Université Jean Monnet
Faculté des Sciences et Techniques
Master Informatique
Parcours Machine Learning and Data Mining
Cohort 2022-2024

Deep Learning
Project

Machine Translation: French to English

Students :
Hedi Zéghidi
Mohamed Moudjahed
Erick Gomez

Professor :
Amaury Habrard
Sri Kalidindi

Contents

1 Introduction

Transformer research is dedicated to three main areas: architecture modification, pre-training methods, and applications (Sutskever et al. [2014]). In this report, we focus on architecture modification and fine tuning. The application area is French to English translation. First, we train LSTM and GRU architectures. Next, we train different variations of a simplification of the original transformer (Vaswani et al. [2017]). Finally, we fine tune the MarianMT model (Wolf et al. [2020]).

All the models are trained on the same dataset (Global Voices). After removing repeated and wrongly classified sequences, we obtain 329 thousand pairs phrases, divided in training (80%), validation (10%) and testing (10%) sets, we also added a second dataset to add more samples to each set (4 thousand sentences to the validation, and testing sets, and 40 thousand to the training set) from the website `https://opus.nlpl.eu/wikimedia.php`. In Section 1.3 we present a more detailed overview of the data.

The strategy we set in the experiments has the following guidelines. In the case of the GRU and LSTM, the goal is to obtain the best possible translation, given that these architectures are far from state of the art. Regarding the basic transformer, we have two analysis goals: 1) the performance of models with reduced expressive capacity in comparison with the base model, and 2) the impact of the inclusion of a more diverse train data in a model with reduced expressive capacity. Finally, for the MarianMT fine tuning, we explore the performance of models where we increase the number of layers with trainable parameters.

In conclusion, an increase in training data using the Wikimedia sequences does not improve the performance of less expressive transformer models. The smallest transformer model is as capable as the baseline transformer. Global Voices concentrates mainly in news, and its lack of diversity is candidate to explain why smaller transformer models have a comparable performance to the base transformer. Fine tuning the MarianMT models reveals negative transfer. However, if we must choose a tuning strategy, it is better to increase the learning rate as the model approaches the final layer.

1.1 Overview of the dataset

The Global Voices collection is a diverse linguistic dataset sourced from citizen media content found on Global Voices websites. It consists of 302.6K document pairs and 8.36M segment alignments, spanning across 756 language pairs. Within the PGV collection, there is a specific dataset known as Parallel Global Voices EN-FR, focusing on English-French language pairs.

This dataset originates from the Global Voices consortium, a collaboration of websites where volunteers actively contribute and translate news content in over 40 languages. The Global Voices consortium serves as an international community comprising bloggers and citizen journalists dedicated to reporting on and translating various forms of citizen media on a global scale.

Their primary objectives include amplifying voices from marginalized communities and advocating for free expression. Comprising 1,000 document pairs and 25,000 sentence pairs, this corpus serves as a valuable asset for researchers and practitioners exploring the

nuances of the English-French language pair.

The content originally presented in Prokopidis et al. [2016], made available under a Creative Commons Attribution license, was collected in July-August 2015 by the NLP group at the Institute for Language and Speech Processing. The methodology utilized for extracting parallel content from multilingual websites, with a particular emphasis on the Global Voices sites, involved the application of the following techniques:

1. Focused Crawling: This involved using a focused crawler to automatically acquire domain-specific monolingual and bilingual corpora from the web. The targeted domains were specific to citizen media stories from the Global Voices websites.

2. Document Pair Detection: Document pairs were identified by leveraging the website graph. This method involved recognizing pairs of web pages connected by specific links indicating that one page is a translation of the other. The analysis focused on the website's structure and identifying links signifying translation relationships.

3. Language-Independent Methods: The aim was to test methods that are not language-specific and don't rely on the unique properties of the Global Voices website. The methods employed did not utilize language resources like bilingual dictionaries or machine translation engines. Instead, they focused on co-occurrences of images with the same filename in HTML source, edit distance of sequences of digits in the main content of webpages, and structural similarity.

The translation dataset is composed of 342,060 samples and contains the columns: 'fr' for the original French text and 'en' for the corresponding English translation. Each entry represents a translation pair, enabling the exploration and creation of machine learning models designed for translating French to English text.

1.2 Cleaning of the dataset

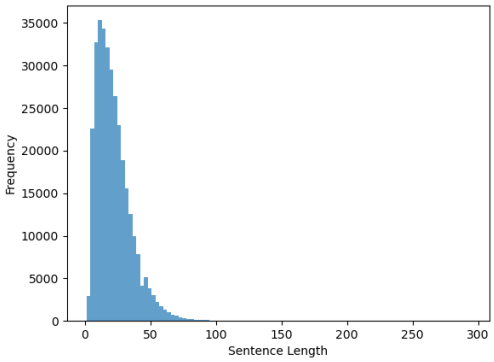
When we analyse the dataset, we observed that certain pair of sentences was wrongly assigned such as the ID 113 with "He concludes, however:" and "Which one do we choose?" or two sentences following each other in english such as the ID 103 with "He was responding to a question posted on Sudan Watch:" and "Is Sudan an African or Arab country? In other words, are we Sudanese, African or Arab?".

Other times we have the same sentence assigned in the two columns but always wrongly such as the sentence "Should we leave these children grow up in institutions simply because they're from a different race?" which appeared in the ID 147 in the column en but is pair with "Qui sommes nous pour juger ? Enfin , Mimz pense que la St-Valentin, c'est niais! :" and also in the ID 143 in the french column paired with "I feel Sudan doesn't have one! On the other hand, Nomadic Thoughts blogged something slightly different.".

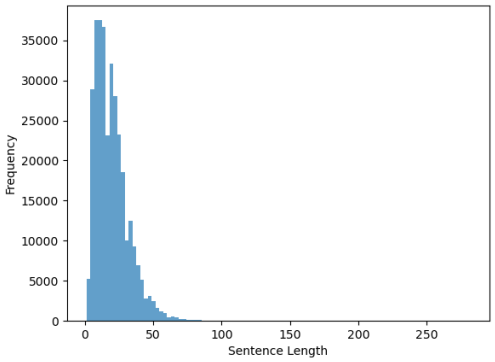
To detect these errors, we used a model to predict the traduction and we compared the prediction and reality with a meteor score, this metrics will be explained later in the report. We then set a threshold to differentiate the correct pairs to the wrong pairs, here we decided to set the threshold at 0.15. We also clean the dataset by deleting the pairs containing hyperlinks such as ID 82332. After the cleaning the dataset, we have 329434 samples, 11605 was detected as wrongly assigned sentences and 1021 with a hyperlinks.

1.3 Analysis of the dataset

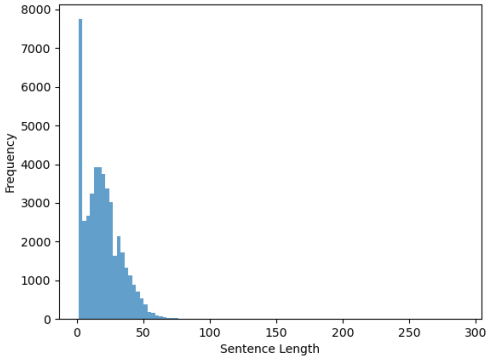
Following the dataset cleaning process, we proceed to analyze the content of both French and English sentences in the Global Voices compendium and the Wikimedia sequences. Initially, we examine sentence lengths, as illustrated in Fig. 1. The analysis reveals that the majority of sentences in the French dataset range from 10 to 50 words, a pattern mirrored in the English dataset. However, the Wikimedia set has a higher proportion of short sequences. Upon calculating average sentence lengths, we find that French sentences average 22 words, while English sentences average 19 words. And for the dataset Wikimedia, we have also a average sentence between 5 and 30 word for the french sentences and english sentences.



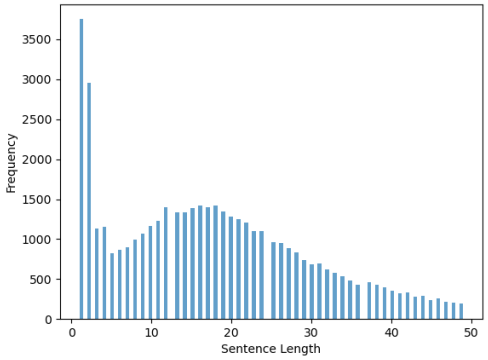
(a) French PGV



(b) English PGV



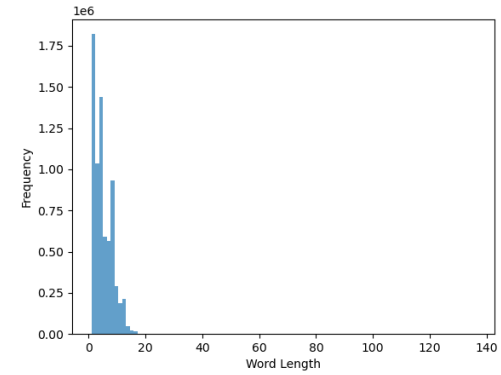
(c) French Wikimedia



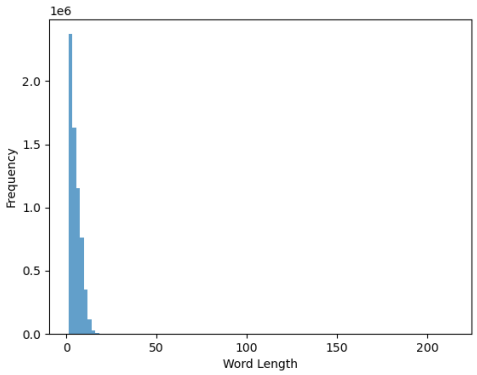
(d) English Wikimedia

Figure 1: Sentence length

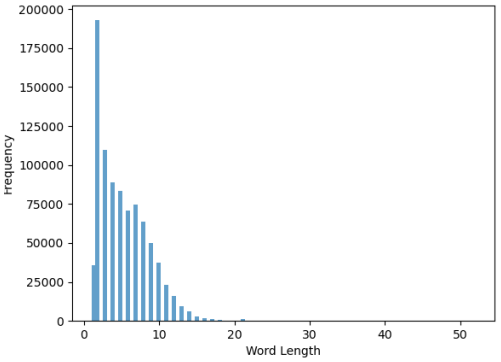
In the Fig.2, we have the word length for the two languages of the dataset and it reveals that the french and english word varies between 1 and 20 letter and the average word has 5 letter for the french and english. The biggest and shortest word are "ignorez-simplement-ce-que-l'essai-dit-il-est-seulement-destiné-à-la-consommation-intérieure-et-contient-beaucoup-de-phrases-cryptées-qui"and "a" for the french sentences and "understood-in-the-context-of-Japanese-election-propaganda-and-in-Japan-nobody-believes-anything-printed-especially-when-the-author's-stringing-together-of-platitudinous-utterances-makes-him-sound-like-he-is-stoned" and "a" for the english sentences.



