Loss for Marian MT Model

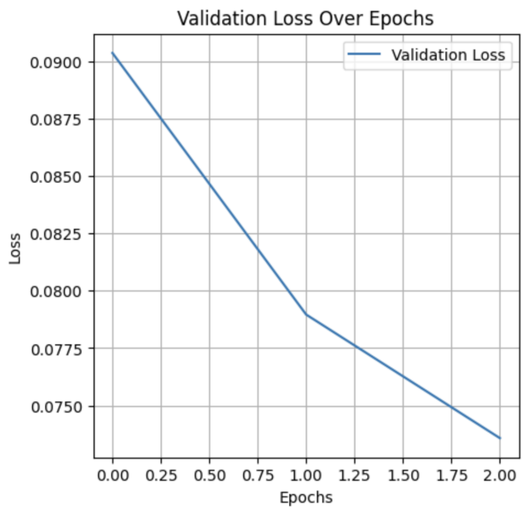


Figure 12: Validation Loss for Marian MT Model

The comparision metrics for these models can be found in Table 1 and 2. Comparison of Model Metrics

3 Loss Function

Loss functions measures the discrepancy between the model's predicted and the actual target values. Essentially, it guides the training process, penalizing the model more as the deviation between prediction and reality increases, ensuring the model is steered towards more accurate predictions.

For our Language Translation Service, we have used NLL Loss Function.

3.1 Negative Log-Likelihood (NLL) Loss Function

The Negative Log-Likelihood (NLL) Loss is a key loss function in machine learning for models outputting probabilistic predictions. It calculates the loss by taking the negative logarithm of the probability assigned by the model to the correct class, thus helping the model to increase accuracy in its predictions. In the context of sequence-to-sequence models for language translation services, NLL Loss is crucial for evaluating how well the model translates a sequence of words. It penalizes the model for low probabilities assigned to correct translations, ensuring accuracy not only in individual word predictions but in the overall sequence alignment and translation coherence. This makes NLL Loss integral in refining sequence-to-sequence models, pushing them towards more precise and reliable language translations.

We have also tried other loss functions like the Cross-Entropy Loss but it was not performing well with this task.

3.2 Cross Entropy Loss

Cross-Entropy Loss is a fundamental loss function, crucial in models where predictions are probabilistic. It evaluates a model by comparing the predicted probability distribution against the actual distribution, effectively penalizing discrepancies. In sequence-to-sequence models for language translation, Cross-Entropy Loss is vital for assessing translation accuracy. It measures the loss for each word in the translated sequence, ensuring that the model's output aligns closely with the target language.

3.3 Innovations on Loss Functions

In our development process, we did not innovate in the area of loss functions. We found that the existing loss functions we applied were already suitable and performed effectively. Therefore, we decided to continue using these well-functioning loss functions rather than seeking novel alternatives for our language translation service.

4 Optimization Algorithm

In our project, we have used three different optimization algorithms to train various models. And the Adam Optimization Algorithm was working well than the other two. The different algorithms are:

4.1 Adam Optimizer

The Adam optimizer, standing for Adaptive Moment Estimation, is a pivotal algorithm enhancing the traditional stochastic gradient descent (SGD) method. It uniquely adjusts the learning rate for each neural network weight, using both the first and second moments of the gradients (mean and uncentered variance). Adam combines features of AdaGrad and RMSProp, notably using RMSProp's approach to gradient second moments but without mean adjustment. In sequence-to-sequence models for language translation, Adam's efficiency is particularly beneficial. It enables faster convergence and improved performance by adeptly navigating the complex optimization landscape, crucial for the intricate task of translating languages. This makes Adam a preferred optimizer for training sequence-to-sequence models, significantly enhancing their accuracy and effectiveness in language translation tasks.

4.2 SGD Optimizer

Stochastic Gradient Descent (SGD) is a fundamental optimization algorithm in machine learning, known for its simplicity and effectiveness. It updates model parameters iteratively, using a single data point at a time, which makes it computationally efficient, especially for large datasets. In sequence-to-sequence models for language translation services, SGD plays a crucial role. While it may not be as fast as more advanced optimizers like Adam, its straightforward approach to minimizing the loss function is valuable in navigating the high-dimensional spaces typical of sequence-to-sequence models.

