IFT6759 - Project 2, Team 1
Low-Resource Machine Translation
Mila - Université de Montréal

Chih-Chao Hsu
chih-chao.hsu@umontreal.ca

Carolyne Pelletier
carolyne.pelletier@umontreal.ca

Akshay Singh Rana
akshay@umontreal.ca

Ying Xiao
ying.xiao@umontreal.ca

Yifan(Andy) Bai
yifan.bai@umontreal.ca

1 Introduction

This paper documents the project, Low-Resource Machine Translation, which aims to implement a neural machine translation (NMT) system capable of translating English into French using 11k aligned examples and 2 unaligned monolingual corpora 474k for each language. Training an NMT system requires aligned pairs of sentences, such as *"Hello!"* and *"Bonjour!"*, where capitalization and punctuation are matched. "Low-Resource" implies low availability of such examples, this case being 11k. Moreover, source English texts are uncapitalized and unpunctuated, whilst French texts are propely formatted. The system needs to generalize and translate with proper capitalization and punctuation, such as *"hello translate this thanks"* to *"Salut! Traduisez cela, merci!"*. This requires models capable of incorporate contextual information within each language for accurate translations.

To address these challenges, we utilize techniques such as word embeddings, attention mechanism, sequence-to-sequence (Seq2Seq) and transformer models. To deal with low-resources scenario, we leverage unaligned examples using techniques such as pre-training to train the models to capture contextual information. An exploratory data analysis is completed to obtain statistics about aligned and unaligned corpus, providing insights helpful for understanding. We follow this with extensive literature review, covering relevant papers that inspire our choices. Next, we illustrate methodology in details which includes data pre-processing pipeline, word embeddings, description of models among others. Lastly, we present experiment results with detailed analysis and discussions.

Source code for this project can be found in our Github repository.

1.1 Evaluation Metric

The system will be evaluated on a separate blind test set based on **bilingual evaluation understudy (BLEU)** score. It aims to evaluate translation quality based on correspondence between translated results to source texts. Scores are calculated for individual translated sentences by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. It uses a modified form of precision to compare a candidate translation against multiple reference translations. The metric modifies simple precision since NMT systems tend to generate more words than are in a reference text. However, restrictions such as various decoders used and flags for various evaluation scripts cause tokenization processed being handled differently. We also need to handle test sets for scoring. These reasons prompt the need for a standardized implementation, of which this project adopts, **sacreBLEU**. It produces the official WMT scores but works with plain text. Comparing with the standard version, it standardizes handling of tokenization process, which computes **BLEU** score and outputs detokenized results, as well as downloading and management of test sets. This helps us to get an output prediction file with translated sentences, which will be scored against a reference target file.

2 Data Analysis

A collection of unique tokens, being words, subwords or characters, is called a vocabulary, which have effects on vocabulary size, sentence length and out-of-vocabulary tokens (OOVs). The longest token string is a word, and word-based vocabulary is large in size hence risk of being computationally expensive. It may also increase number of training OOVs. The tradeoff is shorter sentence lengths, hence alleviating drawbacks due to vocabulary size and permitting smaller embedding sizes. To keep analysis consistency and build baseline intuitions, we used tokenizer and punctuation remover provided for pre-processing. Punctuations are removed for aligned examples while the unaligned are tokenized and punctuation-removed. The vocabularies are word-based based on the entire corpus and non-alphabet-containing tokens are considered as OOV.

2.1 Language-wise Comparison

Figure 8 in the Appendix shows boxplot of sentence lengths distributions. Whiskers represent outliers drawn from 1.5 interquartile range (IQR). All datasets have similar means and majority of sentences are within the same length range, implying potential of using uniform embedding sizes thus saving running time. For each language, the mean/standard deviation figures are nearly identical, and longest sentence is nearly twice in unaligned data compared to aligned, as summarized in Table 1.