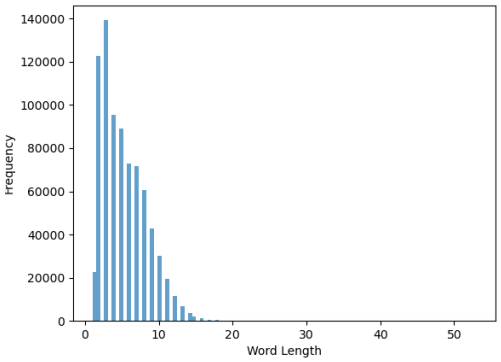
(a) French Global Voices



(b) English Global Voices



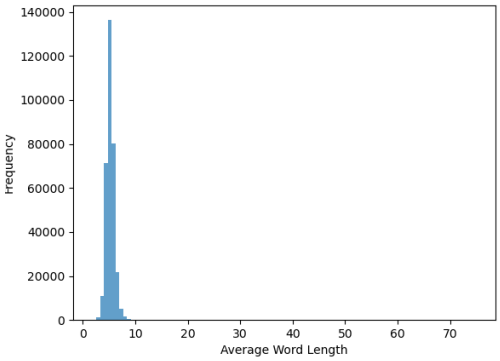
(c) French Wikimedia



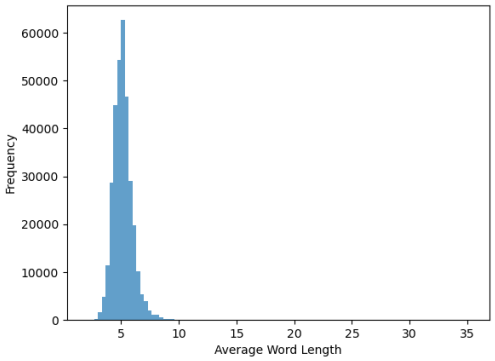
(d) English Wikimedia

Figure 2: Word length

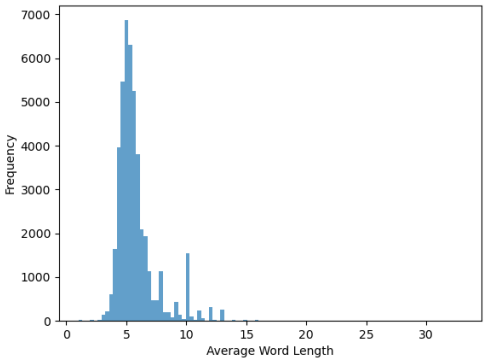
Now, we analyze the average word length by sentence in the Fig. 3, we have in average 5 letter words in each sentence for French and English sentences.The longest sentence for French and English are the same pair and are a poem written by Chinese poet, filmmaker and artist Hung Hung for the protest of Taiwan in 2014 which begin by "Nous arrivons, quand l'été arrive Nos pas peuvent être doux, nos pas peuvent être fermes [...]" with 294 words in french and "We are coming, when the summer is coming Our footsteps can be soft, our footsteps can be firm Our sounds can be beautiful, our sounds can be hoarse [...]" with 282 words in English.



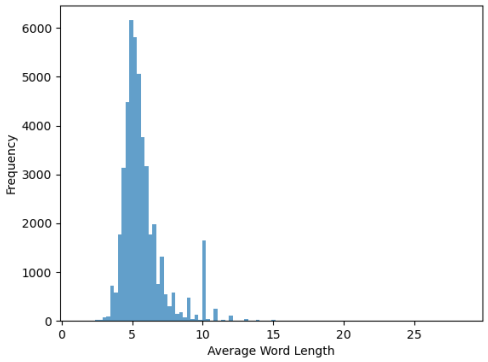
(a) French Global Voices



(b) English Global Voices



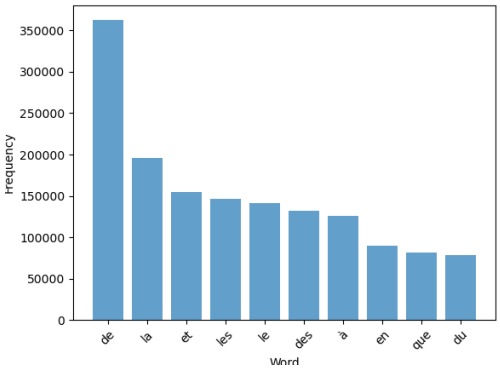
(c) French Wikimedia



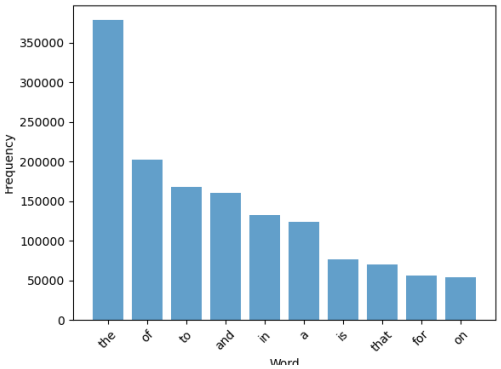
(d) English Wikimedia

Figure 3: Average word length by sentence

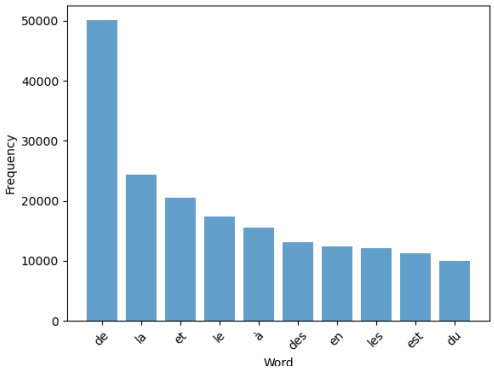
When we look at the 10 most frequent word in the dataset PGV and Wikimedia (as in the Fig.4), the majority of them are the stop words of each languages, for the french sentences its "de", "la", "les", "le", "des", etc. and for the english sentences its "the", "of", "to", "a", "is", etc. For the French and English sentences, it suggests that these are common French and english articles and prepositions. These words are fundamental to constructing meaningful and grammatically correct sentences in French. However, to gain a more comprehensive understanding of the French and English corpus, we study the 10 most frequent word without the stopwords.



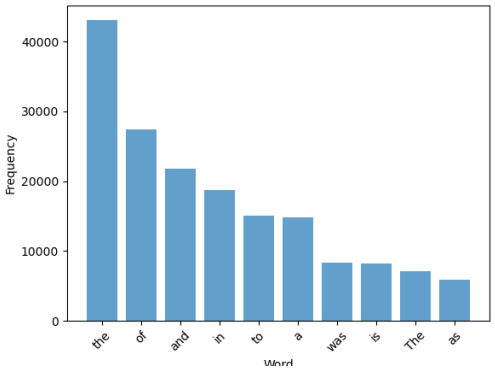
(a) French Global Voices



(b) English Global Voices



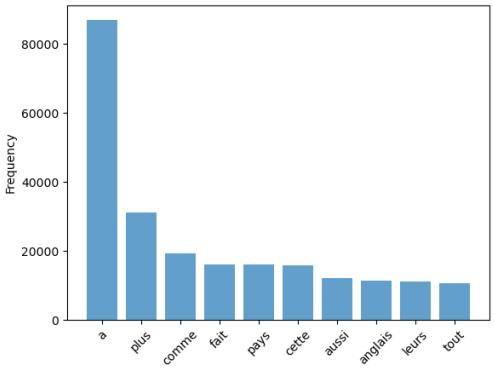
(c) French Wikimedia



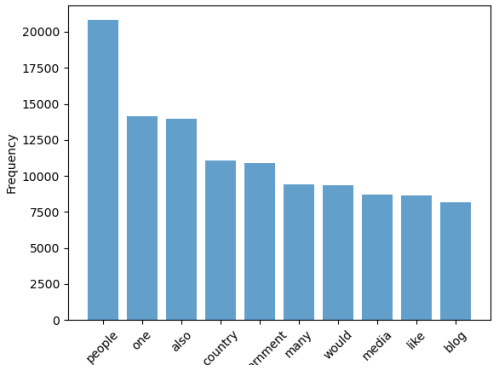
(d) English Wikimedia

Figure 4: Top 10 most frequent word

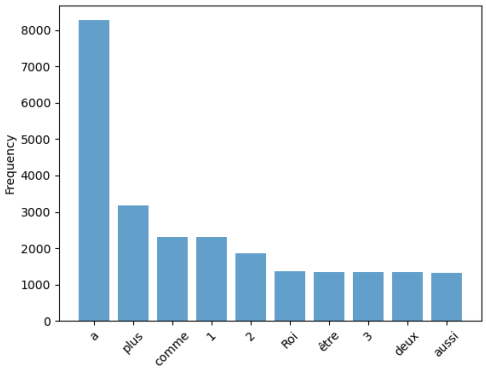
If, we don't count the stopwords for the dataset, we have the Fig.5 where the most frequent word are "a","plus","comme" for the french sentences and "people","one","also" for the english sentences. The majority of these words are adverb used in all the sentences by everyone, everyday.



