M2 DSC - Université Jean Monnet

# Text To Text Translation

Deep learning

Idriss BENGUEZZOU
Ghilas MEZIANE
Fatiha KHALYFA

22/01/2024

# Table des matières

# 1 Introduction

## 1.1 Problem Definition

In the context of our second year of Master's in Data & Connected Systems at Jean Monnet University, we undertook this academic project, which involved developing a system for translating text from French to English using deep learning techniques. The task of translating text from French to English is a big challenge in the field of natural language processing. We are aiming to explore various architectures and strategies to enhance translation accuracy and efficiency. Text translation finds its applications in various domains, making it a pivotal tool in bridging cultural and linguistic differences.

## 1.2 Dataset Overview

In this project we'll use the Parallel Global Voices dataset, a collection of English-French parallel texts sourced from a Kaggle competition and containing 342,060 samples. The Parallel Global Voices dataset is derived from Global Voices, a multilingual network of websites (http ://globalvoices.org/), where volunteers translate and publish news stories in over 40 languages. The original content is available under a Creative Commons Attribution license, as provided by the authors and publishers. Published in July-August 2015 by the NLP group of the Institute for Language and Speech Processing, the dataset pairs documents based on their link information, identifying those that are translations of one another. Following this, segment alignments were automatically extracted. Both document pairings and segment alignments are distributed under a Creative Commons Attribution license, facilitating their use in academic research and machine learning applications.

## 1.3 Objectives

The primary objective of this academic project is to explore, experiment with, and evaluate various deep learning models and techniques within the context of text translation by developing an efficient machine translation system capable of translating French text to English. It aims to provide practical experience and validate the theoretical knowledge we have acquired during our Master's program in Data & Connected Systems at Jean Monnet University, and gain insights into the challenges and opportunities in the field of natural language processing. This project serves not only as a platform for academic exploration but also as a means to demonstrate our understanding of advanced machine learning concepts in a real-world application.

# 2 Metrics Chosen

## 2.1 BLEU Score

Evaluating the performance of machine translation models requires metrics that accurately reflect the quality of translations in terms of linguistic correctness, fluency, and similarity to the target language. For this project, we have chosen to use the BLEU (Bilingual Evaluation Understudy) Score as our main metric for evaluation. The calculation of BLEU Score involves the following steps :

1. **N-Gram Precision :** For each n-gram (e.g., unigram, bigram) in the translated text, BLEU counts the maximum number of times it appears in any single reference translation. The count is then divided by the total number of n-grams in the translated text to calculate the precision.

$$Precision = \frac{\text{Number of words in the output text that occurred in the reference text}}{\text{Total number of words in the output text}}$$

2. **Clipping :** To prevent over-counting of frequent n-grams, BLEU applies a clipping mechanism, where the counts are capped at the maximum count in the reference.

$$ClippedPrecision = \frac{\text{Clipped number of words in the output text that occurred in the reference text}}{\text{Total number of words in the output text}}$$

3. **Brevity Penalty :** To penalize overly short translations, BLEU includes a brevity penalty factor. If the length of the translated text is less than the reference, the BLEU score is multiplied by a penalty factor.

$$\text{Brevity Penalty} = \begin{cases} 1 & \text{if } c > r \\ e^{\left(1-\frac{r}{c}\right)} & \text{if } c \leq r \end{cases}$$

$c$ is the number of words in the output text, and $r$ is the number of words in the reference text.

4. **Geometric Mean :** BLEU Score is the geometric mean of the n-gram precisions, typically calculated for unigrams up to 4-grams, multiplied by the brevity penalty.

$$\text{Weighted Geometric Mean Precision} = \prod_{n=1}^{N} CP_n^{w_n}$$

Given that CP$_n$ represents the clipped precision for n-grams and $w_n$ are the weights assigned to each clipped precision.

### 2.1.1 Pros

After some research we found out that the BLEU Score is a widely accepted metric in the field of natural language processing and machine translation. It quantifies the correspondence between a machine's output and a human translation, focusing on the precision of word choices and the order of phrases. BLEU score is calculated based on the comparison of n-gram overlap between the translated text and a set of reference translations, by providing a quantitative measure of translation quality.

One of the key reasons for choosing BLEU Score is its ability to provide a consistent and comparative measure of translation quality across different models and datasets. It offers a balance between fluency and adequacy of translation, making it suitable for evaluating our models trained on the Parallel Global Voices dataset. Finally, the Blue Score calculation is simple, explainable and language-independent.

### 2.1.2 Cons

### 2.1.3 Limitations of BLEU Score

While the BLEU Score is widely used for evaluating machine translation models, it has notable shortcomings, particularly in understanding the semantics and significance of words in a translation. Key limitations include :

— **Lack of Semantic Understanding :** BLEU does not consider the meanings of words or their significance in the context of the text. It evaluates translations based on surface-level lexical matches rather than capturing the underlying semantic content.

— **Equal Weight to All Words :** In the BLEU metric, all words, regardless of their importance in the sentence, are treated with equal significance. For instance, propositions, which generally hold less importance in conveying the core message, are given the same weight as key nouns and verbs. This can lead to skewed assessments where minor lexical discrepancies significantly impact the score.

— **Inability to Handle Word Variants :** BLEU is limited in its ability to recognize and evaluate different variants of words. It fails to account for synonyms, paraphrasing, or alternative expressions that might convey the same meaning as the reference but use different wording.

— **Word Order Limitations :** While BLEU considers the order of words through n-gram matching, it does not fully capture the complexities of word order in language. As a result, translations with rearranged but semantically correct word order may be unduly penalized.

## 2.2 Why not the raw model accuracy

In contrast, using raw model accuracy, typically reported during training epochs, is not a reliable indicator of translation quality in machine translation tasks. Accuracy in this context often refers to the percentage of correctly predicted tokens or words, which does not necessarily translate to meaningful or contextually accurate translations. A high accuracy score might still result in translations that are grammatically incorrect or semantically inconsistent with the source text. This is particularly true for complex translation tasks like ours, where linguistic nuances and context play a crucial role. Therefore, relying only on model accuracy can be misleading and does not provide a complete idea of the model's performance in translating text from French to English.

In summary, the BLEU Score is chosen as it offers a more comprehensive and relevant evaluation of translation quality, capturing aspects that raw accuracy metrics might overlook. It aligns well with the objectives of our project, enabling us to assess the effectiveness of our translation models in a more meaningful way.

# 3 Exploratory Analysis

Before diving into the experimental phase of our project, it was imperative to acquire a comprehensive understanding of the dataset's nature and structure. To achieve this, we implemented an exploratory analysis through a Jupyter notebook, designed to gather a maximum of relevant statistics bout the Parallel Global Voices dataset.

