Language Translation Service

Divya Pamidi
University at Buffalo
divyapam@buffalo.edu

Sai Sasidhar Vemavarapu
University at Buffalo
saisasid@buffalo.edu

Harsha Vardhan Bitra
University at Buffalo
hbitra@buffalo.edu

Abstract

This project involves creating a language translation model using deep learning. It trains on numerous English sentences and their translations in French and German, improving its ability to understand and translate new sentences. This advanced neural machine translation tool aims to reduce language barriers and facilitate communication between speakers of different languages by learning from vast datasets of translated text. Unlike rule-based systems, it adapts to complex linguistic structures, offering more natural-sounding translations.

1 Data-set

For our project we used Europarl data-set. The Europarl dataset is a collection of transcripts from the European Parliament proceedings, available in 21 European languages. It's a key resource in NLP and machine translation research, offering parallel texts aligned sentence-by-sentence or paragraph-by-paragraph across different languages. This alignment makes it an excellent tool for training and evaluating machine translation systems. The dataset's diversity, encompassing various political and legislative discussions, adds real-world complexity, aiding in the development of robust, multilingual NLP models. While invaluable for its authentic, diverse content, its political nature and language imbalance present unique challenges for language processing tasks.

1.1 Data Pre-processing

Before modeling, key modifications were made to the dataset to optimize results and minimize loss:

- Downloading and Extracting Data
- Reading and Splitting Text
- Cleaning Text Data: Removing non-ASCII characters, lowercasing all words, removing punctuation, filtering out non-alphabetic characters and words with numbers.
- Saving Processed Data
- Building Vocabulary
- Normalizing strings
- Filtering Data Dataset Splitting
- Split the filtered pairs into training, validation and testing subsets (90
- Model Training Preparation

2 Model Description

2.1 LSTM

The Long Short-Term Memory (LSTM) algorithm is a sophisticated recurrent neural network design created to tackle difficulties inherent in grasping long-term relationships within sequential data. This

unique LSTM model, equipped with special memory compartments and gate mechanisms, aptly solves problems such as disappearing or escalating gradient issues; hence they are perfectly suited for projects involving natural language processing and time series analysis. The selective storage and retrieval structure of LSTMs contributes towards robust rendering of intricate sequence associations.

2.2 GRU

The Gated Recurrent Unit (GRU) is a type of RNN that efficiently processes sequential data, such as language, using a simpler structure with reset and update gates. This design balances computational efficiency with performance, making GRUs popular for digital language processing and complex time-series analysis.

In our project, we focused on LSTM and GRU models for language translation, utilizing an encoder-decoder framework suited for sequential tasks. We integrated the beam search algorithm for enhanced prediction during decoding and fine-tuned the models by adjusting parameters like the number of layers and hidden size. We also implemented dropout in GRU training to handle language variability.

2.3 GRU with Bahdanu Attention

The GRU-Bahdanau attention combination fuses the functionality of Gated Recurrent Unit (GRU) and Bahdanau's selective focus mechanism to boost its ability in capturing complex linkage patterns in sequenced data. This harmonized fusion enhances both sequential processing efficiency, thanks to the GRU element, and sensitivity towards critical details therein courtesy of the Bahdanau attribute - which is particularly beneficial for work such as machine translation demanding detailed comprehension of context interdependencies. Its incorporation lays out an effective strategy primarily within natural language interpretation tasks plus other related areas.

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1}\boldsymbol{h}_t + \boldsymbol{W_2}\bar{\boldsymbol{h}}_s\right)$$

2.4 GRU with Luong Attention

The integration of the Gated Recurrent Unit (GRU) with the Luong attention mechanism significantly enhances the handling of sequential data, particularly in language decoding tasks. In this approach, the GRU first encodes the input, such as a sentence in a source language, by generating a hidden state for each input token. This encoding is similar to traditional models but becomes more effective with the addition of Luong attention.

Luong attention enhances the model by focusing on specific parts of the input sequence during decoding, addressing a limitation of traditional fixed-length context vectors. It calculates alignment scores between the current hidden state of the decoder and each of the encoder's hidden states. These scores are obtained by passing the hidden states through linear layers and a tanh activation function, followed by multiplication with a weight matrix.