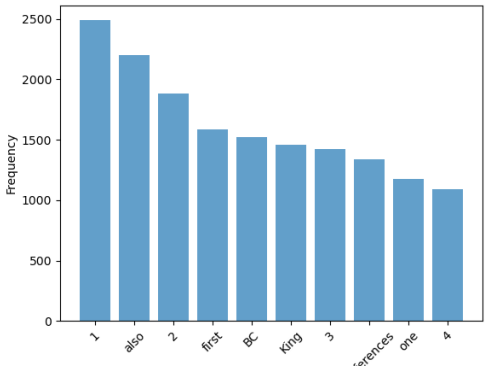
(a) French Global Voices



(b) English Global Voices



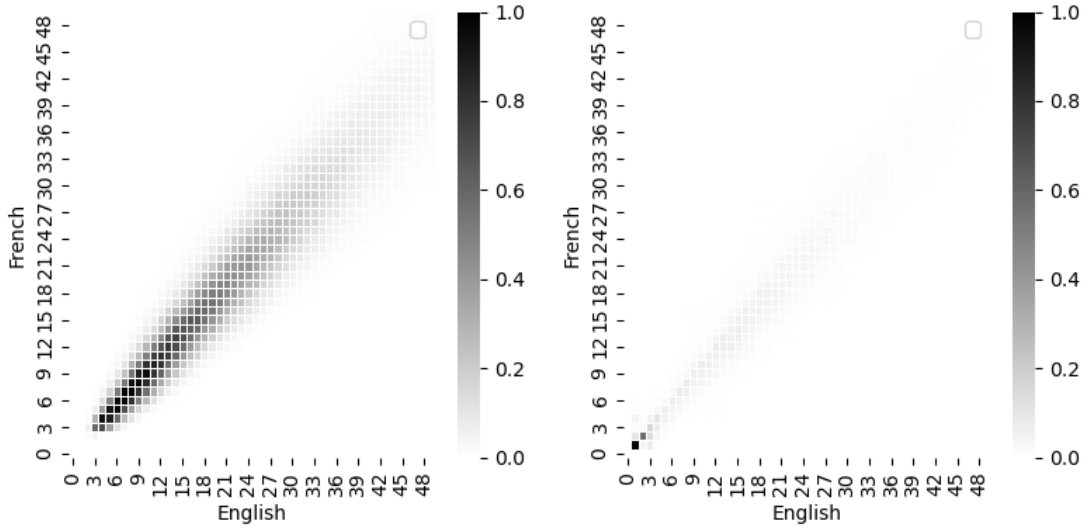
(c) French Wikimedia



(d) English Wikimedia

Figure 5: Top 10 most frequent word without stopwords

In the Fig.6, we observe that the number of words between the pair of french and english sentences is mostly equivalent and the sentence are concentrated between 6 and 20 words.



(a) Parallel Global Voices

(b) Wikimedia

Figure 6: Heatmap of number word for each pair of sentences

2 Literature Review

The sequence to sequence machine translation starts with Sutskever et al.. Their contribution departs from the prediction of labels in deep neural networks. They instead apply a "multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector". While the model presented in the paper has been successful, there are some limitations. Here are some potential disadvantages:

- Lack of Attention Mechanism. Attention mechanisms allow the model to focus on different parts of the input sequence when generating each part of the output, providing improved performance on long sequences.
- Inability to Capture Dependencies. The seq2seq model relies on a recurrent neural network (RNN) to capture dependencies between elements in a sequence. Long-term dependencies can be challenging for RNNs due to the vanishing gradient problem.
- Training Time and Computational Resources. The use of RNNs makes it less parallelizable compared to the Transformer architecture.

The lack of attention mechanism, the inability to capture long term dependencies, and its high computational cost limit the expressiveness of the encoder-decoder multilayered LSTM in comparison to the Transformer architecture proposed by Vaswani et al.. Their paper is the basis for the state-of-the-art machine translation models Hendy et al. [2023].

The models based on RNNs and LSTM present difficulties in treating long-range dependencies and issues in parallelization. The Transformers adopt the self-attention mechanism, instead of sequential processing, allowing the model to weigh different parts of the input sequence differently when making predictions.

The original Transformer Architecture consisted of an embedding of dimension equal to 512, 6 encoder layers, 6 decoder layers, each with a feed forward layer on a 2048 dimension, and 8 attention heads. Their architecture included a Resnet for each layer, and a normalization mechanism.

Another innovation in their paper was the positional encoding to convey the position of each element in the sequence by adding position dependent quantities to the embedding. The multi-head attention mechanism allows to slice the matrix representation of a sequence into multiple matrices, each with their own attention mechanism. Therefore, each head concentrates on a part of the representation.

3 Metrics chosen

The different metrics we used to evaluate our model are the BLUE score, METEOR score and ROUGE score:

1. The "BLUE" score refers to the "Bilingual Evaluation Understudy" score, a metric used to evaluate the quality of machine-translated text against one or more reference translations. It calculates the precision for each n-gram (contiguous sequence of

n items, typically words) in the candidate translation compared to the reference translations.BLEU has some limitations, such as not considering semantic meaning, fluency, or grammar directly.

2. METEOR (Metric for Evaluation of Translation with Explicit ORdering) is a machine translation evaluation metric that aims to comprehensively assess translation quality. Unlike metrics such as BLEU, METEOR considers unigram precision and recall, incorporates stemming to handle word variations, and explicitly accounts for word order and synonymy. The score ranges from 0 to 1, with higher scores indicating better translation quality. METEOR's multi-faceted approach makes it more robust in capturing various linguistic aspects, offering a more comprehensive evaluation of machine-generated translations in comparison to simpler metrics like BLEU.

4 Recurrent Neuron Network

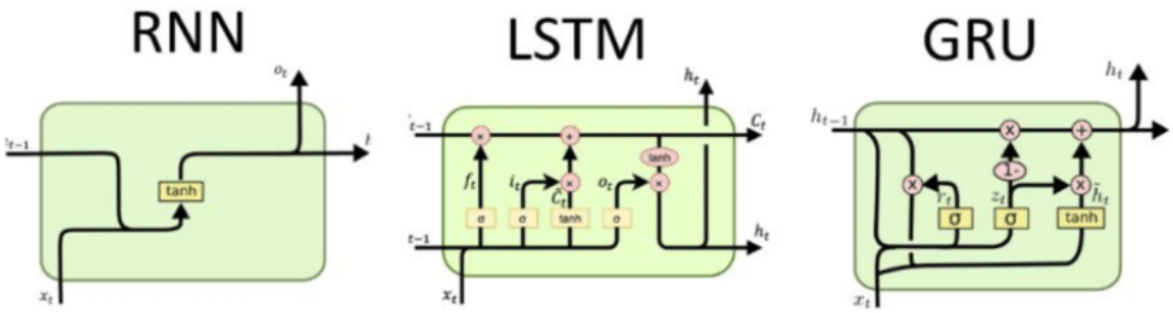


Figure 7: Structure of the RNN, LSTM and GRU networks taken from the "ResearchGate" website

4.1 RNN

RNNs are a class of artificial neural networks designed to recognise patterns in sequences of data such as text or sound. What makes them special is their ability to maintain a kind of 'memory' of previous entries in the sequence thanks to their recurrent connections. In text translation, RNNs are used to analyse sequences of words and understand the context and grammatical structure, producing a coherent translation in another language. However, traditional RNNs have limitations, notably the gradient vanishing problem, which makes it difficult to learn and memorise long dependencies in the data. The RNN diagram 7 shows a simplified network structure where each unit receives two inputs: the current input x_t and the hidden output of the previous step h_{t-1}. The hidden unit processes these inputs using an activation function, typically tanh, to calculate the current hidden output h_t which will be both the output of the RNN at this time step and the recurrent input for the next time step.

4.1.1 LSTM

The LSTMs (Long Short-Term Memory networks) are an evolution of RNNs designed to overcome the vanishing gradient problem. LSTM cells incorporate forget gates f_t,

input gates i_t, output gates o_t, as well as a cell state c_t. At each time step t, the LSTM processes the current input x_t, the previous hidden output h_{t-1}, and the previous cell state c_{t-1}. The various gates regulate the flow of information, allowing the cell state to maintain or forget information over time. The output h_t and the updated cell state c_t are computed at each step. The output h_t also serves as the output for the current time step. In the context of translation, LSTMs more efficiently handle long and complex sentences, retaining relevant contextual information. Therefore, they are better suited for translating texts with long-term dependencies, where conventional RNNs fail.