## 3.1 Our logic for the exploratory analysis

The exploratory analysis was motivated by several purposes :

- **Understanding Data Composition :** By computing basic statistics such as average sentence length and identifying special characters, we gain an appreciation of the textual data's complexity and composition.
- **Data Quality Assessment :** The absence of missing values and the calculation of vocabulary size are indicative of the dataset's quality and its adequacy for training robust machine translation models.
- **Preprocessing Insights :** The number of unique words post-preprocessing provides a measure of the linguistic diversity and will serve us our data cleaning methods.
- **Distribution Analysis :** Examining the distribution of English and French text lengths assists in understanding the alignment within the corpus, which is critical for model training and evaluation.

## 3.2 Key Findings

Our exploratory analysis revealed several key findings :

- The dataset contains a significant presence of special characters, with 298,261 samples including such elements. This underscores the need for careful preprocessing to handle non-standard tokens which may affect translation quality.
- Number of unique words after pre processing : 6 627 178 English words vs 7 357 642 French words

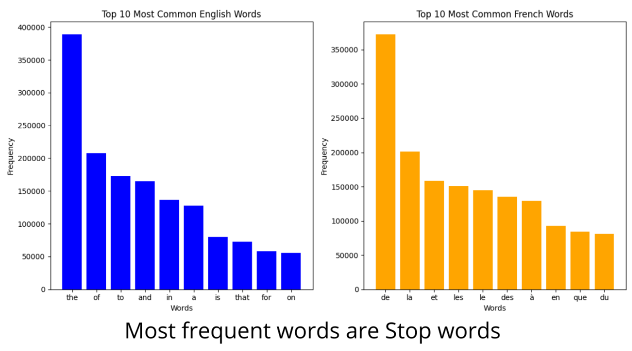


Figure 1 – Top 10 Most common words - English vs French

- Numerical values appeared in 48,642 samples, highlighting the necessity to either normalize or adequately represent numerical information within the translation process.
- Post-preprocessing, the dataset contains a diverse range of unique words, confirming the linguistic richness and the potential complexity the translation models need to handle.
- The length distribution analysis of English and French texts provided a visual representation of sentence structures, informing the expected sequence lengths and aiding in the design of model architectures and training regimes.

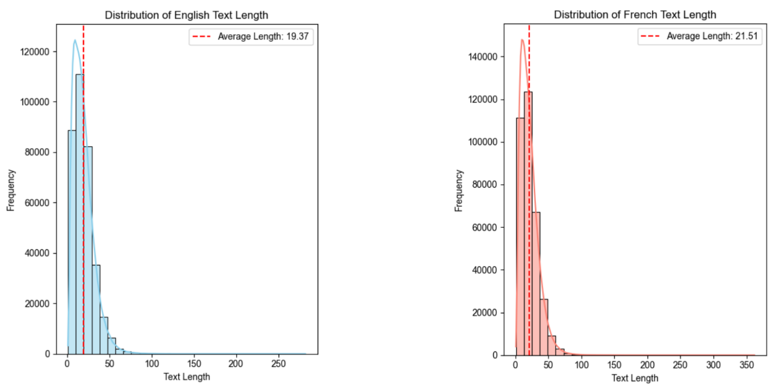


FIGURE 2 – Text Distribution English vs French

Thanks to this exploratory analysis, we were able to adjust our preprocessing techniques and model parameters to better align with the dataset's characteristics, setting a solid basis for the experimental phase.

### 3.3 Preprocessing Decision and Class Implementation

Following the exploratory analysis, a decision was made to implement a `TextProcessor` class for consistent preprocessing. The selected preprocessing steps include :

- **Lowercasing :** To ensure uniformity in text.
- **Contractions Expansion :** To handle contractions appropriately.
- **Unicode Normalization :** To handle different Unicode representations.
- **Removing Non-ASCII Punctuation :** To clean the text from non-ASCII characters.
- **Removing Special Characters :** To clean irrelevant characters and symbols.
- **Removing Extra Spaces :** To maintain text consistency.

These preprocessing steps were chosen to address the observed characteristics of the dataset and ensure a clean and standardized input for the translation model.

# 4 Experimentation

In this section, we present the various experimental approaches that we tried to implement to build our machine translation models. The experimentation process began with the implementation of a baseline model, followed by more complex architectures, allowing us to incrementally evaluate and enhance the translation quality.

## 4.1 Simple Baseline Model

### 4.1.1 Strategy

Our initial step was to establish a baseline model, which we refer to as a simple Seq2Seq model. This model serves as a starting point for our experimentation, providing a basic framework for text-to-text translation.

### 4.1.2 Experimental Setting

The baseline Seq2Seq model is structured as follows :

- An **Input layer** that receives the text sequences.
- A **Long Short-Term Memory (LSTM) layer** to process the input and capture the context within the sequences.
- A **TimeDistributed Dense layer** that generates a probability distribution over the output vocabulary for each timestep, enabling the generation of the translated text.

We used the **RMSprop optimizer**, commonly recommended for recurrent neural networks, and the ***sparse categorical crossentropy*** as the loss function.

During the experimentation phase, we trained the model on different parameters, including the number of epochs, batch size, and LSTM units ect.. We also experimented with different optimizers and loss functions. From our exploratory analysis, we observed that the average length of the English and French sentences was arround 20 words. Therefore we also experimented with different sequence lengths, ranging from 10 to 50 words. The vocabulary size was also varied, we tried to train the model the whole vocabulary of the dataset, and we also tried to limit the vocabulary size to 5 000 words.

### 4.1.3 Results

The first result we obtained with this baseline model was :

- **Model BLEU Score on Test Data :** $1.006 \times 10^{-231}$
- **A concrete example :** A concrete example of the translation made by this model : *translate*("Bonjour comment ça va") = 'the the the'

We obtained these result by training the model with the following parameters :

| | |
|---|---|
| **Loss Function** | Sparse Categorical Crossentropy |
| **Optimizer** | RMSprop optimizer (lr = 0.001) |
| **Epochs** | 10 |
| **batch size** | 128 |
| **LSTM units.** | 64 |
| **Vocabulary size** | All |
| **max sequence length** | 50 |

Table 1 – Parameters of the simple Baseline Model

As we can see, the BLEU score is very low, and the model only predicts the same word for every input sentence. After that we tried to improve the model by changing the parameters, we did improve the result but not significantly. The best BLEU score we obtained with this model was 0.0749, and it was obtained with the following parameters :

- **Model BLEU Score on Test Data :** 0.0749
- **A concrete example :** A concrete example of the translation made by this model : *translate*("je suis fatigué") = 'i am the'

| | |
|---|---|
| **Loss Function** | Sparse Categorical Crossentropy |
| **Optimizer** | RMSprop optimizer (lr = 0.0001) |
| **Epochs** | 30 |
| **batch size** | 64 |
| **LSTM units.** | 126 |
| **Vocabulary size** | 5 000 |
| **max sequence length** | 20 |

Table 2 – Parameters of the simple Baseline Model

Even if this model is our first dive into the project, these initial results, particularly the very low BLEU score, indicated the need for more advanced model architectures and strategies to improve the translation quality.