These alignment scores are then normalized to form attention weights, signifying the relevance of each input token in predicting the current target token. As a result, the context vector, derived as a weighted sum of the encoder's hidden states, is dynamic and adjusts at each decoding step. This flexibility allows the decoder, often another GRU, to focus on relevant parts of the input sequence, leading to more accurate and contextually appropriate output.

Overall, the combination of GRU and Luong attention forms a powerful framework for language translation. It offers a nuanced understanding of linguistic structures and adapts to varying sentence complexities, making it a superior choice for complex language processing tasks.

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]}$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]}$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]}$$

2.5 T5 Transformer

The T5 Transformer, or 'Text-to-Text Transfer Transformer', is an innovative neural network model adept at handling a wide array of text-based tasks. This includes translation, summarization, and question-answering. Its unique strategy involves treating all linguistic operations as text-to-text transformations, thereby increasing its adaptability and efficiency in processing complex language data.

A central component of the T5 model is the T5 Tokenizer, which segments text into manageable tokens, streamlining data structuring for effective language manipulation. This tokenizer plays a crucial role in the model's ability to understand and process language data accurately.

In optimizing the T5 Transformer's performance, some experiments were implemented. The weight decay parameter was set to 0.05, serving as a regularization technique to control the scale of model weights and prevent overfitting. Furthermore, dropout techniques were introduced, with dropout rates including dropout_rate and attention_dropout set to 0.1. This approach aids in generalization by randomly deactivating neurons during the training phase, thus promoting the development of more robust features. Hyperparameter tuning was another focus area, with variations in learning rates (1e-4, 5e-5, 1e-5) and batch sizes (16, 32, 64) being explored to optimize the learning process and assess their impact on model performance.

2.6 Attention-based Transformer (Our Approach)

The Transformer model, a pivotal advancement in machine translation, is detailed in the landmark paper *Attention Is All You Need*. It marks a departure from recurrent layers typical in RNNs and LSTMs, favoring an architecture built entirely on attention mechanisms, predominantly self-attention. This design has proven highly effective in natural language processing tasks.

- **Architecture:** The Transformer employs a sequence-to-sequence framework, suitable for machine translation, characterized by its non-recurrent structure.
- **Encoder-Decoder Structure:**
  - The *Encoder* processes the input (e.g., an English sentence) through multiple layers, each containing self-attention and feed-forward networks.
  - The *Decoder* generates the target language output (e.g., a German translation), also layered with self-attention and encoder-decoder attention.
- **Self-Attention Mechanism:** This feature allows the model to assign varying weights to different parts of the input, enhancing its focus on relevant aspects for translation.
- **Positional Encodings:** Integrated to inform the model of the word order within the sequence, compensating for the lack of recurrent layers.
- **Multi-Head Attention:** Used in both the encoder and decoder, it enables simultaneous focus on various parts of the sequence, capturing complex dependencies.
- **Layer Normalization and Dropout:** Incorporated for training stabilization and overfitting prevention. Layer normalization follows each sub-layer, and dropout randomly eliminates connections during training.

In essence, the Transformer model's adeptness in sequence-to-sequence tasks, like language translation, is attributed to its innovative use of self-attention and multi-head attention mechanisms, coupled with positional encodings. These features collectively enable it to effectively capture intricate linguistic relationships and dependencies.

3 Loss Function

In adopting the Label Smoothing Cross-Entropy Loss strategy in our Transformer models for language conversion, we've refined and improved upon traditional Cross-Entropy Loss methods. Uniquely suited to complex assignments such as machine translation, it dials down model overconfidence while effectively managing linguistic uncertainties. It further sharpens calibration of models and offers strong resilience against any label noise disturbance. This change within the loss function in our

Transformer model was aimed at surmounting these challenges by modifying training objectives and evenly distributing slim probabilities across all categories involved. The outcome has been a significant uptick in translation efficiency owing to decreased losses; thus testifying better handling of fine points of languages along with excellent power projection capabilities onto unexplored data fields. Hence, applying this novel technique -Label Smoothing Cross Entropy Loss – is endorsed especially when key factors include calibrating modeling accurately or dealing efficiently with multifarious lingual expressions.

4 Optimization

An optimizer must possess the key traits of quick convergence and robustness towards randomness to identify the global optimum. Local optima can suffice for practical purposes, and these can be detected by gradient-focused methods. PyTorch provides a selection of optimizers, each equipped with its distinct optimization algorithm.