4.3 AdaGrad Optimizer

Adagrad (Adaptive Gradient Algorithm) is an optimization algorithm, that adjusts the learning rate individually for each parameter, making it particularly effective for dealing with sparse data. In sequence-to-sequence models for language translation, Adagrad's approach is advantageous as it allows for fine-tuned adjustments based on the frequency of features in linguistic data. However, a potential drawback is its accumulated gradient in the denominator, which can lead to a diminishing learning rate, potentially impacting long-term training efficiency. Despite this, Adagrad's tailored learning rate strategy makes it a valuable tool in optimizing language translation models.

4.4 Innovations on Optimization Algorithms

In our exploration, we utilized a range of standard optimization algorithms without introducing any novel modifications to them. Our main objective was to evaluate how different models performed using these conventional algorithms. This was done to determine the most efficient strategy for our project, which involves language translation.

4.5 Comparisions

Figures 13 and 14 illustrate our comprehensive analysis where we compared various optimization algorithms and loss functions within a single model, demonstrating their performance within the Encoder Decoder with Multi Head Attention Mechanism Model.

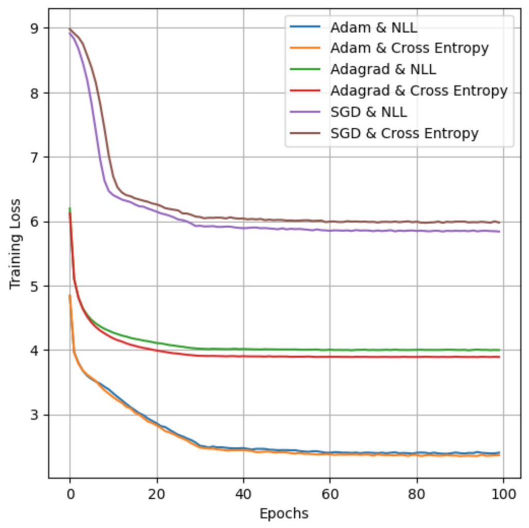


Figure 13: Training Loss for different Loss Functions and Optimization Algorithms

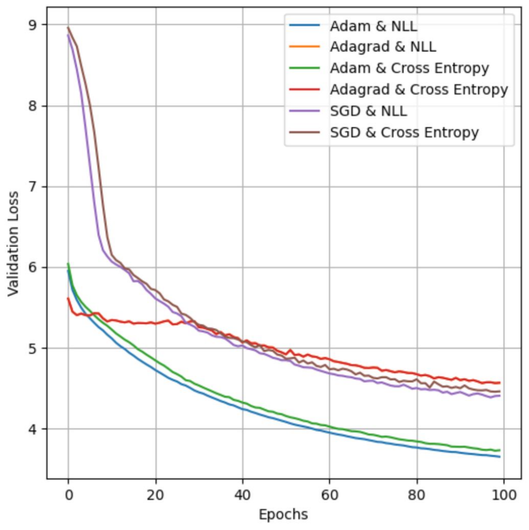


Figure 14: Validation Loss for different Loss Functions and Optimization Algorithms

5 Metrics and Experimental Results

Below are the Metrics that have been used to evaluate our Language Translation Service.

1. BLEU Score: BLEU Score is a recognized method, used in the field of language processing to evaluate the quality of machine-translated text, using human translation as a reference. It works by analyzing segments of text, from individual words to groups of four words, to check for matches with reference text. BLEU focuses on the accuracy of the word or sentence contained in the machine translation compared to the reference document. It calculates separate scores for groups of words of different lengths and merges them into an overall score.

2. METEOR Score: METEOR Score provides a more sophisticated assessment of machine translations. It not only checks the accuracy of the translation (like BLEU) but also considers the content scope of the reference. It stands out for its ability to recognize synonyms and different word forms, allowing for more flexible and meaningful translation evaluation. In addition, METEOR also pays attention to word order, helping to evaluate the grammatical and structural accuracy of the translation. This measure is generally considered more reflective of human judgment due to its balanced approach between word accuracy and overall language mastery. However, the complexity of METEOR, especially regarding its requirements for high-quality linguistic databases of synonyms and word form recognition, can make its effective implementation difficult.

3. BERT F1 Score: BERT F1 goes beyond standard F1 metrics by integrating the context and subtlety inherent in the text. It evaluates the relevance of the predicted text to the reference material in a more refined way, considering not only the words used but also the underlying context they convey. This makes it a more complex tool for evaluating the accuracy and relevance of text predictions.