## 4.2 Baseline Model with Bidirectional and Embedding Layers

### 4.2.1 Strategy

Building upon our initial baseline model, we enhanced the architecture by incorporating bidirectional LSTM layers and embedding layers. The enhancement of our baseline Seq2Seq model with bidirectional LSTM and embedding layers is motivated by the need to capture more nuanced and contextually rich features of the language, which are essential for producing accurate translations.

This new version of the baseline model aims to better capture the context and intricacies of the language, ultimately leading to more accurate translations.

### 4.2.2 Experimental Setting

The architecture of the enhanced model includes :

- **Input Layer** : Accepts input sequences of fixed length, specified as 20 tokens per sequence.
- **Embedding Layers :** Embedding layers transform discrete, symbolic input tokens (words or characters) into dense vectors of fixed size. This representation allows the model to capture semantic and syntactic relationships between words, leading to a more meaningful understanding of the input text. Embeddings are particularly effective in reducing the dimensionality of the input space.
- **Bidirectional LSTM Layers :** Processes the embedded sequence in both forward and backward directions, providing a more comprehensive view of the sequence context. This is particularly beneficial for translation tasks, where the meaning of a word or phrase often depends on its surrounding context.
- **TimeDistributed Dense Layer** : Generates the output sequence by applying a Dense layer to each time step of the LSTM output. This layer is crucial for mapping the LSTM outputs to the target vocabulary space, facilitating the generation of translated text.

As for the baseline model, we used the **RMSprop optimizer**, commonly recommended for recurrent neural networks, and the ***sparse categorical crossentropy*** as the loss function. We started with the same parameters as the baseline model that gave us the best results, and we tried to improve the model by changing the parameters. We also tried different embedding dimensions, ranging from 64 to 1024.

### 4.2.3 Results

We tried to fine tune the parameters of the model to improve the BLEU score we first tried different parameters, here is an example of the parameters we used to train the model :

| | |
|---|---|
| **Loss Function** | Sparse Categorical Crossentropy |
| **Optimizer** | RMSprop optimizer (lr = 0.0001) |
| **Epochs** | 20 |
| **batch size** | 64 |
| **Embedding dimension** | 216 |
| **LSTM units** | 128 |
| **Vocabulary size** | All |
| **max sequence length** | 50 |

Table 3 – Parameters of the Baseline Model with Bidirectional & Embedding layers

But the BLEU score we obtained (**0.00912**) was not better than the one we obtained with the baseline model, so we used the same parameters as the baseline model that gave us the best results, and we tried to improve the model by playing with the embedding dimension. We tried different embedding dimensions, ranging from 64 to 1024. The best BLEU score we obtained with this model was 0.00912, and it was obtained with the following parameters :

| | |
|---|---|
| **Loss Function** | Sparse Categorical Crossentropy |
| **Optimizer** | RMSprop optimizer (lr = 0.0001) |
| **Epochs** | 20 |
| **batch size** | 64 |
| **Embedding dimension** | 128 |
| **LSTM units** | 128 |
| **Vocabulary size** | 5 000 |
| **max sequence length** | 20 |

Table 4 – Parameters of the Baseline Model with Bidirectional & Embedding layers

Adding the bidirectional and embedding layers to the baseline slightly improved the BLEU score to **0.0813** compared to the baseline model.

It is worth noting that adding the bidirectional and embedding layers to the baseline increased significantly the number of parameters of the model, which led to a significant increase in the training time and the memory usage. Wich is why we decided to limit the dataset size to 50 000 sentences, instead of using the whole dataset. This could explain why the BLEU score did not improve significantly compared to the baseline model, even though we added bidirectional and embedding layers to the model.

Experimentations on simple seq2seq models did not yield significant improvements in translation quality. This led us to explore more advanced architectures and techniques to enhance the model's performance.

## 4.3 Baseline Encoder Decoder with Attention Mechanism

### 4.3.1 Strategy

After experimenting with the bidirectional embedding model, we decided to further explore the capabilities of neural machine translation by implementing an Encoder-Decoder architecture with an Attention mechanism. This decision was motivated by the following considerations :

**Enhanced Contextual Understanding :** The transition from a bidirectional model to an Encoder-Decoder architecture with an Attention mechanism represents a strategic evolution in our approach to machine translation. Now that we saw that the bidirectional model improved the understanding of context by processing input sequences in both directions, we thought that the Encoder-Decoder architecture would offer a more sophisticated approach to handling sequential data. The separation of comprehending the input (encoding) and producing the translated output (decoding), allows the model to specialize in each task.

**Need for Handling Long-Range Dependencies :** One limitation we observed in our two previous models was the challenge in keeping an acceptable accuracy while translating longer sentences with complex structures. The Encoder-Decoder architecture, especially when augmented with an Attention mechanism, is better to manage long-range dependencies.

**Attention Mechanism :** The addition of the Bahdanau Attention mechanism is here to add the ability to align and weigh different parts of the input sequence while generating each word in the output. This mechanism enables the model to dynamically focus on specific parts of the input text, improving its ability to preserve meaning, especially in sentences where the syntax structure changes significantly between the source and target languages.

