Low-Resource Machine Translation

Alexander Peplowski
Université de Montréal
alexander.peplowski@gmail.com

Harmanpreet Singh
Université de Montréal
harmanpreet.singh@umontreal.ca

Marc-Antoine Provost
Université de Montréal
provoma1@hotmail.com

Mohammed Loukili
Université de Montréal
mohammed.loukili.1@umontreal.com

ABSTRACT

In this paper, we improve Neural Machine Translation (NMT) by making use of monolingual data to augment the parallel training corpus with back-translations of target language sentences. We utilize an iterative back-translation method for generating increasingly better synthetic parallel data from monolingual data to train our low-resource neural machine translation scenario of English to French dataset. We perform multi-step back-translation experiments on Transformer architecture. The experimental results show that the proposed method achieves a BLEU score of around 17.64 on the English to French test set. All the code and notebooks supporting this work are published in the Team 08 github repository [1].

1 INTRODUCTION

Machine Translation is the task of automatically translating a source language to a target language. The present paper describes a study that was made about machine translation in a low resource setting. Recent systems have shown to often surpass human-level performance in some language translation task [15]. Unfortunately, these systems are known to require a large volume of translated parallel text (pair of source and target text) and to be computationally expensive at training and inference time. With so many languages present today, there are tons of languages and dialects that don't have access to massive amount of parallel data. Moreover, many countries where those languages are spoken may not have access to the computational power needed to harness such data. These issues have hindered Neural Machine Translation's (NMT) practical use cases in low-resource settings. Accurately performing low-resource NMT systems have the potential to translate forgotten dialects and languages which are the core of many cultures and overcome many of the weaknesses of conventional phrase-based systems. This highlights the needs to build systems that can work for everyone, even when parallel text is scarce. As such, the goal of the study was to implement a Machine Translation system from English to French with limited parallel examples. Moreover, our algorithm needed to generate French sentence with proper capitalization and punctuation even when missing from the source corpus.

To evaluate the performance of our model, the BiLingual Evaluation Understudy (BLEU) metric was used [27]. BLEU is a quality metric score for machine translation systems that attempt to measure the correspondence between a machine translated output and a human translation. See section 4.8 for a more detailed explanation of the evaluation metric.

2 LITERATURE REVIEW

Machine Translation is a relatively old topic that dates back from the 1950s with rule-based Machine Translation [28]. In this section however, we will focus on the Neural Machine Translation (NMT) era that has achieved tremendous breakthrough, thus gaining rapid growth and popularity from the academic community as well as the corporate community.

One of the first key idea to NMT comes from sequence-to-sequence models introduced by Cho et al. [5] and Sutskever et al. [24] The former used an encoder-decoder architecture using Recurrent Neural Networks (RNNs) where the encoder, encodes the input sequence to a fixed-length vector representation, and the decoder, decodes the representation into another sequence of symbols[5]. They also introduced the GRU (Gated Recurrent Unit) units which are a variant of the Long-Short-Term Memory (LSTM) model[10], responsible for memorizing and forgetting information from the past sequences. The latter [24], used similar architecture as the former, but the main difference was the inversion of the input sequence, and the use of LSTM as hidden units, as they proved their capability for explicit memory deletes and updates. This alleviates the exploding or vanishing gradient problem that hindered learning long-term dependencies with standard RNNs. However, while alleviating one problem, this newly proposed architecture created a bottleneck issue at the encoder.

In order to address this issue, Bahdanau et al. [3] proposed an extension to the encoder-decoder model which learns to align and translate jointly. Their key idea is that each time the proposed model generates a word in a translation, it soft-searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words. This distinguishes from the original encoder-decoder approach from the fact that it does not attempt to encode the whole input sentence into a single fixed-length vector. Instead, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors dynamically while decoding the translation.

In addition to the capacity of an NMT model to represent sequence of symbols, word representation can also help learning algorithms to achieve better performance in natural language processing tasks by grouping similar words [14] with the same meanings. In (Mikolov et al.) [14], they introduced Skip-gram model (word2vec) which shows interesting representation for words that share same characteristics, as shown in this example: "Berlin" - "Germany" + "France" = "Paris". As an extension to Word2Vec, FastText [4] is an embedding technique proposed by Facebook that treats each word

as a sum of character n-grams. This enables the model to generate better embeddings for rare and out-of-vocabulary words. Other models have emerged following this success, like GloVe (Global Vectors for Word Representation) [17] which try to produce a model that benefits from statistical characteristics from the corpus like the occurrence of the words. Even if these models showed great improvement on the RNN-based NMT model's performance, the main drawback is that the word representation will be the same regardless of context. To tackle this problem, Allen Institute came with a new model called ELMo (Embeddings from Language Models) [18] which claims to have a model that capture word's syntax and semantics as well as the change of these characteristics across linguistic contexts. All these methods gave a real improvement on RNN-based encoder-decoder models, until the arrival of the Transformer model[26], who revolutionized the traditional approach to NMT models.