4. BERT Precision Score: BERT Precision uses BERT's advanced language understanding capabilities to improve accuracy measurements. This approach explores deeper layers of language, including context and subtle meanings, to give a more accurate assessment of how accurately the predicted text matches the expected results, especially in complex language situations.

5. BERT Recall Score: BERT recall focuses on accurately identifying the proportion of correctly identified relevant cases in the prediction text, using BERT embeddings for deeper understanding. This metric not only evaluates surface-level accuracy but also assesses the contextual relevance of predictions, providing a more comprehensive view of recall performance in linguistic tasks.

Scores / Models	**Encoder-Decoder with Attention**	**T5 Model**	**Marian MT Model**
Bleu Score	26.56	19.72	41.28
Meteor Score	34.25	48.09	69.73
BERT Precision	84.45	93.16	94.97
BERT Recall	85.70	90.68	94.95
BERT F1	85.06	91.88	94.96

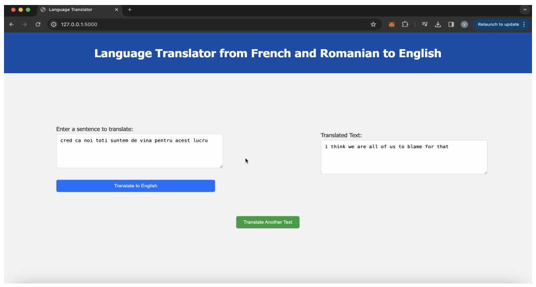
Table 1: Comparison of Model Metrics for Romanian to English

Scores / Models	**Encoder-Decoder with Attention**	**T5 Model**	**Marian MT Model**
Bleu Score	26.68	20.86	28.78
Meteor Score	38.36	47.64	56.75
BERT Precision	86.46	91.21	93.15
BERT Recall	86.78	90.84	92.96
BERT F1	86.61	92.65	93.05

Table 2: Comparison of Model Metrics for French to English

The above Table 1 shows the metrics compared among different models in Romanian to English and Table 2 shows the metrics for French to English.

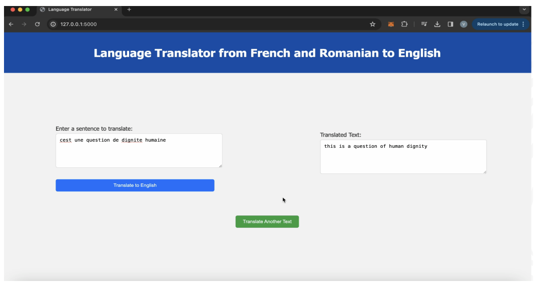
We have also created a user-friendly web application that interfaces with our developed model, leveraging Flask, HTML, CSS, and JavaScript technologies. This application is designed to accept input in either French or Romanian. One of its key features is the automatic detection of the input language, facilitated by the integration of the Google API. Once a user inputs a sentence in either of these languages, the application efficiently identifies the language without any manual selection. Following this, when the user clicks on the 'Translate to English' button, our application employs the underlying model to translate the provided sentence into English. This seamless process, from automatic language detection to translation, makes the application both intuitive and efficient for users seeking quick and accurate translations. Below Figures 15 and 16 are the screenshots of the Web Application of Language Translation Service.



cred ca noi toti suntem de vina pentru acest lucru

i think we are all of us to blame for that

Figure 15: Translation from Romanian to English



cest une question de dignite humaine

this is a question of human dignity

Figure 16: Translation from French to English

6 Contributions and Github

Below in Table 3 are the percentage of contributions made by each team member.

Github Repository Link : https://github.com/thilakreddy4304/LanguageTranslationService

Team Member	Contribution Percentage	Responsibilities
vkoyyada	33.33%	Data Engineering, Encoder Decoder Multi Head Attention, Documentation
tejanave	33.33%	Data Engineering, T5 Model, Documentation, WebApp
thilakre	33.33%	Data Engineering, MarianMT Model, Documentation, WebApp

Table 3: Contributions of each team member to the project.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473, 2014.

[3] https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

[4] https://huggingface.co/docs/transformers/model_doc/t5

[5] https://huggingface.co/docs/transformers/model_doc/marian