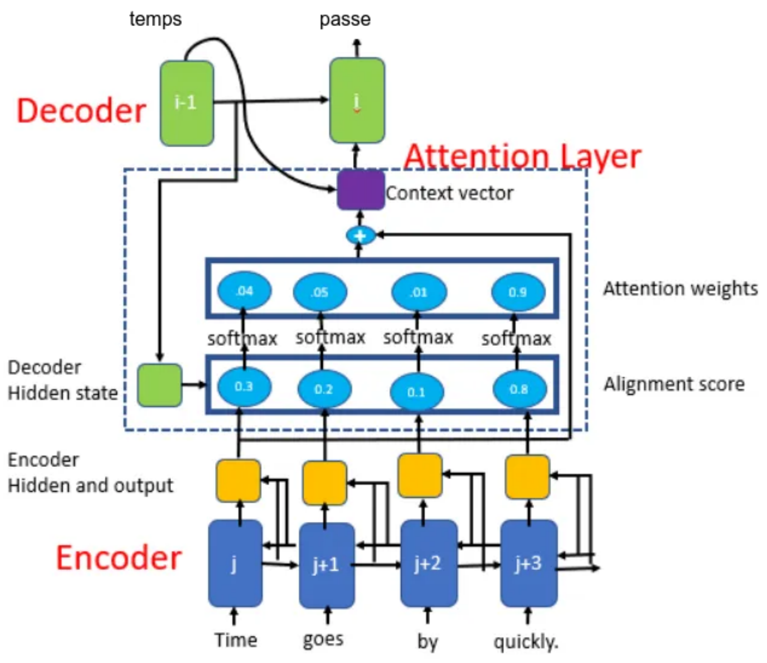


FIGURE 3 – Bahdanau Attention Mechanism - medium.com

This approach represents a more advanced stage in our exploration of neural machine translation techniques, promising improvements in translation quality for more complex and varied sentence structures.

### 4.3.2 Experimental Setting

The architecture of this model consists of three main components :

— **Encoder :** The encoder uses a GRU (Gated Recurrent Unit) layer to process the input text. We implemented encoder's GRU layer is designed with 1024 units, offering a substantial capacity to handle complex sentence structures.

— **Attention Mechanism (Bahdanau Attention) :** It consists of two Dense layers (each with the same number of units as the GRU layer, i.e., 1024) and a third Dense layer with a single unit. The attention mechanism computes a context vector for each timestep in the decoder, dynamically weighing the importance of different parts of the input sequence.

— **Decoder :** The decoder also employs a GRU layer with 1024 units. The decoder's output is then passed through a Dense layer to predict the next word in the target language.

The model uses an Adam optimizer. We trained the model on 30epochs with a batch size of 64, using an embedding dimension of 256 for both the encoder and the decoder. Checkpoints are saved every 2 epochs to preserve the model's state during training.

### 4.3.3 Results

While experimenting with the model, we were confronted with a significant challenge in terms of training time and memory usage. The model's complexity, combined with the large size of the dataset, led us to limit the training to 50 000 sentences.

The best results we obtained with this model were with the following parameters :

| **Loss Function** | Sparse Categorical Crossentropy |
|---|---|
| **Optimizer** | Adam |
| **Epochs** | 30 |
| **batch size** | 64 |
| **Dataset size** | 50 000 sentence |
| **max sequence length** | 20 |
| **vocabulary size** | 5 000 |

Table 5 – Parameters of the Baseline Encoder Decoder + Attention Mechanism

We also tried to train the model on the whole vocabulary but the results were slightly worse.

We wanted to have some insights about the BLEU score of our model based on the length of the sentences, so we calculated the BLEU score for each length range, and we also calculated the number of samples for each length range.

Here are the results we obtained :

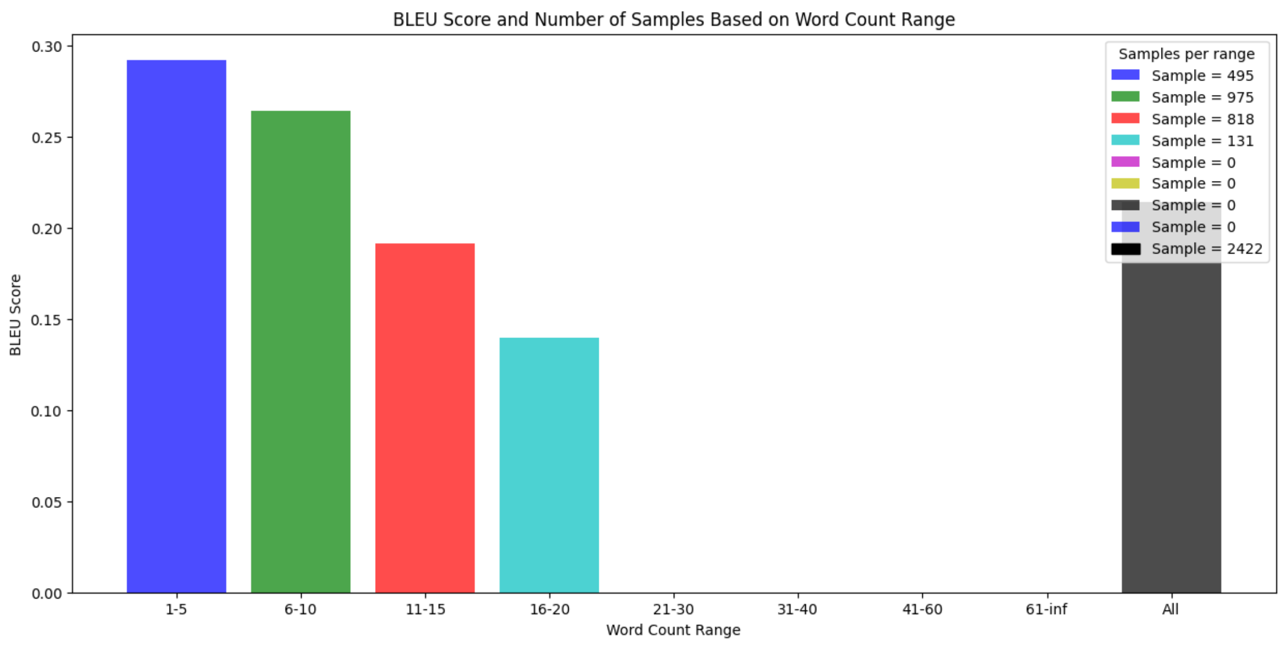


Figure 4 –

| **Length Range** | **BLEU Score** | **Number of Sentence (samples)** |
|---|---|---|
| (1 - 5) | 0.2918 | 495 |
| (6 - 10) | 0.2640 | 975 |
| (11 - 15) | 0.1917 | 818 |
| (16 - 20) | 0.1398 | 131 |
| **Overall Dataset** | **0.2140** | **2422** |

Table 6 – BLEU Scores and Dataset Sizes

We can see that the model performs better on shorter sentences, and the BLEU score decreases as the length of the sentences increases. As we only used uni-directional GRU layers, we think that the model has a hard time capturing the context of long sentences.

## 4.4 Baseline Encoder Decoder Embedding Bidirectional with Attention Mechanism

### 4.4.1 Strategy

Now that we enhanced our baseline model by adding a unidirectional Encoder-Decoder and Attention Mechanism, we wanted to further enhance the model's capability by implementing a bidirectional variant. This decision was inspired by a tutorial on TensorFlow's official website (`https://www.tensorflow.org/text/tutorials/nmt_with_attention?hl=fr`). Our objective was to explore whether a bidirectional approach combined with the encoder-decoder + attention mechanism approach could capture more comprehensive contextual information from the input sequences.