Eschewing previous ideas of recurrence and instead relying entirely on attention mechanism to draw global dependencies between input and output, Google proposed the Transformer model [26]. Transformer is a self-attention based architecture with the ability to attend to different positions of the input sequence to compute a representation of that sequence. It relies entirely on an attention mechanism to draw global dependencies between input and output. Moreover, it is more parallelizable and require significantly less time to train than RNNs and GRUs. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. The encoder in transformer maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_n)$ of symbols one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. It efficiently models variable-sized long-term dependencies in parallel using stacks of self-attention layers. Since the input have a temporal/spatial relationship, positional encodings is added to the inputs to give the model some information about the relative position of the words in the sentence. As stated in the introduction, we were also constrained since we were in a low resource scenario. Thus, it is relevant to summarize prior work relevant to low resource NMT. Proposed by Sennrich et al. [21], an approach used to address this challenge is by using back-translation to do data augmentation. Back-translation operates in a semi-supervised setup where both bilingual and monolingual data in the target language are available. It first trains an intermediate system on the parallel data which is used to translate the target monolingual data into the source language. The result is a parallel corpus where the source side is synthetic machine translation output while the target is genuine text written by humans. The synthetic parallel corpus is then added to the real bitext in order to train a final system that will translate from the source to target language [7].

Other researchers have tried a different approach to tackle the problem of low resource NMT. One off these approach by Sanh et al. is Hierarchical Multi-Task Learning (HMTL) [20]. Multi-task learning is a general method in which a single architecture is trained towards learning several different tasks at the same time. One fundamental motivation between multi-task learning is that loosely

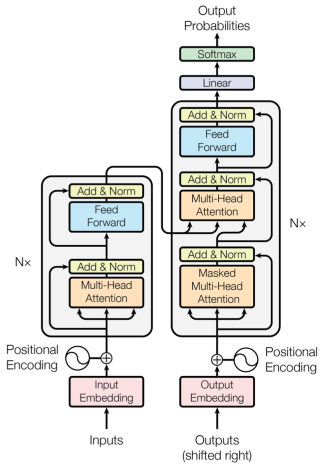

Figure 1: Transformer architecture by Vaswani et al. [26]

related tasks can benefit from each other by inducing richer representations. As such, the authors of this paper propose a unified model that trains and combines four semantic NLP tasks in a hierarchical way, i.e., training on simpler tasks first and using the knowledge to train on more complicated tasks. The authors observed that the combination of those four tasks led to state-of-the-art performance on three of those four tasks. They also stated that lower level shared embeddings already encoded a rich representation and that as they moved from the bottom to the top layers of the model, the hidden states of the layers tended to represent more complex semantic information [19].

Another approach that has gained popularity in the NLP community is model pre-training. Although pre-training for embedding has already been used for several years as discussed in section 4.4 and section 4.5, model pre-training has been increasingly popular for Natural Language Processing tasks. With their paper Multilingual Denoising Pre-training for Neural Machine Translation [12], Facebook Researchers propose mBART - a sequence-to-sequence denoising auto-encoder pre-trained on large-scale monolingual corpora in many languages. This method enables pre-training a complete sequence-to-sequence model by denoising full texts in multiple languages simultaneously. Pre-training a complete model allows it to be directly fine tuned for supervised and unsupervised machine translation, with no task-specific modifications. The experiments show that using the pre-trained mBART model and continuing training on parallel data helped for language pairs for which there was only a limited amount of parallel data.

3 DATA ANALYSIS

3.1 Data sources

We were provided with two main data sources. The first one being 11 000 parallel examples (from English to French). Each example consists of one or many sentences and while the English source text lacks formatting, the French text contains all the proper formatting such as capitalization and punctuation. Note that the aligned data is also already tokenized, that is, each token has been separated

by a space. The second data source was two unaligned monolingual corpora of 474 000 examples : an English corpus of 474 000 examples and a French corpus of 474 000 examples. Both are properly formatted with full capitalization and punctuation and are untokenized.

3.2 Detailed Data analysis

We begin by analyzing the structure of the data that we were provided. Since the unaligned data is not formatted in the same way as the aligned text, we format it such that the same formatting is shared across the whole corpus of a single language. In the case on the English data, the unaligned data is tokenized, lowercased, and stripped of punctuation. For French data, the unaligned data is tokenized only.

3.2.1 Analysis of Tokens in Corpora. In this section we justify the design decisions we made based on the distribution of tokens in the corpora.

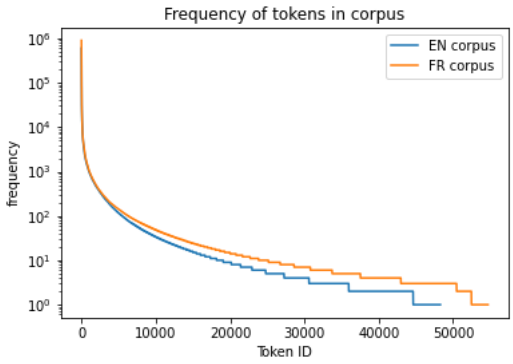

Figure 2: Token Frequency

3.2.2 Number of Unique Tokens. When creating a tokenizer, it is necessary to limit its size. The tokenizer will need to retain in its memory a list of all tokens that exist in the corpus in which it will operate. It might be impractical to have a large set of tokens since language modelling models are habitually required to represent each token in memory. Therefore, we aim to limit the size of the representable tokens for training performance reasons while hopefully still being able to represent the majority of tokens available in the corpus.

Refer to figure 2, where we list the frequency of occurrence of all token ids. We observe that, at the high end of the spectrum, some tokens occur thousands of times in the corpus. It would be important to ensure that these tokens are retained by our tokenizer. Also, we observe that both the English and French corpora have similar token distributions, with the French corpus having slightly more tokens defined.