4.1.2 GRU

The GRU simplifies the LSTM architecture by combining the forget and input gates into a single update gate z_t, and using a reset gate r_t. Like the LSTM, the GRU receives x_t and h_{t-1} as inputs. The gates z_t and r_t determine how information is combined and updated to produce the new hidden output h_t, which is the output of the GRU for this time step and will be used in the calculation of the next time step. This simplification reduces computational complexity and training time, while maintaining performance comparable to that of LSTMs. GRUs are therefore an effective choice for tasks requiring both the ability to manage long-term dependencies and greater computational efficiency.

4.2 Introduction to the LSTM model

The model is a neural machine translation architecture based on an encoder-decoder system. It uses long-term memory with bidirectionality and incorporates an attention mechanism Bahdanau [2015] to improve translation quality. The Keras library was used to design the model, and the chosen architecture is quite well known and often used and explained on the net. *(The implementation of GRU follows largely the same approach as that of LSTM only the LSTM part will be explained)*

4.3 Preprocessing

1. **Data cleansing** The first step is to clean up the data. To do this, I remove all special characters and change all sentences to lower case.
2. **Sentence length reduction:** I then chose to keep only sentences of up to 25 words. This decision is made to simplify the training of the model by reducing the size of the data and avoiding the complications caused by sentences that are too long.
3. **Adding <sos> and <eos> tags:** For my output, I add <sos> (start of sentence) tags at the beginning and <eos> (end of sentence) tags at the end. These tags are crucial because they indicate the start and end of a sentence respectively. This allows the model to learn when to start and end a sentence.
4. **Tokenisation with Keras:** I use the Keras tokenizer to convert my sentences into sequences of numbers. The Keras tokenizer assigns a unique identifier to each unique word in the corpus so that I can work with numerical data.
5. **Transformation into sequences:** After tokenisation, I transform the sentences into sequences of numbers. Each word is replaced by its corresponding numerical identifier.

6. **Sequence padding:** Finally, I normalise the length of the sequences using padding. This means that I add zeros to the shorter sequences so that they reach a standard length, because the model requires all entries to have the same size.

4.4 Architecture and Technical Choices

1. **Creating the embedding layer:** The embedding layer transforms words into dense representation vectors. Embeddings enable efficient processing of textual data in neural networks because they reduce dimensionality, capture semantic relationships (similar words have similar vectors), improve model performance with a better understanding of the context and nuances of language and increase computational efficiency with more compact representations than one-hot representations . Two embedding layers have been designed, one for the encoder and another one for the decoder.

2. **Encoder layer construction :** The encoder uses a bidirectional LSTM structure. The choice of a bidirectional LSTM is motivated by its ability to understand the context in both directions (left and right) of the source text. This enables a richer context to be captured, which is crucial for understanding the whole sentence. A 20% dropout is applied for regularization, helping to prevent overfitting. As input to the encoder we will therefore have a sentence in French and as output the hidden state h_t and the state of the cell c_t.

3. **Decoder layer construction :** The decoder has as input the hidden state h_t and the state of the cell c_t of the encoder. The output of the decoder is the sentence with the <sos> tag added at the beginning.

4. **Attention mechanism :** The attention mechanism allows the decoder to focus on specific parts of the source sentence during translation. This approach is essential for handling long sentences, where each word in the output can be influenced by different words in the input. Attention improves translation accuracy by allowing the model to focus on the relevant elements of the input. This part is explained in detail in 4.4.1

5. **Construction of the rest :**We concatenate the context vector provided by the attention mechanics and representing the relevant part of the input with the vector representing the last word generated. We create an LSTM layer for the decoder then we add a Dense layer with softmax activation which allows us to choose the most probable word as output at each stage of the translation.

4.4.1 Attention mechanism, detailed explanation

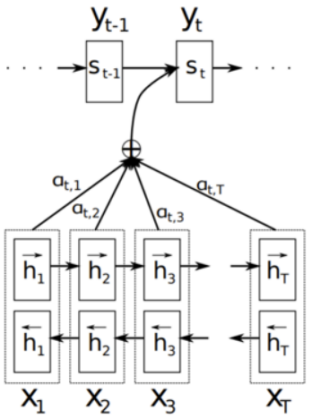


Figure 8: Diagram of the attention mechanism Han [2020]

1. **Encoder Hidden States:**
   In the lower part of the diagram, the vectors $X_1, X_2, \ldots, X_T$ represent the words of the source sentence that are passed through the encoder to produce hidden states, denoted by $\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_T}$ and $\overleftarrow{h_1}, \overleftarrow{h_2}, \ldots, \overleftarrow{h_T}$. These hidden states contain the encoded information of the source sentence.

2. **Decoder Hidden States and Word Prediction:**
   In the upper part, S_{t-1} is the previous hidden state of the decoder and S_t is the current hidden state. The decoder uses the state S_{t-1} and the previously predicted word Y_{t-1} to generate the prediction of the current word Y_t.

3. **Calculation of Attention Weights** $(\alpha_{t,1}, \alpha_{t,2}, \ldots, \alpha_{t,T})$**:**
   Each attention weight is calculated based on the relevance of the encoder's hidden state to the current hidden state of the decoder. These weights determine the importance of each word of the source sentence for predicting the current word of the target sentence. To calculate the attention weights, the hidden states of the encoder and decoder are combined and processed by two dense layers: one with 10 units and tanh activation for transformation, followed by another with a single unit and `softmax_over_time` activation to generate normalized attention scores.

4. **Formation of the Context Vector:**
   The context vector is the weighted sum of the encoder's hidden states, the weights being the attention values (α). The context vector thus contains the essential information of the source sentence necessary for predicting the current word.

5. **Use of the Context Vector:**
   This context vector is combined with the hidden state S_{t-1} and passed through a softmax layer to become the input S_t of the LSTM decoder for the next prediction.

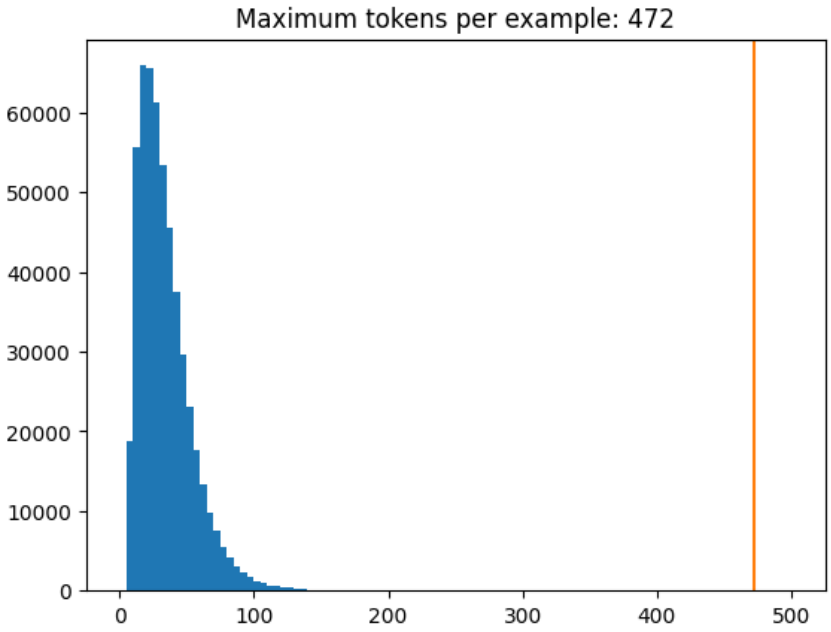


Figure 9: Number of tokens in each sequence (French or English)

5 Basic Transformer

In this section we explain the transformer we implement for sequence 2 sequence translation. The model has as base the proposal by Vaswani et al.. The main difference is the number of layers, we use 4 instead of 6 due to memory constrains, and the original paper uses the same tokenizer for source and target languages, we use different tokenizers. From the basic transformer, we develop several experiments where we change a component of the model (for instance a hyperparameter or the custom learning rate function). Below we detail the construction of the basic model and afterwards the different changes for the experiments.

5.1 Tokenizer

We implement the BERT (Bidirectional Encoder Representations from Transformers) tokenizer Devlin et al. [2018] for the English and French vocabularies separately. The tokenizer converts raw text to an embedding. BERT uses a WordPiece tokenizer, which breaks words into smaller subwords or pieces. The reserved tokens are PAD, UNK, START, and END. The first one adds tokens when the sequence is shorter than expected, the second one is used for unknown words, the last two signal the beginning and ending of a sentence. In the train data set, the size of the French vocabulary is 7652, and the English vocabulary is 7665. The bert tokenizer has a size of 8000, therefore the UNK token will not be used during training.

5.2 Architecture

The model has as input the source and target sequences embeddings. From there it adds the positional embedding to provide information about the absolute position of the tokens in a sequence:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

The positional encoding is different for odd and even positions. The Sine and Cosine functions helping to capture different patterns in the positional encoding via different frequencies. The *pos* term ensures that different positions get different values in the encoding. The division by 100002i/d100002i/d introduces a scaling factor that varies based on the position and dimension. It helps differentiate positional values across different dimensions.

5.2.1 Residual connection and Layer Normalization

This are two different layers applied after the feedforward, but also after the attention mechanism. The Residual Connection is the addition of the input (or residual connection) without any transformation and allows gradients to flow more easily during backpropagation and avoids vanishing or exploding gradients He et al. [2016]. The normalization layer replicates the usual treatment of the data before entering a neural network. The layer normalization step helps stabilize the activations within a layer. It normalizes the values along the feature dimension, reducing the internal covariate shift and making the optimization process more stable Ba et al. [2016].

5.2.2 Attention layer

The Attention allows a model to focus on different parts of the input sequence when making predictions or encoding information. Attention enables the model to selectively attend to relevant parts.