Adam (Adaptive Moment Estimation): Adam, combining AdaGrad and RMSProp's advantages, adjusts learning rates per parameter. This makes it effective for handling sparse gradients and varying parameter scales.

SGD (Stochastic Gradient Descent): SGD, a fundamental optimizer, adjusts weights based on the loss function's negative gradient but can get trapped in local minima.

Comparative Analysis: Comparative experiments between Adam and SGD illustrate performance differences:

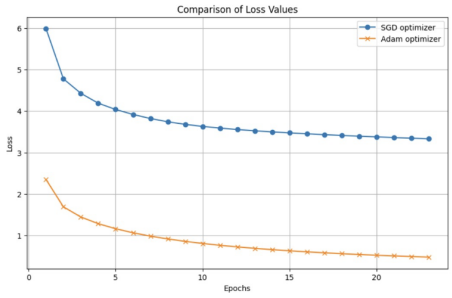


Figure 1: Comparison of loss values between SGD and Adam optimizers.

We have trained our model using different optimizers like Adam and SGD (stochastic gradient descent). We have observed that our model performed better with Adam optimizer when compared with SGD optimizer. The initial loss value with SGD optimizer is very high around 5 but with Adam optimizer it is only around 1. The loss values for both optimizers have been reduced constantly. With Adam optimizer we have achieved lower loss compared with SGD optimizer.

5 Contributions and Website

The contributions to this project were evenly distributed among the team members. The following table lists the contributors:

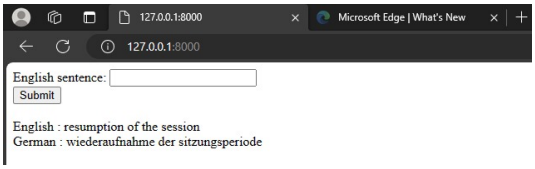
Name	Contribution
Sai Sasidhar Vemavarapu	33.33%
Divya Pamidi	33.33%
Harsha Vardhan Bitra	33.33%

Table 1: Contributor Contributions

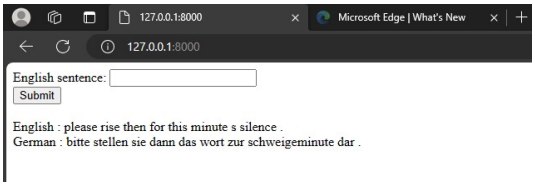
5.1 Website Outputs

The project's website provides detailed information on our research and outcomes:

(a) Output 1



(b) Output 2



(c) Output 3

Figure 2: Screenshots of the website outputs.

For more details and code file on website, we're uploading the folder while submitting the report. We are not able to upload to git as there is size limit issue. Also please refer to our Git repository: `https://github.com/divya-pamidi/dl-project`

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems. Available: `https://arxiv.org/abs/1706.03762`

[2] PyTorch Tutorials. Available: `https://pytorch.org/tutorials/`

[3] Bahdanau, D., Cho, K., Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate. In ICLR 2015. Available: `https://arxiv.org/abs/1409.0473`