Refer to figure 3, where we analyze the impact of different cutoff choices for the maximum number of tokens. We decide that it is acceptable to limit our token definition to the most frequent 30 000 tokens. With this choice, we obtain 99.6% and 99.3% coverage for the English and French corpora respectively. In this case, when we

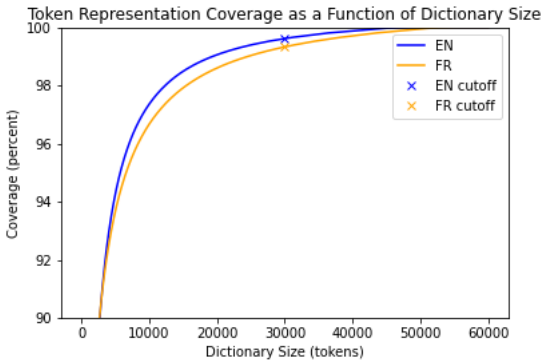

Figure 3: Token Coverage

talk about coverage, we refer to the percentage of tokens from our corpora that are representable by our tokenizer with a specified cutoff token frequency.

3.2.3 Maximum Tokens per Sample. When using models that have a finite input size, it is important to define a limit on the length of sequence that can be passed. In a ideal case, we would like this maximum length to be at least as large as the longest sample in our corpus. However, for training performance, it is often impractical to set a high limit on the sequence length. Therefore, in this section we try to determine at which point should we set a limit on the input token sequence.

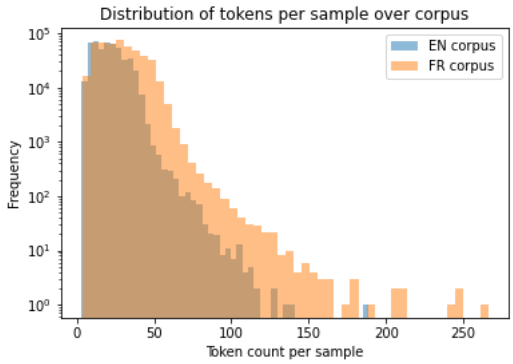

Figure 4: Sample Length

Refer to figure 4, where we show the number of tokens required to represent a single sample from the corpora. We can easily observe that the majority of samples can be represented by fewer than 50 tokens. We also observe that samples from the French corpus require more tokens per sample in general.

Refer to figure 5. In this figure, we show the impact of choice of different maximum tokens per sequence choices. We show the percentage of tokens that would be retained by the tokenizer as a function of cutoff choice. It is clear that we can retain most of the information in the corpus when setting a threshold over 60. In our case, we decide to use a threshold of 80 for the English corpus and

120 for the French corpus. These choices yield a coverage of 99.97% and 99.98% respectively.

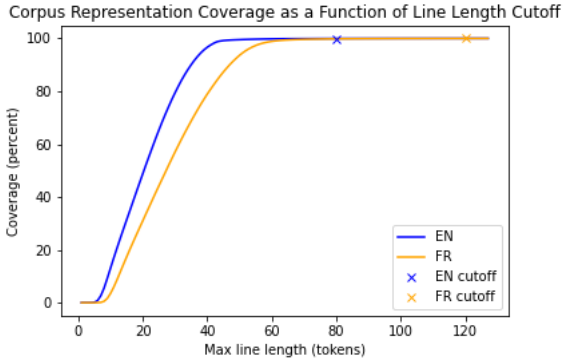

Figure 5: Sample Length Coverage

4 METHODOLOGY

4.1 Data processing

We are given two types of input examples, aligned English-French sample, which are already been tokenized. The preprocessing steps are specific to the model that we are using, but generally some steps are necessary in all cases, which are:

- Clean the data from irrelevant characters
- Add special tokens (start, end) to source and/or target sequences
- Encode the tokens using the vocabulary index
- Pad the sequence to the maximum length of the original language

Some models as described in [24] inverse the input sequence and do a padding on the left side, to reduce the distance between the input sequences on both the encoder and the decoder sides. We have tried this approach for our experiments on sequence to sequence models.

4.2 Handling Capitalization

As part of the problem statement, we are required to correctly punctuate the French translations as well as handle capitalization correctly. We note that tokenizing directly without handling capitalization will generate a couple of problems: A large dictionary of tokens will be generated since words with and without capitalization will be detected separately. Secondly, some tokens will encode identical words or sub-words but will be different due only to capitalization. Words with the same meaning will have different encoding, which is clearly problematic.

To get around this issue, we introduce a capitalization token. The capitalization token is produced when the leading letter of a word is a capital. The token is produced and the leading letter of the word is lower-cased instead. Essentially, capitalized words are split into at least two tokens, the capitalization token, and the word with leading lowercase letter.

The capitalization scheme will be applied as a first step before applying a tokenization scheme. That is, the capitalization token is added first, and then some other tokenization scheme can be applied. This process is done in reverse when decoding from tokens to raw text.

4.3 Tokenization Strategy

The actual tokenization scheme used by our project imports the Huggingface [29] implementation of the byte pair encoding scheme [23] as used by the RoBERTa [13] implementation. Byte pair encoding is useful for encoding sub-words and for reducing the maximum number of tokens required to represent a corpus. Both of these properties allow for easier representation of rare words in tokens and for reduced training time.

A separate tokenizer is created for the English corpus and the French corpus. Start and end tokens are added when encoding a string. The maximum vocabulary size for the tokenizer is 30k tokens for both English and French. The maximum sequence length was chosen to be 80 for the English corpus and 120 for the French corpus as to have almost 100% coverage from both languages.