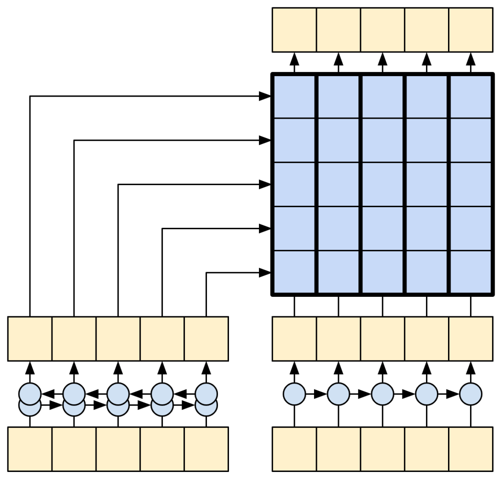


FIGURE 5 – Architecture of the model

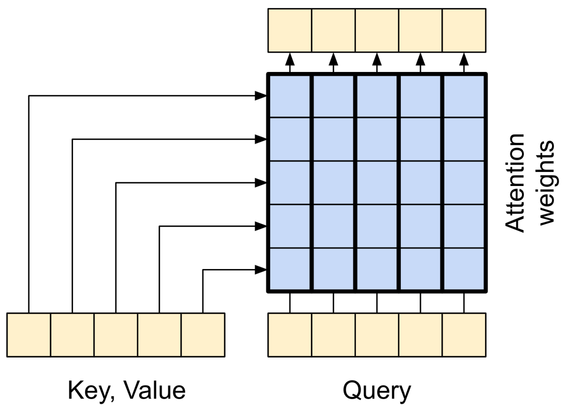


FIGURE 6 – The attention layer

### 4.4.2 Experimental Setting

The architecture of our advanced model comprises three important components :

— **Bidirectional Encoder :** The encoder uses an Embedding layer and a Bidirectional GRU layer. The embedding layer converts tokens into vectors of size 256 (embedding dimension). The bidirectional GRU, with 1024 units, processes these vectors from both directions. This architecture aims to capture a richer representation of the source text.

— **Cross Attention Mechanism :** We employed a CrossAttention layer with a MultiHeadAttention mechanism. This layer focuses on different parts of the encoded input when generating each word in the translation, allowing the model to capture nuances and dependencies.

— **Decoder :** The decoder, similar to the encoder, utilizes a GRU layer with 1024 units. It receives the context vector from the attention layer and generates the output sequence. The decoder is designed to work in tandem with the CrossAttention mechanism, using the context provided to produce more accurate and contextually relevant translations.

The model is trained using an Adam optimizer and the sparse categorical cross-entropy loss function. We set the batch size to 64 and trained the model for 30 epochs, saving checkpoints every 2 epochs.

### 4.4.3 Results

For this model the best results we obtained were with the following parameters :

| | |
|---|---|
| **Loss Function** | Sparse Categorical Crossentropy |
| **Optimizer** | Adam |
| **Epochs** | 100 |
| **batch size** | 64 |
| **Dataset size** | ALL |
| **vocabulary size** | 5 000 |

TABLE 7 – Parameters of the Baseline Encoder Decoder + Attention Mechanism

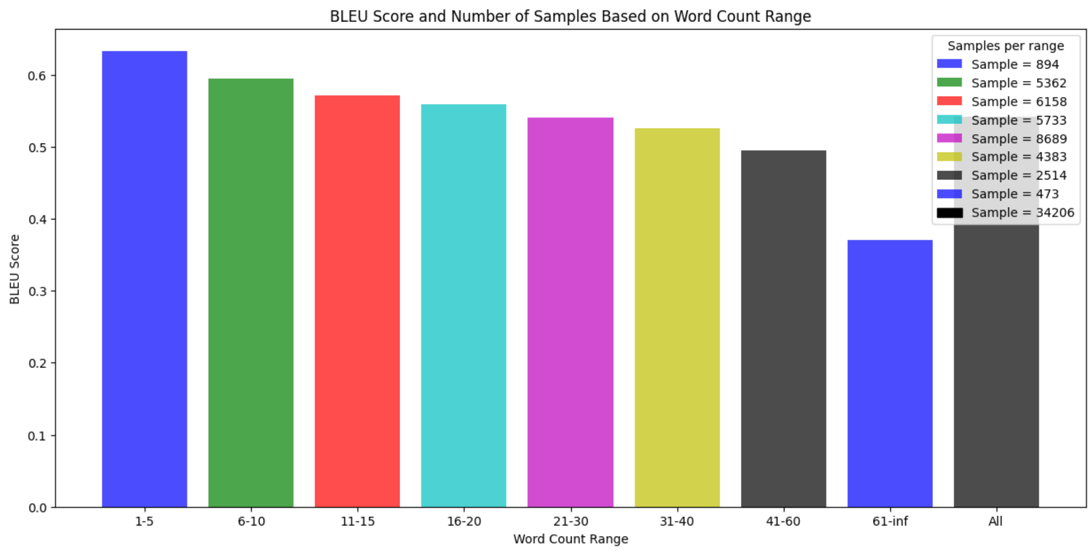


FIGURE 7 –

| Length Range | BLEU Score | Number of Sentence (samples) |
|---|---|---|
| (1 - 5) | 0.6323 | 894 |
| (6 - 10) | 0.5945 | 5362 |
| (11 - 15) | 0.5718 | 6158 |
| (16 - 20) | 0.5589 | 5733 |
| (21 - 30) | 0.5403 | 8689 |
| (31 - 40) | 0.5258 | 4383 |
| (41 - 60) | 0.4944 | 2514 |
| (61 - inf) | 0.3706 | 473 |
| **Overall Dataset** | **0.5416** | **34206** |

TABLE 8 – BLEU Scores and Dataset Sizes

As we can see the BLEU score is better than the one we obtained without the bidirectional encoder, but it is still not very high. Moreover, the model perform very well on long sentences, compared to our previous models. We think that to further improve the model it could be great to add a **postional encoding layer** to the model, to help it capture even more the context of the sentences.

## 4.5 Transformer T5 Model

### 4.5.1 Strategy

In our continuous pursuit of improving translation quality, we decided to experiment with the Transformer T5 model. T5, or "Text-to-Text Transfer Transformer", is known for its versatility and efficiency in various natural language processing tasks, including translation. Our goal was to use the advanced capabilities of T5 to enhance the accuracy and fluency of our translations.