6 Metrics and Experimental Results

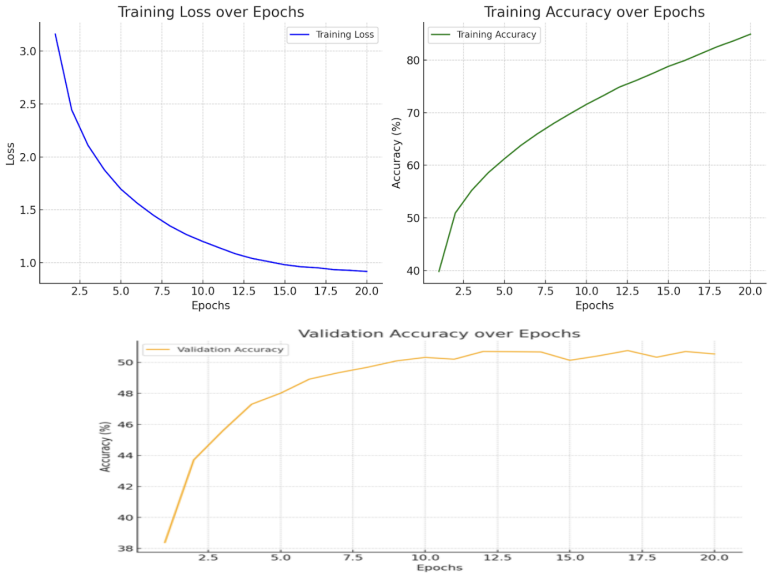


Figure 3: LSTM results

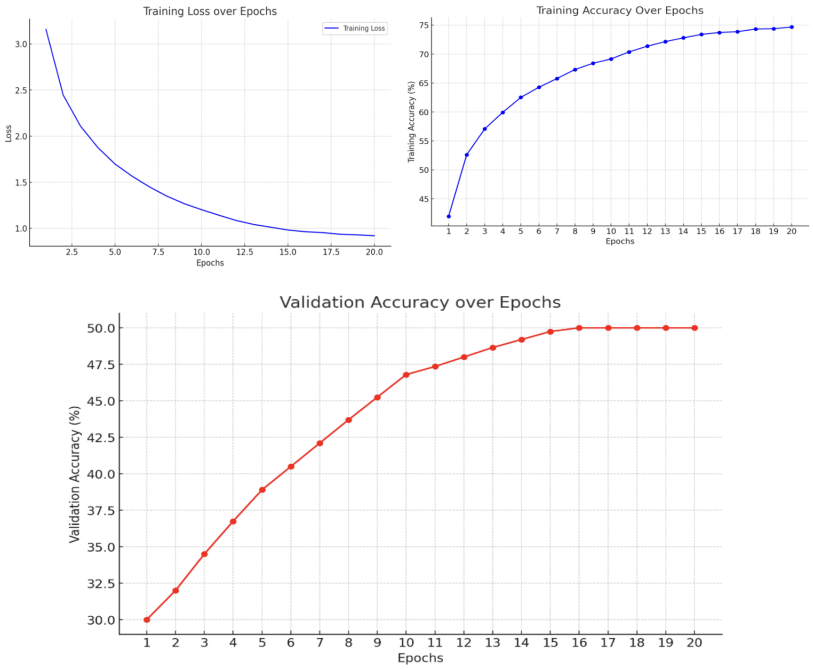


Figure 4: GRU results

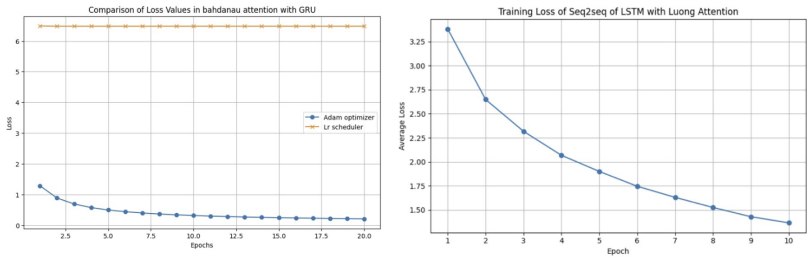


Figure 5: Training Loss of GRU with Bahdanau attention and LSTM with Luong attention

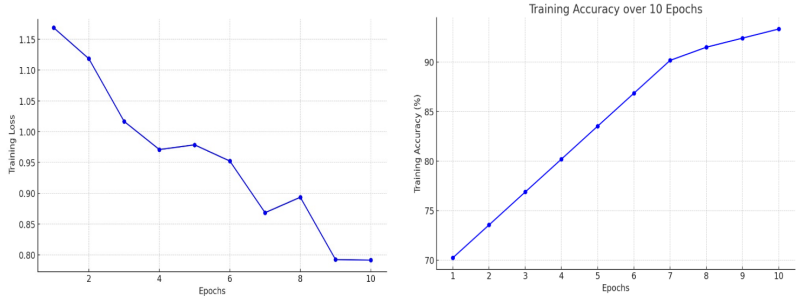


Figure 6: T5 results

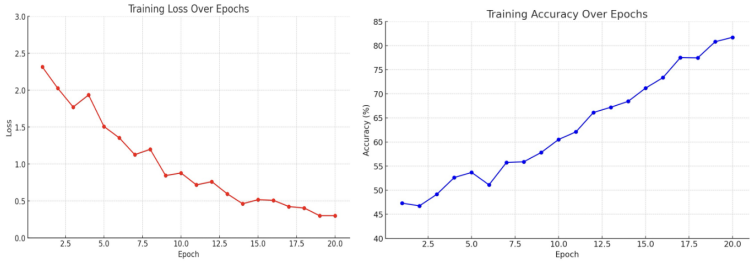


Figure 7: Transformer with attention results