5.2.3 Feedforward Network

The feedforward netowork is comprised by two fully connected linear layers with a ReLU activation between them, and a dropout layer at the end. The feedforward network includes a add and normalization layer before its end.

5.2.4 Encoder and Decoder

The encoder is comprised by a stack of blocks. Each has a attention layer and a feedforward network. The output of the encoder feed every block of the decoder with keys and value. Each decoder layer includes a causal self attention, a cross attention, and a Feedforward network.

5.2.5 Transformer

The encoder and the decoder form the transformer. The final output of the decoder goes to a linear dense layer that feeds to a softmax layer used to make the predictions.

$$\text{CustomSchedule}(step) = \sqrt{d_{\text{model}}} \cdot \min\left(\frac{1}{\sqrt{step}}, \frac{step}{\text{warmup\_steps}^{-1.5}}\right) \quad (1)$$

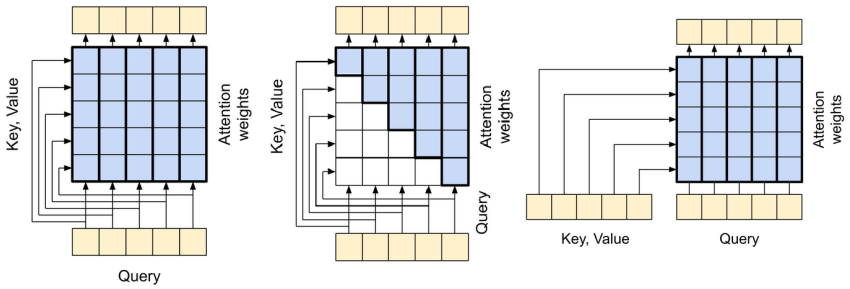


Figure 10: Left. General attention mechanism that processes the source sequence in the encoder. Center. Masked attention sequence that processes the target sequence at the start of the decoder, it is called Causal self attention. Right. Cross-attention mechanism that joins the information from each block of the enconder to each block of the decoder. Source: https://www.tensorflow.org/text/tutorials/transformer

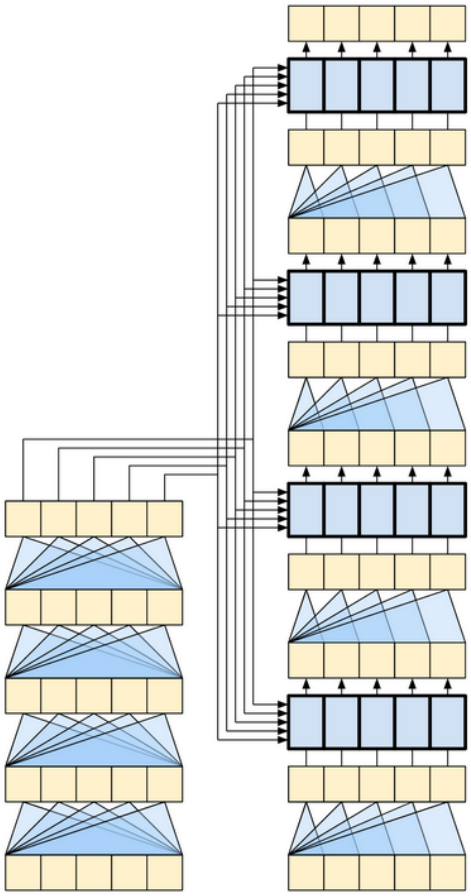


Figure 11: Schema of the basic transformer architecture with 4 layers in the encoder and 4 layers in the decoder. Source: https://www.tensorflow.org/text/tutorials/transformer

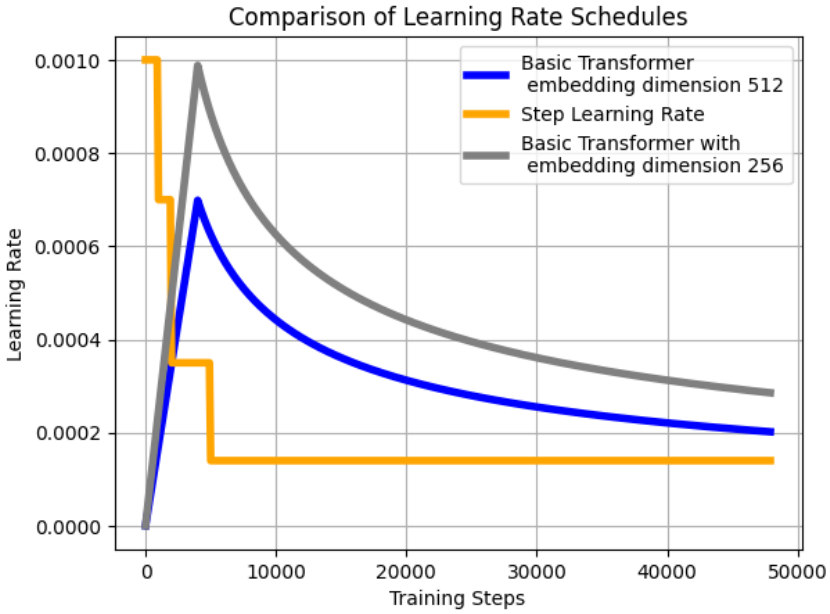


Figure 12: Custom learn schedule for the basic transformer model with embedding dimension of 512, an experiment with embedding dimension of 256, and decrease only as function of the number of training steps.

6 MarianMT

6.1 Overview

The model studied is the model "Helsinki-NLP/opus-mt-fr-en", as his name suggests it has been developed by the Helsinki-NLP team, often associated with the University of Helsinki, which is known for its contributions to natural language processing (NLP) research and development. The team has been involved in multiple projects and initiatives related to machine translation, multilingual models, and advancements in NLP.

This specific model uses the Marian NMT (Neural Machine Translation) model which is a framework developed by this team (Helsinki NLP). It is designed for training and deploying neural machine translation models. And as in the second part of the name suggests, this model has been trained on the dataset opus-mt between french and english.

OPUS is a hub of parallel corpora first published in Tiedemann and Nygaard [2004] and it's a parallel and free collection of translated documents from the internet, containing about 30 million words in 60 languages. Its goal is to give a open parallel corpora that uses standard encoding formats, including linguistic annotation and that can be freely used and distributed, allowing everyone to run experiments on bitexts and compare their results easily.

This version (0.2) of the OPUS corpus have about 30 million words in 60 languages, collected from three open source documentation:

- OpenOffice.org documentation (http://www.openoffice.org) contains about 2.6 mil-

lion words in six languages.

- KDE manuals including KDE system messages (http://i18n.kde.org) with 24 languages with about 3.8 millionwords in total
- PHP manuals (http://www.php.net/download-docs.php) containing about 20 million words in 60 languages

And the OPUS-MT Tiedemann and Thottingal [2020],Tiedemann et al. [2023] is a project that is part of the OPUS project, which aim is to democratize neural machine translation by providing open access to pre-trained models and tools for machine translation. It also focuses on increasing language coverage and translation quality by developing modular translation models and speed-optimized compact solutions for real-time translation on regular desktops and small devices.

6.2 Structure

The first step of the MarianMT model (Soliman et al. [2022]) is the preprocessing with the MarianMT tokenizer (as in the Fig.13). It utilizes SentencePiece, a library for unsupervised text tokenization. During training, SentencePiece learns a subword vocabulary from the provided corpus using techniques like Byte Pair Encoding (BPE) or unigram language modeling. The trained tokenizer can then segment input text into subword units and assign unique numerical IDs to each subword based on the learned vocabulary. The numerical IDs generated by the tokenizer serve as embeddings and are fed into the MarianMT model for further processing, contributing to the model's ability to handle diverse linguistic structures and improve translation performance.

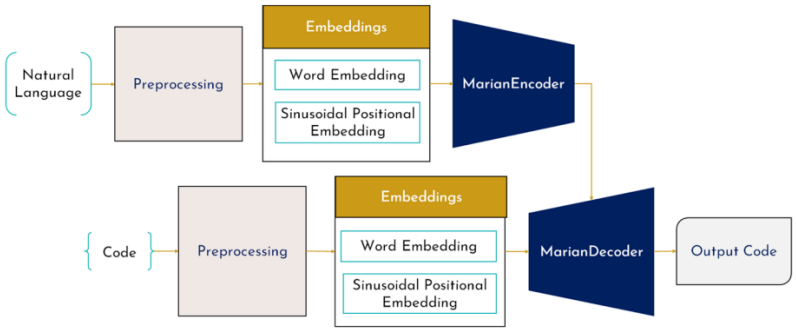


Figure 13: MarianMT architecture (image from the paper Soliman et al. [2022])

After the pre-processing, the embeddings are given to the encoder layer which consists of 6 layers with each having a self-attention mechanism, a Layer normalization after self-attention, a activation function (SiLU), two fully connected layers (fc1 and fc2) with output dimensions, and a final layer normalization.