### 4.5.2 Experimental Setting

The T5 ("Text-to-Text Transfer Transformer") model represents a significant advancement in the field of neural machine translation. Here are key aspects of its formation and structure :

— **Architecture :** T5 is built on the Transformer architecture, which is characterized by its use of self-attention mechanisms. The model is composed of an encoder and a decoder, each with multiple layers.

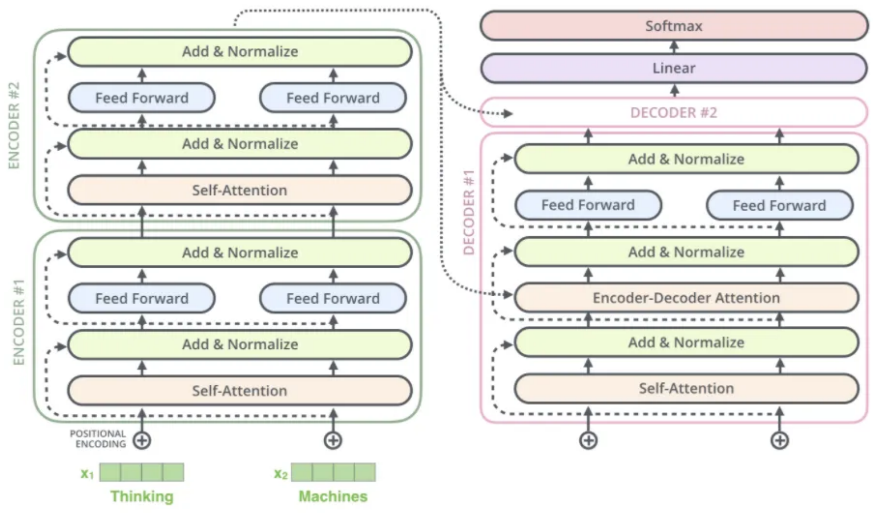
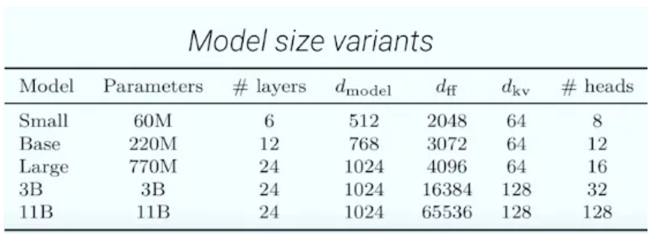


FIGURE 8 – T5 Architecture - medium.org

The "t5-small" variant, which we used, has a smaller number of layers and parameters compared to larger versions.


*Model size variants*

| Model | Parameters | # layers | $d_{model}$ | $d_{ff}$ | $d_{kv}$ | # heads |
|---|---|---|---|---|---|---|
| Small | 60M | 6 | 512 | 2048 | 64 | 8 |
| Base | 220M | 12 | 768 | 3072 | 64 | 12 |
| Large | 770M | 24 | 1024 | 4096 | 64 | 16 |
| 3B | 3B | 24 | 1024 | 16384 | 128 | 32 |
| 11B | 11B | 24 | 1024 | 65536 | 128 | 128 |

FIGURE 9 – T5 Paper - https ://arxiv.org/pdf/1910.10683.pdf

— **Layer Details :** Each layer in the T5 encoder and decoder consists of a multi-head self-attention mechanism followed by a feed-forward neural network. Layer normalization and residual connections are employed throughout the model, enhancing training stability and model performance.

— **Training Setup :** For our specific translation task, we utilized the Hugging Face Transformers library to fine-tune the "t5-small" model. The training process involved customizing the Seq2Seq training arguments, including setting an appropriate learning rate, batch size, and the number of training epochs. We also employed mixed-precision training to optimize the training efficiency.

Through fine-tuning the T5 model on our translation task, we aimed to achieve high-quality translations.

### 4.5.3 Results

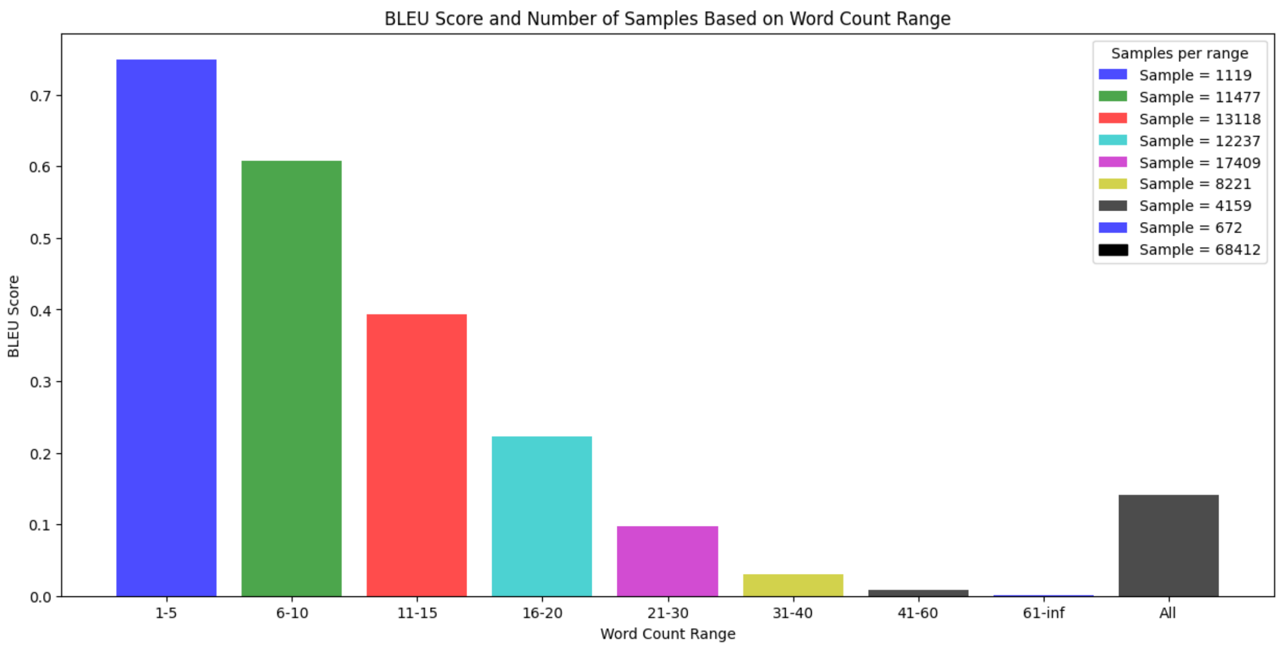


FIGURE 10 – Enter Caption

| Length Range | BLEU Score | Number of Sentence (samples) |
|---|---|---|
| (1 - 5) | 0.7486 | 1119 |
| (6 - 10) | 0.6072 | 11477 |
| (11 - 15) | 0.3938 | 13118 |
| (16 - 20) | 0.2228 | 12237 |
| (21 - 30) | 0.0967 | 17409 |
| (31 - 40) | 0.0309 | 8221 |
| (41 - 60) | 0.0078 | 4159 |
| (61 - inf) | 0.0016 | 672 |
| **Overall Dataset** | **0.1405** | **68412** |