4.4 Word Embeddings

Many Word Embeddings techniques with different architectures were tried during the project. Some of the architectures that we experienced with were Word2Vec, FasText, ELMo, and BERT embeddings. Word2Vec, FastText and ELMo embeddings were trained on the unaligned corpora. We will see in the following paragraphs how these embeddings improved the models' performance.

4.5 BERT Language Modeling

One effective approach for generating word embeddings is by training a BERT-style [6] masked language model. The original approach consists of first pre-training the model using a combination of Masked Language Modelling (MLM) task (refer to figure 6) and a Next Sentence Prediction (NSP) task. Following the pre-training, the model can be fine-tuned on the task of interest.

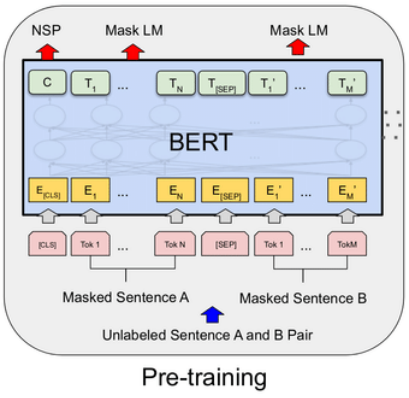

Figure 6: BERT Pre-training

To leverage the BERT model, our approach uses the BERT implementation from the Huggingface library [29]. The hyper-parameters of the BERT model are selected as to minimize model capacity, since the original BERT implementation is a very high-capacity model

and we have a very small dataset. Therefore, our model was trained using a hidden size of 128 and a depth of 2 layers.

We setup our training environment by defining a monolingual MLM task: Given an unaligned input sentence, an in the original BERT paper, we will randomly select 15% of the input tokens for training, that is, the model will need to predict the correct token for only these selected token (gradients are ignored for the unaffected tokens). 80% of the selected tokens will be replaced with a masking token, 10% will be replaced with a random token, and the remaining 10% will be left as-is. The model's task is to predict the correct token. Training the BERT model generates word (token) embeddings, which will be used downstream.

4.6 Model description

4.6.1 Sequence to Sequence Models. We have tried different sequence to sequence models in our experiments. We have started with simple architectures as described in Cho et al. [5] and Sutskever et al. [24], that is encoder-decoder model where the encoder learns a fixed-length vector that next be fed to the decoder. As this kind of model performs poorly on the given data-set, we have tried more sophisticated models by using bidirectional LSTM encoder. This model does not show any better performance in comparison with the former, so in order to get better performance, we have tried to add pre-trained word embeddings technique to help the model learn better. We have tried to use ELMo [18] model to get pre-trained embeddings from unaligned corpus. To train ELMo we have followed the instructions given by Allen institute [18] using their Tensorflow implementation https://github.com/allenai/bilm-tf. We have not used their pre-trained embeddings, but instead we have pre-trained the model from scratch on the provided unaligned English and French corpus separately. Please refer to our repository for full steps [1].

4.6.2 GRU with Attention. Another architecture tried during the span of the project was a sequence-to-sequence model with Attention. We took inspiration from the sequence-to-sequence model with Attention proposed by Bahdanau et al. [3] described in section 2. This model architecture acted as a baseline before implementing more refined models. We used Gated Recurrent Units for the encoder and decoder architecture and Bahdanau Attention's [3] implementation in Keras. The encoder and decoder contained a single layer with 512 cells and 100-dimensional embeddings when using pre-trained embeddings such as Word2Vec and FastText and 256-dimension embeddings otherwise. Data processing steps described in section 4.1 were used, such as <start> and <end> token and sentence padding. In fact, the model reads through all the source word until the end-of-sentence token is reached. Then, it starts predicting one word at a time, which is then used to calculate the loss using the Cross entropy loss objective (see section 4.7.1). Target words were also passed as the next input to the decoder (teacher forcing).

4.6.3 Transformer Model. We performed extensive number of experiments on transformer model based on encoder-decoder architecture. We follow the architecture stated in the original transformer paper [26] with few variations.

The encoder and decoder as demonstrated in figure 1 are composed of a stack of N = 4 identical layers. Each layer in encoder has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. The residual connection [8] is employed around each of the two sub-layers, followed by layer normalization [2]. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{model} = 128$. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, residual connections are employed around each of the sub-layers, followed by layer normalization. The self-attention sub-layer in the decoder stack is modified to prevent positions from attending the subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i. We use the $h = 8$ parallel attention heads. In addition to attention sub-layers, each of the layers in encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between. The dimensionality of input and output of the position-wise feed-forward networks is $d_{model} = 128$, and the inner-layer has dimensionality of $d_{ff} = 512$. The learned embeddings are used to convert the input tokens and output tokens to vectors of dimension d_{model}. Moreover, the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. Since transformer model doesn't contain recurrence and convolution, positional encodings of size d_{model} are added to the input embeddings at the bottoms of the encoder and decoder stacks for the model to make use of the order of the sequence. The *sine* and *cosine* encodings was generated following the approach described in [26].