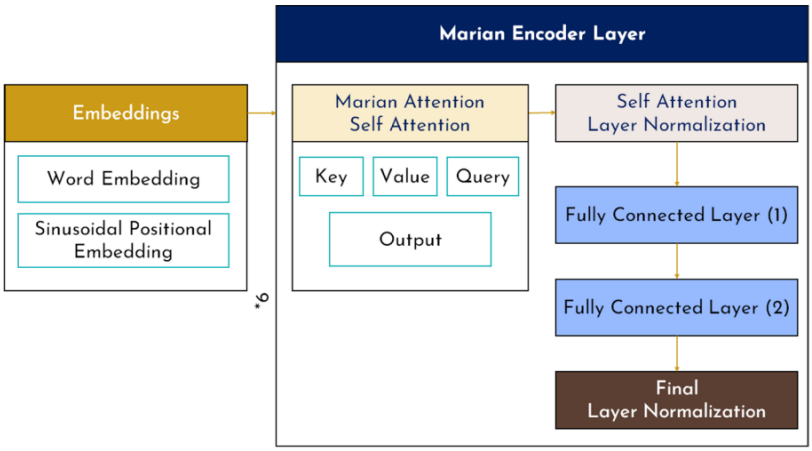


Figure 14: MarianMT encoder (image from the paper Soliman et al. [2022])

The decoder is also composed of 6 decoder layer with the same structure as the encoder layer (self-attention mechanism, a Layer normalization after self-attention, a activation function (SiLU), etc).

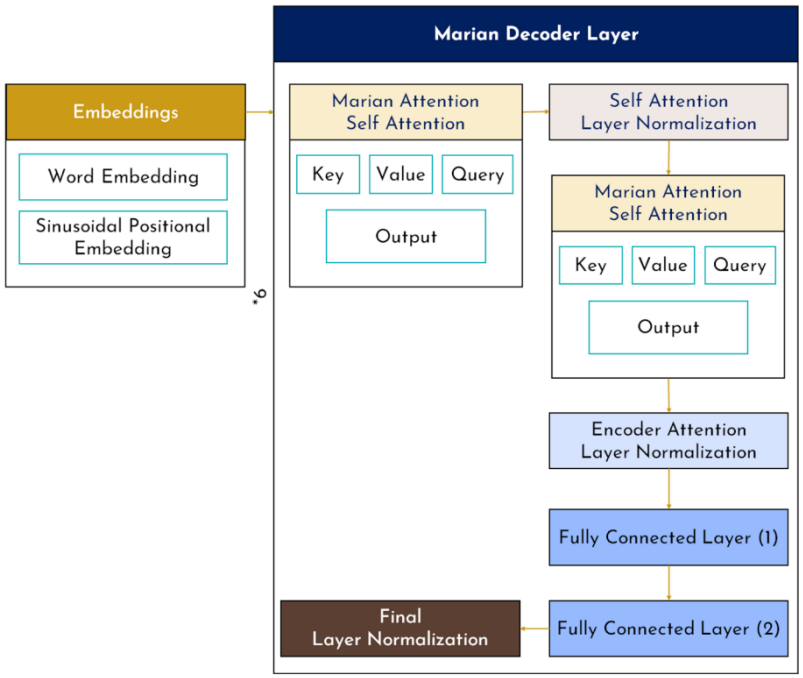


Figure 15: MarianMT decoder (image from the paper Soliman et al. [2022])

7 Experiments

7.1 GRU and LSTM

7.1.1 Strategy

To explore the different optimization strategies of recurrent neural network models, notably GRU and LSTM, we address several areas of improvement. It should be noted that these strategies are not exhaustive and other approaches could also be considered.

1. **Integration of Additional Data:** The integration of additional data from Wikipedia, in addition to that of Global Voice, aims to enrich the training dataset. This increase in diversity aims to improve the model's ability to generalize and understand more varied contexts.

2. **Add More Units in LSTM Layers:** Increasing the number of units in LSTM layers from 256 to 512 is intended to improve the model's ability to learn more complex representations. This change can be beneficial when dealing with tasks that require a high capacity for memorizing and processing information.

3. **LSTM with More Units and Tokens:** In addition to increasing the number of LSTM units, we can consider a configuration that also raises the maximum number of tokens from 25 to 30. This allows the model to process more sequences long, potentially offering a better understanding of extended contexts.

4. **LSTM lr=0.01:** Increasing the learning rate to 0.01 is an attempt to speed up model convergence during training. A higher learning rate may allow the model to fit the data more quickly, but it could also increase the risk of overfitting or instability.

5. **Other Possible Strategies:**

   (a) Test different batch sizes: Changing the batch size can influence the efficiency of learning and the speed of convergence of the model. A larger batch size can improve training stability, but may also require more memory.

   (b) Thoughtful selection of vocabulary used: Simplifying vocabulary can help focus learning on the most relevant words, potentially reducing noise and improving model performance on specific tasks.

Note on Training Neural Network Models: It is important to emphasize that training neural network models is a time- and resource-intensive process. For example, training on a system like SLURM can take around 16 hours, which forces us to be limited on our quantity of testing, especially since the latter has on many occasions found itself saturated with other projects.

7.1.2 Experimental seting

In Table 1 we summarize the experimental setting of each LSTM and GRU experiment, every experiment is trained for 20 epochs.

	GRU	GRU data+	LSTM	LSTM data+	LSTM units+	LSTM units+ tokens+	LSTM lr=0.01
Vocab Size	all	all	all	all	40 000	all	all
Max tokens	25	25	25	25	30	25	25
Embedding Size	200	200	200	200	300	300	200
Number of Units in Layers	256	256	256	256	512	512	256
Dropout in Encoder Layer	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Optimzer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001	0.001	0.001	0.001	0.01
Batch Size	128	128	128	128	128	128	128
Train Data	Global Voice	Global Voice + Wikipedia	Global Voice	Global Voice + Wikipedia	Global Voice	Global Voice + Wikipedia	Global Voice

Table 1: Experimental setup for GRU and LSTM models.

7.2 Transformers

7.2.1 Strategy

In Section 5 we present the details of the architecture. This model is the basis for the rest of the experiments. Each of the experiments change one parameter of the basic model. The goal is to give a baseline for the different experiments in terms of performance and to compare the impact of a less expressive model. In this section we have two strategy axis: 1) the performance of models with reduced expressive capacity in comparison with the base model, and 2) the impact of the inclusion of a more diverse train data in a model with reduced expressive capacity. Below we present the strategy for each transformer experiment.

- **Base Transformer.** This basic architecture is a 4 layer version of the model proposed by Vaswani et al.. We shorten the number of layers to 4 due to memory constraints.

- **3 layers** Using the same architecture and parameters as the base model, we decrease the number of layers to 3. The goal is to capture the decrease of performance, while saving computing resources.

- **Embedding 256.** Another way to decrease the number of parameters, and therefore the computing time, is to reduce the dimension of the embedding. A shorter embedding is less expressive and it also changes the schedule for the learning rate (Fig. 12)

- **Feedforward 256.** The feedforward network in each layer gives a non-linear representation of the output from the attention mechanism. The main goal is to extract features. A reduced feedforward dimensionality decreases the capacity of the model to extract features, but has as counterpart a reduction in the number of parameters.

- **Multiheads 4.** The base model includes 8 heads in the attention mechanism. Reducing the number of heads to 4 implies a trade-off. The trade-off here is that the model might be more computationally efficient with fewer heads, but it may sacrifice part of the parallel processing benefits that come with a higher number of heads.

- **Step Learn Rate Schedule.** The base model follows the warm up and cool down" schedule. The goal is to avoid the initial high sensitive to the initial learning rate by increasing it slowly and allow for exploration of the loss space. After, the

learning rate decreases to converge to a solution. In short, the warm up and cool down learning rate schedule helps balance the exploration and exploitation phases of training. In this experiment we propose a different schedule to contrast the warm up and cool down. Here we explore the sensitivity of the transformer to the learning rate. In Fig. 12, the step line shows the opposite of the warm up and cool down schedule. A high learning at the beginning rate cuts down the exploration phase, and therefore reduces the model expressiveness.

- **3 Layers and more data.** We increase the train data using *Wikimedia* text sequences. We ask ourselves if a less complex model can increase generalization if we add variability on the train set.
- **Multihead 4 and more data.** In tandem with the previous experiment, we inquire about the impact of more data on a less expressive model.

7.2.2 Experimental setting

In Table 2 we summarize the experimental setting for the transformer. Every experiment is trained for 10 epochs and none of them enter the overfitting stage where the validation loss starts to increase.

	Base Transformer	3 layers	Embedding 256	Feedforward 256	Multiheads 4	Step Learn Rate	3 layers and more data	Multiheads 4 and more data
English Tokenizer	BERT	BERT	BERT	BERT	BERT	BERT	BERT	BERT
French Tokenizer	BERT	BERT	BERT	BERT	BERT	BERT	BERT	BERT
Vocab Size	8000	8000	8000	8000	8000	8000	8000	8000
Max tokens	128	128	128	128	128	128	128	128
Buffer	20000	20000	20000	20000	20000	20000	20000	20000
Positional Enconding	Absolute	Absolute	Absolute	Absolute	Absolute	Absolute	Absolute	Absolute
Learn Rate Schedule	Warm up and smooth	Warm up and smooth	Warm up and smooth	Warm up and smooth	Warm up and smooth	**Warm up and step**	Warm up and smooth	Warm up and smooth
Optimizer Adam β_2	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
Optimizer Adam ϵ	1e-9	1e-9	1e-9	1e-9	1e-9	1e-9	1e-9	1e-9
Number of layers	4	**3**	4	4	4	4	**3**	4
Embedding dimension	512	512	**256**	512	512	512	512	512
Feedforward dimension	512	512	512	**256**	512	512	512	512
Multiattention heads	8	8	8	8	**4**	8	8	**4**
Dropout rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Batch Size	64	64	64	64	64	64	64	64
Train Data	Global Voices	Global Voices	Global Voices	Global Voices	Global Voices	Global Voices	Global Voices and Wikimedia	Global Voices and Wikimedia

Table 2: Experimental setup. We present the setup for 8 experiments. The Adam optimizer uses the default parameters, except for β_2 and ϵ.