TABLE 9 – BLEU Scores and Dataset Sizes

This pre-trained model yielded the best results so far. However, the model's performance drop sudenly for long sentences. We think that this is due to the fact that the model was pre-trained on a specific architecture, and that the architecture of the model is not adapted to our translation task. Moreover, we found out that many pre-trained models have a maximum token limits, so maybe the model was not able to process long sentences because of this limit.

## 4.6 Experimentation with Torch Transformer Model

### 4.6.1 Strategy

In the continuity of our experimentation, we decided to try another transformer. After using a pre-trained model (T5) we thought that experimenting a custom Transformer model built with PyTorch that will be trained on our data, would be interesting. This would also provide us with a deeper understanding of the inner workings of the Pytorch transformer.

### 4.6.2 Experimental Setting

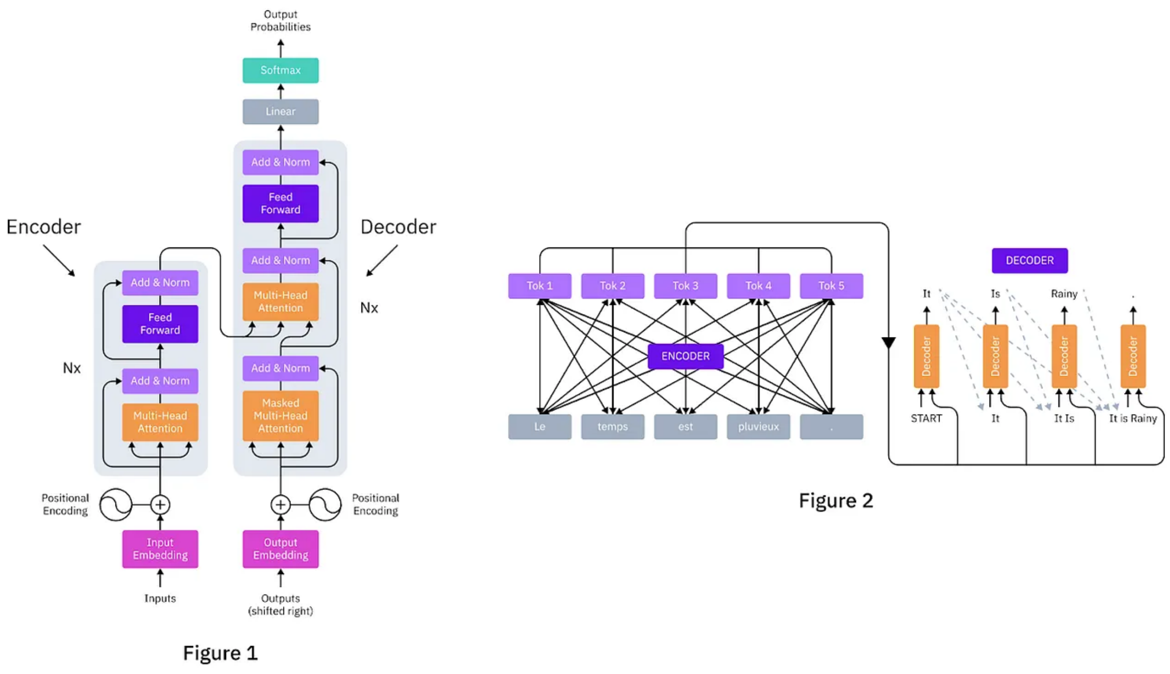


Figure 11 – Pytorch Transformer Architecture - Paul Gresia - pgresia.medium.com

— **Encoder :** The encoder consists of 3 'TransformerEncoderLayer' instances. Each layer contains a multi-head attention mechanism ('MultiheadAttention') with an output feature size of 192. This mechanism allows the encoder to process different parts of the input sequence in parallel, capturing a broad range of contextual information. The layers also include two linear transformation layers ('Linear') with 192 features, followed by dropout layers ('Dropout') with a dropout rate of 0.1, and layer normalization ('LayerNorm'). These components work together to refine and stabilize the representation of the input sequence.

— **Decoder :** The decoder is similarly structured with 3 'TransformerDecoderLayer' instances. Each of these layers is equipped with a multi-head attention mechanism ('MultiheadAttention'), also featuring an output dimension of 192. This multi-head attention mechanism in the decoder is crucial for focusing on relevant segments of the encoded input while generating the output sequence. Similar to the encoder, each decoder layer includes two linear transformation layers with 192 features, accompanied by dropout layers with a dropout rate of 0.1, and a normalization layer. Additionally, the decoder incorporates cross-attention modules within each layer to interact with the encoder's output and refine the generated output.

— **Positional Encoding :** The PositionalEncoding module is utilized to embed positional information into the input sequences. By incorporating sinusoidal functions, it imparts a sense of order to the model, enabling it to consider the sequential relationships between tokens. This is crucial for tasks like machine translation where the position of words in a sentence is vital for understanding meaning.