4.7 Transformer Training Details

Both English and French language sentences are first converted into tokens using the byte-pair [22] representation used by many state-of-the-art models in machine translations. We build separate sub-word tokenizers for each language based on the vocabulary of aligned and unaligned datasets. This helps us against the out-of-vocabulary words problem that arises when new words occur during inference. English sentences were encoded to the length of 80, while French sentences were encoded till 120 words based on the coverage as described in figure 5. Moreover, French corpus also consisted of punctuation leading it to have longer encoded token lengths. The transformer is an auto-regressive model: it makes predictions one part at a time, and uses its output so far to decide what to do next. During training we use teacher-forcing, i.e. passing the true output to the next time step regardless of what the model predicts at the current time step. As the transformer predicts each word, self-attention allows it to look at the previous words in the input sequence to better predict the next word. To prevent the model from peaking at the expected output, the model uses a look-ahead mask. Target sequences are padded so we apply a padding mask when calculating the cross-entropy loss described in section 4.7.1. Adam optimizer was used with a custom learning rate scheduler according to the formula in the paper [26]. The learning

rate was increased linearly for the first warm-up steps of 4000 training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. Residual dropout rate of 0.1 and layer normalization regularization techniques were used. The BERT language masking model generated embeddings were utilized. We save checkpoints every 2 epochs and and continuously monitor the loss, accuracy and BLEU score during evaluation. During evaluation and translation setting, we pass the start token as an input to the decoder, calculate the padding masks and the look ahead masks, and the decoder outputs the predictions by looking at the encoder output and its own output (self-attention). We select the last word and calculate the argmax of that. Later the predicted word is concatenated to the decoder input, and passed to the decoder. In this approach, the decoder predicts the next word based on the previous words it predicted.

4.7.1 Loss Function. For neural sequence model training, maximum likelihood (ML) has been commonly adopted to optimize model parameters with respect to the corresponding objective. In practice, ML-based loss is often represented with a word-level cross entropy form, which has proven to be effective for NMT modeling. The training objective of our machine translation models is to minimize the categorical cross entropy loss function.

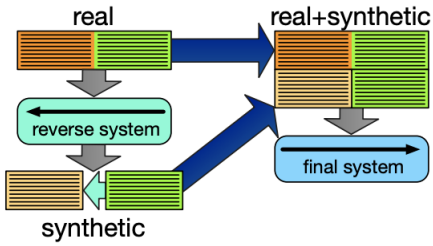

Figure 7: Iterative Back-Translation for NMT by [9]

4.7.2 Iterative Back Translation. Back Translation [7, 21] is a simple and effective method to leverage the monolingual data as it does not require modification to the machine translation training algorithms. We train our transformer based NMT system from source-to-target (i.e. EN to FR) using the aligned training corpus. Later, this model is used to synthesize 30k FR sentences from unaligned corpus. This synthetic parallel corpus along with existing aligned data is now used to train a reverse translation model, FR to EN in our case. The BLEU score on this model during first iteration was 8.3. This reverse translation model is further used to synthesize 100k EN monolingual sentences. The BLEU score was improved with every iteration. We perform this back-translation method thrice, and generate the synthetically translated corpus repeatedly, adding more unaligned data in each iteration. The final FR to EN model used to generate synthetic translation had a final BLEU score of 18 on our held out test set. This data was used to train our original task of EN to FR translation, yielding 17.6 BLEU score on our test set. Figure 7 illustrates the idea of back translation. We maintain a certain ratio of aligned and unaligned corpus when training these models by duplicating the aligned corpus when required. Unaligned-vs-aligned ratio of around 6 gave us the best results in our experiments. At the end, we also fine-tune the final EN to FR model by training the model on aligned dataset for 10 epochs.

Setting	EN to FR BLEU Score
transformer NMT baseline	10.42
back-translation	12.64
back-translation iterative+1	14.30
back-translation iterative+2	17.64

Table 1: Low Resource setting: Impact of the quality of the back-translation systems on the benefit of the synthetic parallel for the final system in a low-resource setting.

4.8 Model evaluation

In order to evaluate our models, we divided the aligned corpora into a training and validation set using Scikit learn's [16] function *train_test_split* with shuffling and a random seed for reproducibility purposes. We made sure to keep the indices of the shuffled tensors in order to translate them back to their original sentence representation and construct our text representation of the training and validation set. These files were then available in our project directory on the cluster for testing our models. Due to the limited amount of parallel examples, we used a 90-10 training and validation split. We evaluated our models with the BLEU metric. It is a quality metric score for machine translation systems that attempts to measure the correspondence between a machine translated output and a human translation. As such, the higher the score is, the closer the machine translation is to a professional human translator. This metric scores a translation on a scale of 0 to 1, but is often stated on a scale of 1 to 100 (like in this paper), although it does not refer to a percentage of accuracy. BLEU uses a modified form a precision to compare a candidate translation against multiple reference translations. First, let's define m being the number of words from the candidate translation that are found in the reference translation and w_t the total number of words in the candidate translation. The modified precision used in BLEU metric for uni-grams is the following; for each word in the candidate translation, the algorithm takes its maximum total count, $m_{\max}$, in any of the reference translations. For the candidate translation, the count m_w of each word is clipped to a maximum of $m_{\max}$ for that specific word. These clipped counts m_w are then summed over all distinct words in the candidate. Then, the sum is divided by the total number of uni-grams in the candidate translation. Usually, the number of n-grams used is 4. [11] We can illustrate this by the following example from Papineni et al.[11]

Candidate	the the the the the the the
Reference 1	the cat is on the mat
Reference 2	there is a cat on the mat

Table 2:
Example of modified precision used in the BLEU metric for Machine Translation from the original paper [11].