The different experiments alter the number of weights in each model. In Fig 16 we observe that using only 4 heads reduces the number of parameters by a factor of 6 in comparison to the baseline transformer.

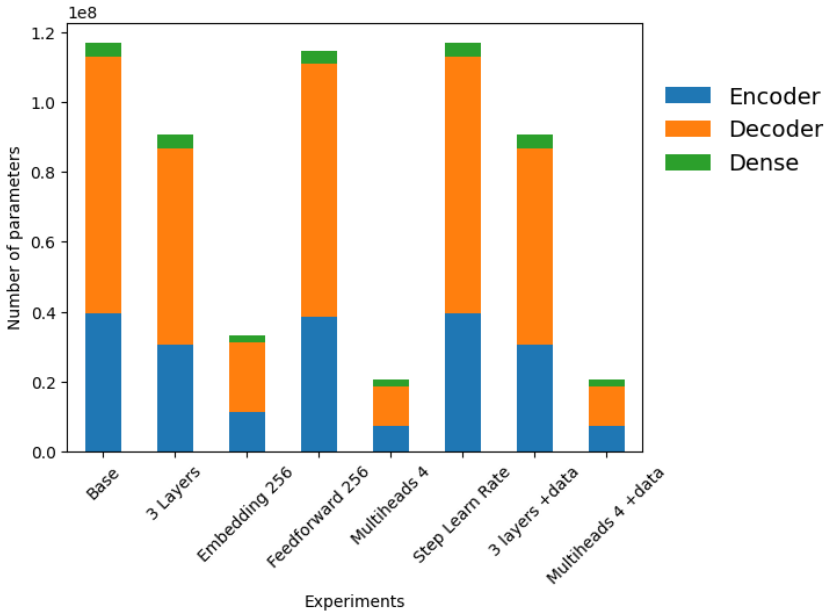


Figure 16: Number of parameters in each experiment

7.3 MARIANMT

In total, we did 6 fine-tuning experiments with the MarianMT model, all the them on 3 epochs with 3 three different settings on the two different dataset (base dataset and base dataset with new sample of *Wikimedia*). The goal is to evaluate the performance of the model when we increase the number of layers that can be fine tuned.

- **MARIANMT Baseline**: With the model pre-trained, we simply predict on the testing set without fine tuning
- **MARIANMT Finetuning 1 layer**: With this model pre-trained, we freeze all the layer except the last layer of the decoder and we train with a learning rate of 3e-5.
- **MARIANMT Finetuning 2 layers**: The same structure as before except we freeze all the layer except the last 2 layers of the decoder and we train with a learning rate of 3e-5.
- **MARIANMT Finetuning with a Learning rate schedule**:In this setting, we have a different learning rate for different parts of the model. For the encoder, we have a learning rate 3e-5 and for each layer of the decoder we have 9.375e-06, 1.875e-05, 3.75e-05, 7.5e-05, 0.00015, 0.0003, respectively (such as in the Fig.17).

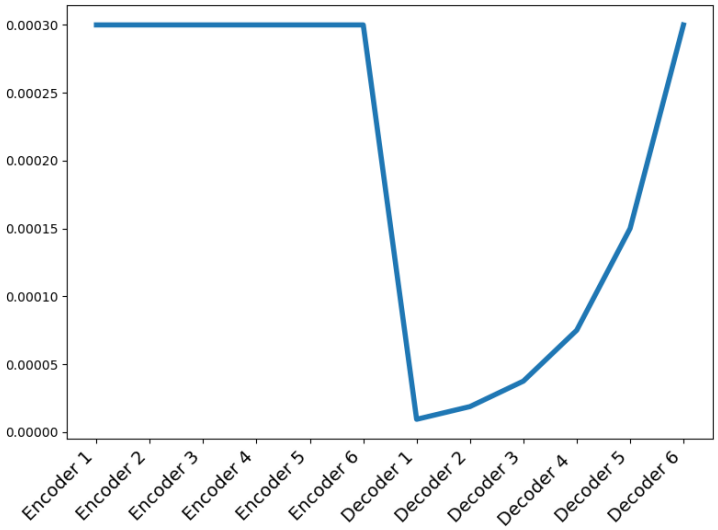


Figure 17: Learning rate schedule for each part of the encoder and decoder

7.4 Results

7.4.1 GRU and LSTM

The meteor score performance of the GRU model improves with the inclusion of additional train data from Wikimedia. The difference increases with the increment of lenght in the sequences (see Fig. 18. The LSTM model achieves a better meteor score than the GRU models. The addtion of trainig data does not improve the LSTM performance. Among all the LSTM and GRU models, the LSTM with learning rate of 0.01 (10 times more than the learning rate in the other models) is the worst performer.

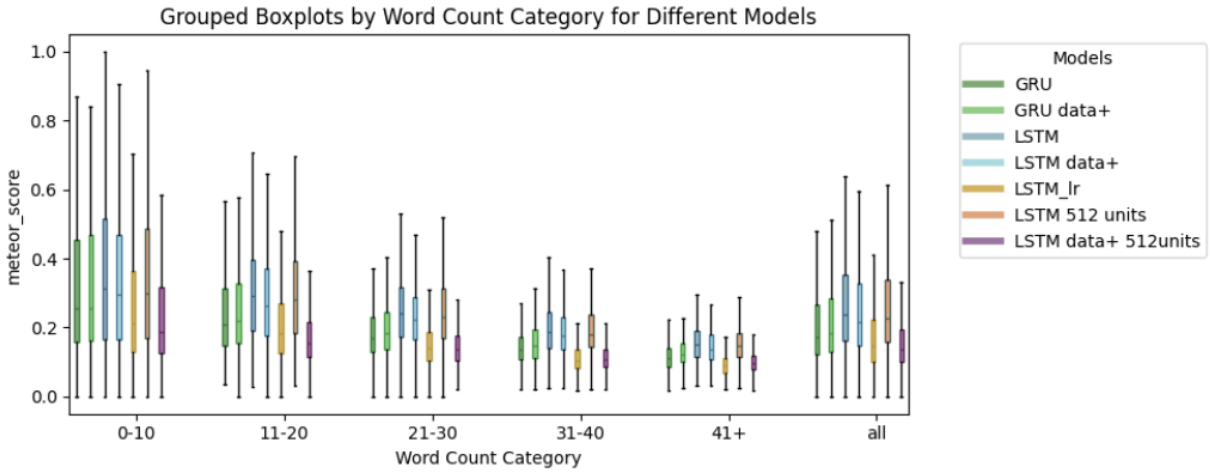


Figure 18: Grouped Boxplots by Word Count Category for Different RNN Models by using the meteor score

With respect to the Blue score in Fig. 19, longer sequences are more challenging for all of the GRU and LSTM models. As in the meteor score, the LSTM model is better than the GRU models and the model with higher learning rate is still the worst performer. Mirroring the meteor results, additional data does not help learning.

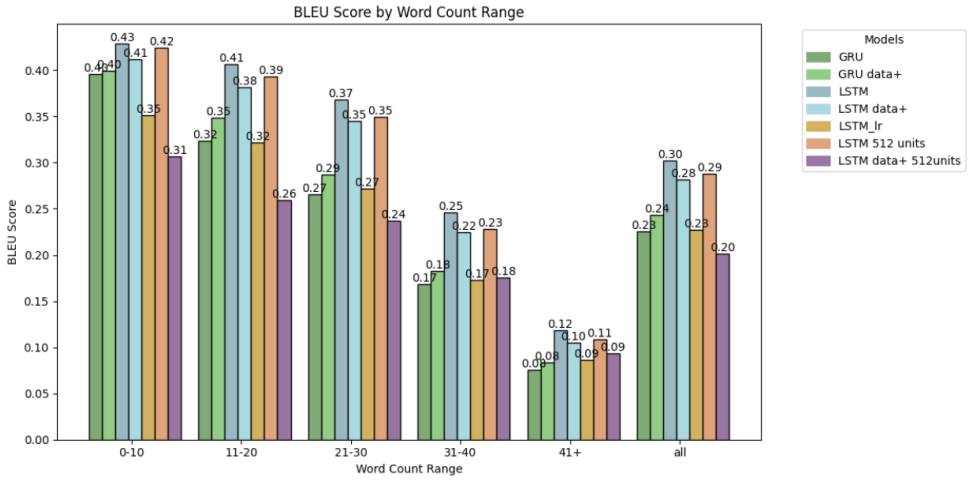


Figure 19: Blue score in GRU and LSTM models

	GRU	**GRU data+**	**LSTM**	**LSTM data+**	**LSTM units+**	**LSTM units+ tokens+**	**LSTM lr=0.01**
Epochs Achieved	20/20	19/20	18/20	15/20	16/20	16/20	49/50

Table 3: Saving each model with the lowest val_loss

7.4.2 Transformers

The results comprehend the meteor and the blue score. In Fig 20 we present the meteor score according to sequence length for the transformer experiments. In the short sequence category, i.e. phrases with 1 to 10 words, two of the models with less expressive capacity (3 layers and Feedforward 256) have a median meteor score higher than the base transformer. While, in the same category length, a reduction of the embedding dimension decreases the meteor score. After the addition of sequences from Wikimedia to the train set, the performance decreases. For the rest of the categories, and in general, the meteor score results are almost the same for all experiments model. The model with less parameters is the Multiheads 4, and it is as capable as the base model, no matter the sequence length category.