### 4.6.3 Results

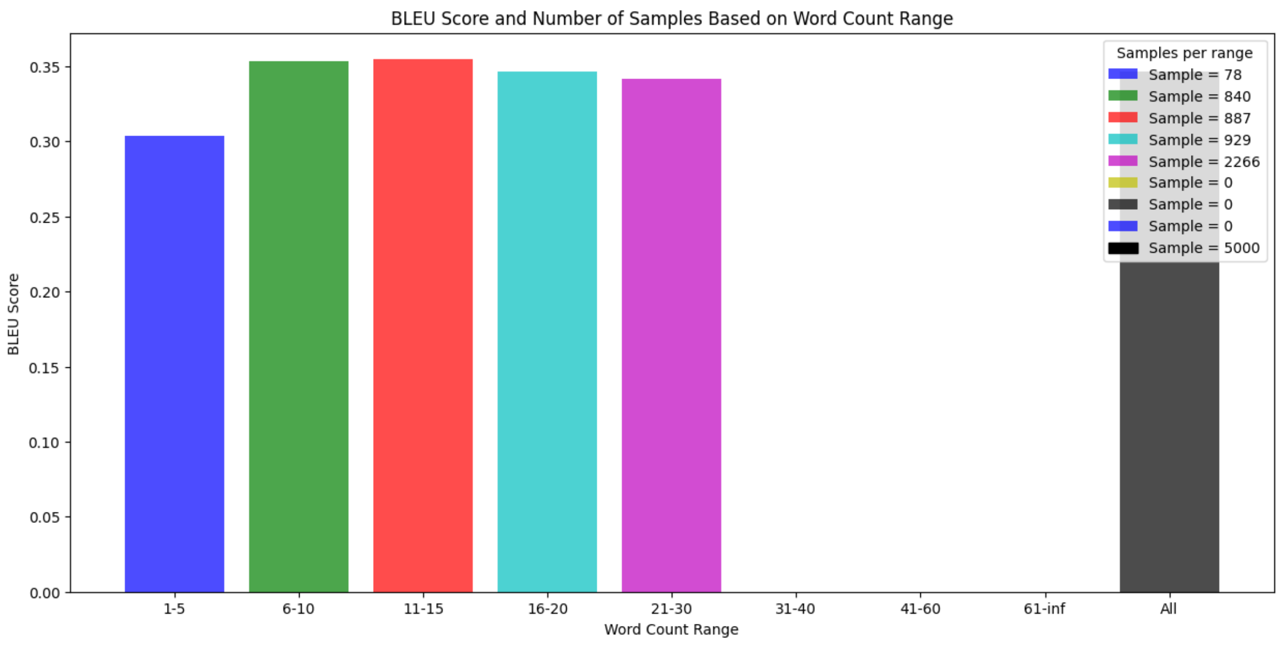


FIGURE 12 –

| Length Range | BLEU Score | Number of Sentence (samples) |
|---|---|---|
| (1 - 5) | 0.3034 | 78 |
| (6 - 10) | 0.3532 | 840 |
| (11 - 15) | 0.3545 | 887 |
| (16 - 20) | 0.3467 | 929 |
| (21 - 30) | 0.3416 | 2266 |
| **Overall Dataset** | **0.3466** | 5000 |

TABLE 10 – BLEU Scores and Dataset Sizes

The BLEU scores show a relatively consistent performance across different length ranges. There is no drastic drop in performance for longer sentences, indicating that the PyTorch Transformer model handles varying sentence lengths reasonably well. We don't have results for sentence with more than 30 words due to memory limitation. Comparing with the T5 model results, the PyTorch Transformer shows lower BLEU scores. However, it's essential to consider that the T5 model is pre-trained on a large corpus, while the PyTorch Transformer is trained specifically on our data. Further fine-tuning or architectural adjustments may enhance performance.

# 5 Project Management

## 5.1 Regular Meetings and Strategy Planning

We implemented a schedule of weekly calls. These meetings served as a platform for various critical activities :

— **Progress Review :** Each week, we reviewed the work completed, discussing any challenges encountered and sharing insights or learnings from our individual tasks.
— **Strategy Development :** These sessions were crucial for strategizing our next steps. We deliberated on which models to test next, considering both their theoretical potential and practical feasibility. These discussions ensured that our choices were aligned with our project goals and resource availability.
— **Task Allocation :** Based on our strategy discussions, we assigned tasks for the coming week, ensuring an equitable distribution of work and allowing each member to contribute to different aspects of the project.

## 5.2 Collaborative Work Approach

In our pursuit to explore various models and techniques in machine translation, we adopted a collaborative approach to working on the Jupyter notebooks :

— **Shared Responsibility :** Rather than dividing the project into segments, each team member contributed to all parts of the project.
— **Knowledge Sharing :** By working collectively on the notebooks, we were able to share knowledge and skills, enabling us to learn from each other and tackle complex problems more effectively.

# 6 Conclusion

In this project we tried to implement a series of experiments to implement models of text to text translations. We started with basic Seq2Seq models and tried to progress continuously to more sophisticated architectures, including models with bidirectional and attention layers, as well as advanced transformers and pre-trained models like T5.

Each model presented unique challenges and learning opportunities. Moreover, the process of developing models from scratch using provided invaluable learning experiences. Writing our own notebooks allowed us to delve deep into neural machine translation. For instance, attention mechanisms, positional encoding, encode/decoder and transformer, gaining a deep understanding of how these components contribute to the model's overall performance.

The combination of using pre-trained models and implementing models from scratch provided a well-rounded educational experience. It allowed us to bridge the gap between theoretical concepts and practical implementation, leading to a better understanding of deep learning for neural machine translation.

Throughout this project, we employed evaluation metrics such as the BLEU score to quantify the performance of our models. This metric, while useful, also revealed its limitations, emphasizing the need for comprehensive evaluation that combines both quantitative and qualitative measures.

However, it's essential to acknowledge the challenges faced during the project. Limitations in terms of memory, time constraints, and GPU usage became apparent, particularly when working with more complex models since it became difficult to train the models on the whole dataset. These constraints influenced our decision to reduce the dataset size and adjust certain parameters to make the experiments feasible within the available resources.

In conclusion, this project enhanced our understanding of current techniques in machine translation, especially in the field of natural language processing.

## References

— **Notebooks :**

1. **Hugging Face Translation Notebook :** GitHub
2. **Fireblaze AI School Blog :** Fireblaze AI School
3. **Luis Roque GitHub :** GitHub
4. **Neural Machine Translation Tutorial :** GitHub
5. **English-French NMT :** GitHub

— **Transformers :**

1. **Hugging Face T5 :** T5-Transformer
2. **Vishal Karda GitHub :** GitHub
3. **Hugging Face Documentation :** Documentation
4. **PyTorch Transformer Tutorial :** PyTorch Tutorial
5. **Transformer Paper :** ArXiv
6. **PyTorch Transformer Tutorial 2 :** PyTorch Tutorial
7. **Making PyTorch Transformer Twice as Fast :** Medium
8. **Detailed Guide to PyTorch's nn.Transformer Module :** Towards Data Science

— **TensorFlow :**

1. **Hugging Face Translation TensorFlow Notebook :** GitHub
2. **TensorFlow NMT with Attention :** TensorFlow Tutorial
3. **TensorFlow Transformer Tutorial :** TensorFlow Tutorial

— **Attention :**

1. **Bahdanau Attention Blog :** Medium
2. **Attention Is All You Need Paper :** ArXiv
3. **Harvard NLP Attention Blog :** Harvard NLP