In the example above, the word "the" appears twice in reference 1 and once in reference 2, thus $m_{\max=2}$. Also, $m_w = 7$, but since

$m_{\max=2}$, m_w is clipped to 2. The modified unigram precision would thus equal to $P = \frac{2}{7}$. To produce a score for a whole corpus, the geometric mean of the test corpus' modified precision scores is taken and then multiplied by an exponential brevity penalty factor to prevent very short sentences from receiving a too high score. The geometric average of the modified n-grams precision, p_n is computed using n-grams up to length N and positive weights w_n summing to 1. Next, let c be the length of the candidate translation and r be the effective reference corpus length. To compute the brevity penalty, BP;

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{1}$$

Then,

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2}$$

The ranking behavior is more evident if seen in the log domain,

$$\log \text{BLEU} = \min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} w_n \log p_n \tag{3}$$

Model	**Pretrained Embeddings**	**SacreBLEU Score**
Baseline RNN	—	0.26
	W2V	—
Bidirectional LSTM	—	1.36
	W2V	—
GRU with Attention	—	2.65
	W2V	5.15
	FastText	5.05
Transformer Model	—	8.18
	BERT	10.42
Transformer Model + Back-Translated Data	—	14.86
	BERT	17.64

Table 3: BLEU Score summary of various models on held-out test set

5 RESULTS AND DISCUSSION

5.1 Bidirectional LSTM

In general, sequence to sequence models showed poor performance according to BLEU metric [27]. The first model, that is an LSTM Encoder-Decoder got a BLEU score of 0.26. The second model, which has a bidirectional LSTM in the encoder part, got a better result of 1.36. Both results are far behind the result that we have got with our best model. Another model that we have experienced with, bidirectional LSTM with ELMo embeddings, was very heavy to run and very slow to train. In fact, the ELMo training could not be done under 12 hours, which is the maximum time allowed on Helios cluster. We have used similar hyperparameters as the original algorithm, except for the output dimension which we chose 256. With the weights given by ELMo, we generated beforehand the embeddings of the input sequences for the encoder (English sentences) and the decoder (French sentences), as it would take too much time if we generated embeddings on the fly. We trained the model for 20 epochs, which took about 7 hours. The inference phase took more time than any other model used. This is because of the ELMo generation for every predicted token. The final result of this model was a BLEU score of 0 which is surprisingly worse than other results. The model has more capacity and more information to theoretically perform better than the others. We suspect two main reasons to this results. 1) The ELMo algorithm was performed in Tensorflow 1.15 which is not compatible with Tensorflow 2.x this incompatibility doesn't help us to save best parameters of this model for the inference phase. 2) The ELMo training was not optimal and hence did not give us optimal embeddings and did not help the model perform better.

5.2 GRU with Attention

As reported in the results, see table 3, the pre-trained embeddings greatly helped the performance of the sequence-to-sequence models with attention. Reported in section A, is a translated sentence of the best performing model in this family, with its attention map 10. We can see that the model was able to learn punctuation even when missing from the source language. We can also observe that the model was able to learn negation thanks to its attention mechanism. However, while able to learn those, the model struggled to correctly translate source sentences and often made mistakes. We can infer that it is due to the low-resource setting we are in and the occurrence of rare words that the model couldn't recognize, especially without the pre-trained embeddings on the unaligned corpora. Moreover, the architecture tried had a low capacity, with a single uni-directional layer composed of GRU gates and 100-dimensional embeddings, which may have impacted the learning capability of the tried architecture.

5.3 BERT Language Modelling

The training progress for the BERT MLM model for the English corpus can be observed in figure 8. We observe that the progress is slow, which is expected, since this is a difficult task; The MLM task loss for a single token is computed based off the negative log likelihood loss from a prediction across the entire corpus. Nevertheless, we do see improvement in both the validation and training loss as the model is trained. In fact, there are no signs of over-fitting, which might imply that a higher capacity model might yield better results. Ultimately the performance of this model is evaluated based on the quality of embeddings which it generated, which will be discussed in the results section.

5.4 Transformer model with Back-Iteration

The transformer model trained significantly faster than the architectures based on recurrent layers. It was our best model which outperformed all the previously reported models. From the different variations of transformer architectures compared in [26], the model with the smallest capacity worked best in our scenario when trained from scratch. The could be because of the scarcity of aligned training data, as larger transformer models either work well on huge amounts of data, or they work well when fine-tuned on already pretrained models on big datasets. Using BERT embeddings helped the performance of our model as compared to using

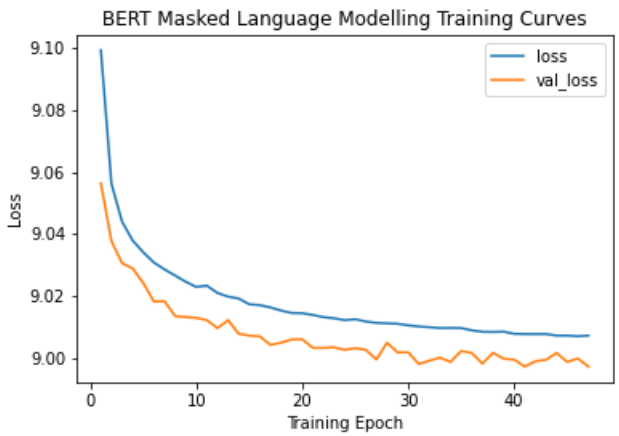

Figure 8: BERT MLM Learning Curve

randomly initialized embeddings by transferring knowledge from language-finetuning task. Back translation improved our BLEU scores further as it added more data, which acts as a regularizer. Model trained on only parallel training data could overfit quickly, while more data avoids or delays overfitting. Different experiments were performed by weighting amounts of real parallel data with varying amounts of synthetic data. Larger amounts of synthetic data helped. We opt to use different ratio (e.g., 1(real):2(synthetic) and 1(real):3(synthetic)). Finally, best results were found considering the ratio of 1(real):6(synthetic).