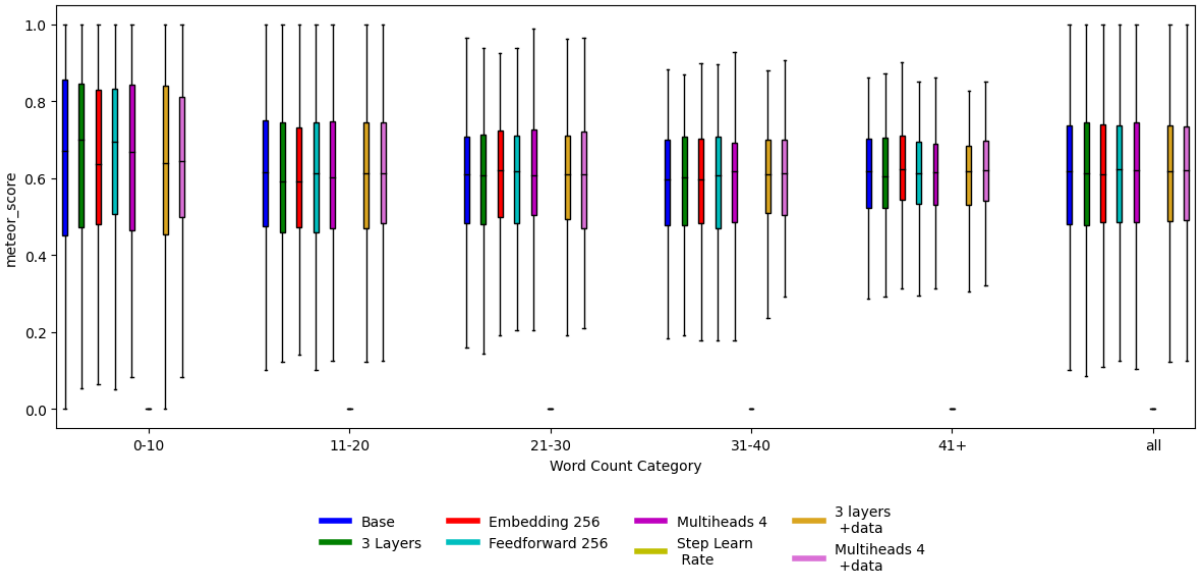


Figure 20: Meteor score in Transformer experiments

The blue score for the whole corpus in the transformer experiments is in Fig. 21. We cannot observe a trend on the values. Briefly, the transformer experiments in the category of short sequences behave differently in terms of semantics (meteor) but not in terms of alignment (blue) Both in Fig 20 and Fig 21, the experiment with a step decreasing learning rate schedule shows a 0 score. The warmup and cooldown schedule focuses on a slow exploration of the loss space. A model without this exploring phase cannot output translations. The substraction of the exploration phase is the aspect that reduces the expressive capacity of the model the most.

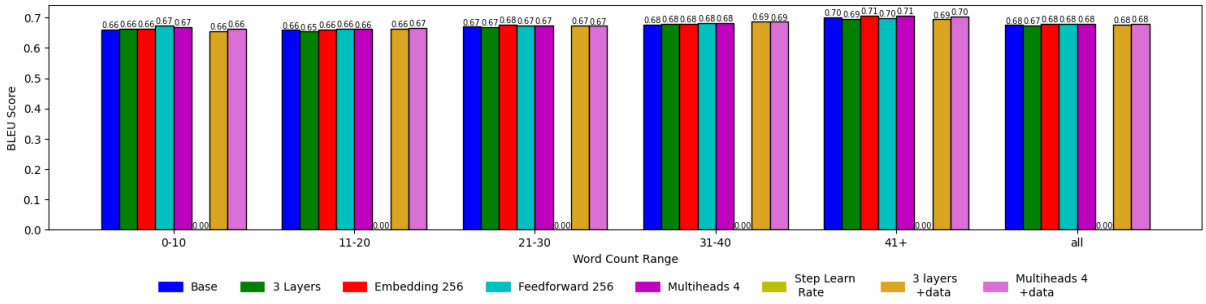


Figure 21: Blue score in Transformer experiments

7.4.3 MarianMT

In the Fig.22, we present the meteor score results of the different experiments using MarianMT tuning. Foremost, we have better results without fine tuning than with it, this behaviour is called negative transfer. In general, as we increase the number of trainable layers, the meteor score improves. We observe this trend in each word count category.

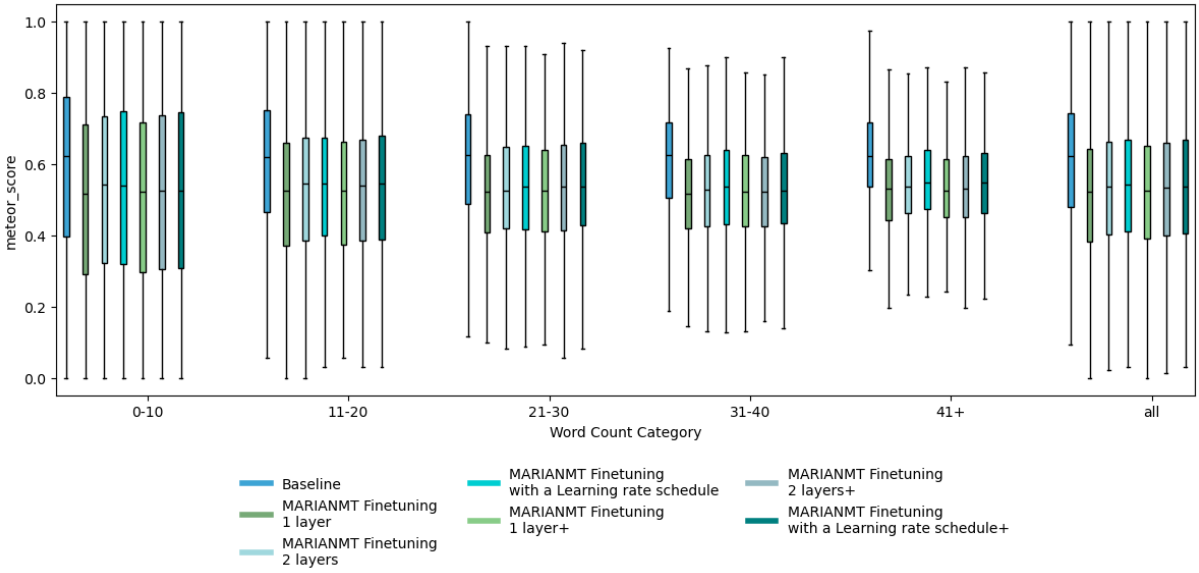


Figure 22: Meteor score of baseline and tuned MarianMT

in Fig 23, the blue score shows the same trend we observe in Fig. 22. Therefore, we arrive to the same conclusion: more trainable parameters increase the performance.

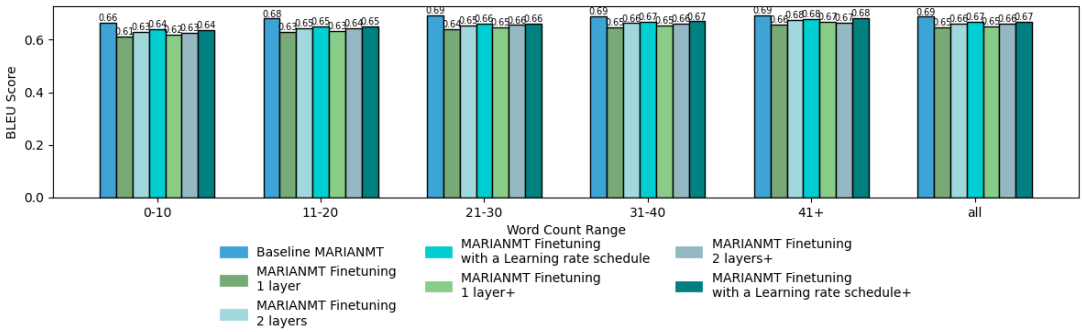


Figure 23: Bleu score of baseline and tuned MarianMT

8 Conclusion

From the results, we obtain the following conclusions:

- As expected, the LSTM model performs better than the GRU model and an higher learning rate hinders learning.
- A reduction in model expressiveness for the transformer cannot be overcome by increasing the size of the train set.
- The difficulty of the translation task is determined by the sequences we find in Global Voices. Using 4 attention heads instead of 8 means a reduction of 80% in the number of parameters, but the performance is not compromised.
- Tuning the MarianMT model shows worse performance in comparison to the pre-trained model. One possible reason is the quality of the Global Voices data.

- Among the different tuned MarionMT models, the fewer frozen parameters, the better results we get.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

et al. Bahdanau. Neural machine translation by jointly learning to align and translate. 30, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Yushuo (Shawn) Han. English to italian machine translation with bilstm and attention. 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210*, 2023.

Prokopis Prokopidis, Vassilis Papavassiliou, and Stelios Piperidis. Parallel global voices: a collection of multilingual corpora with citizen media stories. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 900–905, 2016.

Ahmed S Soliman, Mayada M Hadhoud, and Samir I Shaheen. Mariancg: A code generation transformer model inspired by machine translation. *Journal of Engineering and Applied Science*, 69(1):1–23, 2022.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Jörg Tiedemann and Lars Nygaard. The opus corpus-parallel and free: http://logos. uio. no/opus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, 2004.

Jörg Tiedemann and Santhosh Thottingal. Opus-mt–building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*. European Association for Machine Translation, 2020.

Jörg Tiedemann, Mikko Aulamo, Daria Bakshandaeva, Michele Boggia, Stig-Arne Grönroos, Tommi Nieminen, Alessandro Raganato, Yves Scherrer, Raul Vazquez, and Sami Virpioja. Democratizing neural machine translation with opus-mt. *Language Resources and Evaluation*, pages 1–43, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.