5.5 Result presentation

As we can see in table 3, the best model is Transformer [26] with the combination of BERT and back-translated data, which got a BLEU score of 17.64. This result was expected, as the transformer model performs better than RNN based models. Regarding the sequence to sequence models, the GRU with Attention performs better than other sequence-to-sequence RNN-based models. This is because of the "Attention" mechanism that gives the encoder the most relevant input sequences to predict the next word. This mechanism cleverly solved the bottleneck issue of the RNN-based models. According to the same table, we can notice that words embedding give a boost to the performance of our models. This is because they give more "meanings" to the words and are able to capture the relationships between each others. The results confirm that, indeed, "Attention is all you need" [26] and that other techniques such as BERT pre-training and Back-translation are useful in a low resource setting. In fact, the transformers model got a BLEU score of 10.42 with BERT pre-training and a BLEU score of 14.86 with Back-translated data in comparison with the BLEU score of 8.18 with vanilla Transformer. We also notice that back-translation performs better, in this context, than BERT pre-training. The reason is probably because, translation tasks need a lot of aligned data to perform better, and back-translation is in fact a good way to add more aligned data to better train the model. Finally, the combination of the two techniques, that is BERT and back-translation gave us the highest BLEU score of 17.64.

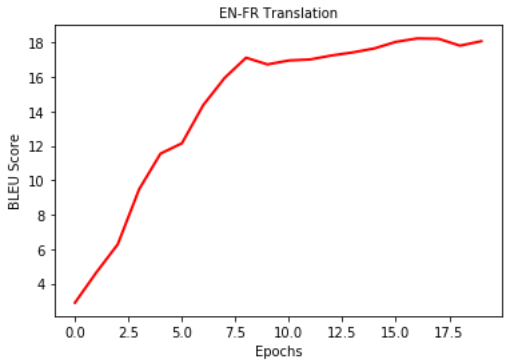

Figure 9: EN-to-FR BLEU Score for transformer model with back-translation

6 CONCLUSION

RNN-style models have multiple problems like computation cost, weakness with long sequences as it causes gradient exploding or vanishing. Also they need a lot of aligned data to perform better. The attention mechanism, and word embedding, gave a good improvement to these models as it give more context to the sequences. Without surprise, the transformer model outperforms all RNN-based models. We believe this result is due to the power of its attention mechanism. in fact, transformer use a multi-head attention that gave not just one context but multiple context to a given ensemble of sequences. In addition, his deepness and his residual connections gave him the ability to deal with very long sequences. Pre-training is a very good way to push the performance ahead like BERT or other similar techniques. We have shown that Transformer performs better with a combination of strategies like pre-training and back translation. However, back translation showed an effective improvement and a good strategy for low resource machine translation as it acts as data augmentation for the model's training and hence improves the performance significantly.

6.1 Future Work

Machine translation can be improved by using beam search [30] to get better predictions in sequence-to-sequence models. Additionally, a regularization technique called label smoothing [25] is also known to be useful to improve accuracy and BLEU score. Averaging last few checkpoints to obtain a single model could also be deployed.

We observed in the loss curve of the BERT model (figure 8) that the model was not over-fitting given the hyper-parameters and number of training epochs we have chosen. It might be beneficial to train a new version of the model with larger hidden layer sizes or additional layers. Additionally, more training epochs might have improved the performance of the model. Also, it would be interesting to explore the impact of using different BERT-derived approaches on performance. Multi-task hierarchical model could also be explored.

ACKNOWLEDGMENTS

This paper was jointly written and edited by all the authors. We thank Krishna for his continuous support throughout this project.

REFERENCES

[1] Team 08. winter 2020. Translation-Team08-IFT6759. https://github.com/notAlex2/Translation-Team08-IFT6759.git

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

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, Article arXiv:1409.0473 (Sept. 2014), arXiv:1409.0473 pages. arXiv:cs.CL/1409.0473

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv e-prints*, Article arXiv:1607.04606 (July 2016), arXiv:1607.04606 pages. arXiv:cs.CL/1607.04606

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv e-prints*, Article arXiv:1406.1078 (June 2014), arXiv:1406.1078 pages. arXiv:cs.CL/1406.1078

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:cs.CL/1810.04805

[7] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding Back-Translation at Scale. *arXiv e-prints*, Article arXiv:1808.09381 (Aug. 2018), arXiv:1808.09381 pages. arXiv:cs.CL/1808.09381

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

[9] Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. 18–24.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[11] Todd Ward Kishore Papineni, Salim Roukos and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *Association for Computational Linguistics* (2002).

[12] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual Denoising Pre-training for Neural Machine Translation. *arXiv e-prints*, Article arXiv:2001.08210 (Jan. 2020), arXiv:2001.08210 pages. arXiv:cs.CL/2001.08210

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *CoRR* abs/1310.4546 (2013). arXiv:1310.4546 http://arxiv.org/abs/1310.4546

[15] Sergey Edunov Nathan Ng, Michael Auli. 2019. Recent advances in low-resource machine translation. https://ai.facebook.com/blog/facebook-leads-wmt-translation-competition/

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[17] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv e-prints*, Article arXiv:1802.05365 (Feb. 2018), arXiv:1802.05365 pages. arXiv:cs.CL/1802.05365

[19] Victor Sanh. 2018. Beating the state-of-the-art in NLP with HMTL. https://medium.com/huggingface/beating-the-state-of-the-art-in-nlp-with-hmtl-b4e1d5c3faf

[20] Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks. *arXiv e-prints*, Article arXiv:1811.06031 (Nov. 2018), arXiv:1811.06031 pages. arXiv:cs.CL/1811.06031

[21] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving Neural Machine Translation Models with Monolingual Data. *arXiv e-prints*, Article arXiv:1511.06709 (Nov. 2015), arXiv:1511.06709 pages. arXiv:cs.CL/1511.06709

[22] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* (2015).

[23] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725. https://doi.org/10.18653/v1/P16-1162

[24] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *arXiv e-prints*, Article arXiv:1409.3215 (Sept. 2014), arXiv:1409.3215 pages. arXiv:cs.CL/1409.3215

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv e-prints*, Article arXiv:1706.03762 (June 2017), arXiv:1706.03762 pages. arXiv:cs.CL/1706.03762

[27] Wikipedia. 2020. BLEU Metric. https://en.wikipedia.org/wiki/BLEU

[28] Wikipedia. 2020. Neural Machine Translation. https://en.wikipedia.org/wiki/Neural_machine_translation

[29] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

[30] W Yonghui, M Schuster, Z Chen, QV Le, M Norouzi, W Macherey, M Krikun, Y Cao, Q Gao, K Macherey, et al. 2016. Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

A APPENDIX

A.1 Attention visualisation

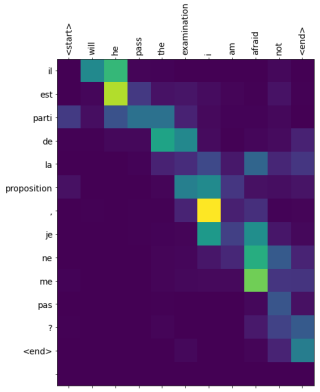

Figure 10: Attention plot of GRU with Attention and Word2Vec embeddings.

A.2 Experiment samples

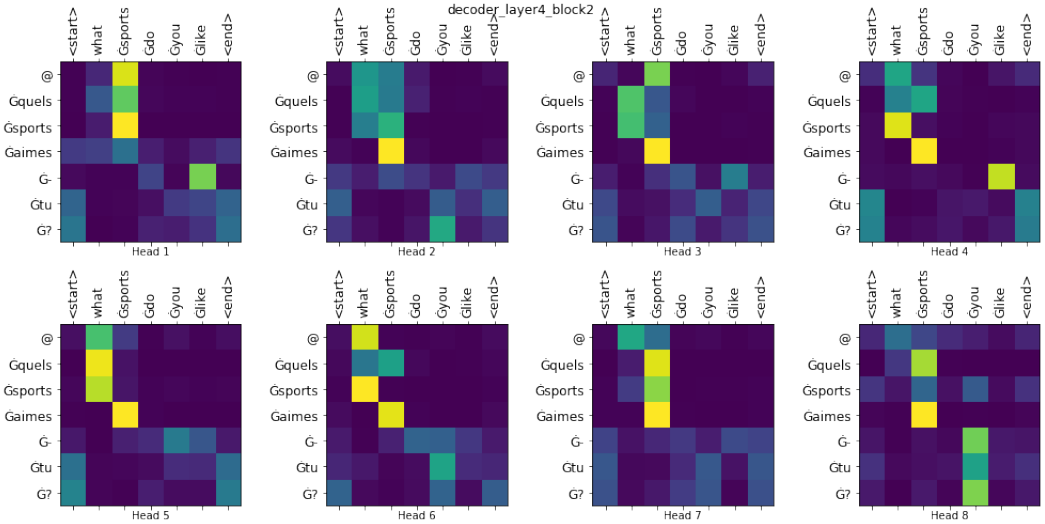

Figure 11: Attention plot of best Transformer model

Source	this leads me to the second question why is intercultural dialogue important
Machine translation	Cela me mène à la deuxième question : pourquoi le dialogue interculturel - il est important ?
Human	Cela m' amène à ma deuxième question : pourquoi le dialogue interculturel est-il important ?
Source	i would also like to thank the minister and the commissioner for their statements and i agree with the commissioner 's statement it is time that fine words were translated into action
Machine translation	Je voudrais également remercier le ministre et le commissaire pour leurs déclarations et je suis d' accord avec le commissaire , car il est temps que les mots de la Commission soient présentés à l' action .
Human	Je voudrais également remercier la ministre et la commissaire pour leurs déclarations et je suis d' accord avec la commissaire lorsqu' elle dit qu' il est temps que les beaux discours se traduisent en actes .
Source	third minorities must benefit from the protection of the law
Machine translation	Troisièmement , les minorités doivent bénéficier de la protection de la loi .
Human	Troisièmement , les minorités doivent bénéficier de la protection de la loi .
Source	small and medium - sized enterprises are essential in the creation and maintenance of employment
Machine translation	Les petites et moyennes entreprises sont essentielles dans la création et la préservation de l' emploi .
Human	Les PME sont essentielles pour la création et la préservation de l' emploi .

Table 4: Sample translations of our best performing Transformer model.