	Unaligned EN	Aligned EN	Unaligned FR	Aligned FR
Vocabulary/OOV Size	57293/2715	13657/341	80769/2545	18222/327
OOV Words Percentage(%)	1.68	1.73	1.56	1.6
Shortest/Longest Sentence	1/180	1/95	1/224	1/105
SL Mean/Standard Deviation	18.61/9.76	18.67/9.86	20.56/10.82	20.71/10.91

Table 1: Vocabulary and Sentence Length(SL) Information Summary

2.2 Aligned and Unaligned Examples

Figure 1 show base-10 log transformation of word count and direct comparisons of sentence lengths distributions respectively, between two languages within unaligned and aligned datasets respectively. Figure 1 gives a hint to the size of vocabulary we could use for unaligned and aligned data, as words with high counts dominate the total, this case around 20 000 and 5 000 respectively. Both figures show within each set, two languages are well-aligned in both word count and sentence length distribution, which is a confirmation of Figure 7 and Figure 8 respectively, both found in the Appendix.

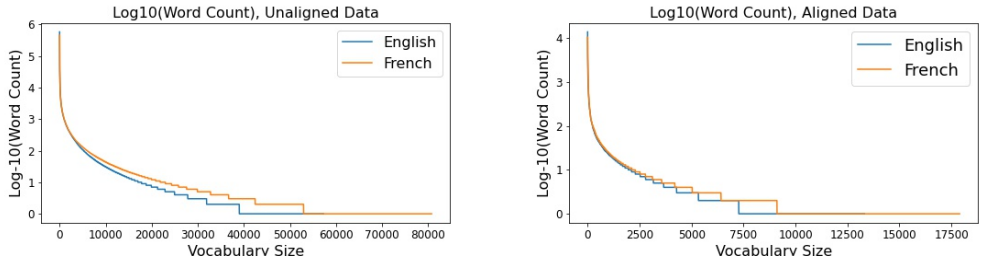

Figure 1: Log 10 Transformation of Word Count, *Left*: Unalgined and Aligned *Right*: Aligned Datasets

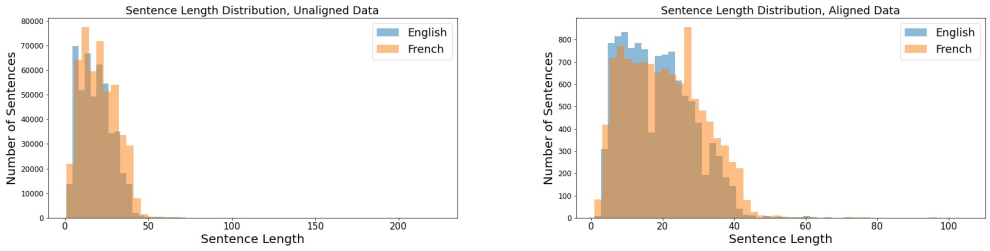

Figure 2: Sentence Length Distribution, *Left*: Unalgined and Aligned *Right*: Aligned Datasets

3 Literature Review

Probabilistic language models [2] are integral part of NMT systems but limited in functionality as they limit their context to n-1 in n-grams models. NMT, unlike phrase based language models, considers the entire input sentence as a unit for translation and does away with the input sentence to be broken down into words. Recurrent neural networks(RNN) (or its different architecture like LSTM [11], GRU [4]) can capture long-distance dependencies in language but problems like vanishing gradient for large sentences and variable input sequence arise. Along this line of research of using neural networks for machine translation, [3] proposed a model called RNN Encoder–Decoder that consists of two RNNS. One RNN encoder maps a variable-length source sequence to a fixed-length vector representation, and the other decodes the vector representation back to a variable-length target sequence. The encoder and decoder of proposed model are jointly trained to maximize conditional probability of a target sequence given a source. However, a potential issue is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences. [5] showed that performance of a basic encoder–decoder deteriorates rapidly as length of an input sentence increases. Using a fixed-length vector is a bottleneck in improving performance of this basic architecture.

To overcome this, [1] proposes an extension to the encoder–decoder model which learns to align and translate jointly. During generation of a word in translation, the model looks for positions in a source sentence where the most relevant information is concentrated. It then predicts a target word based on context vectors associated with these source positions and all the previous generated target words. This frees the model from having to squash all the information of a source sentence regardless of length into a fixed-length vector, as the decoder can rely on all input vectors by using powerful mechanism namely attention. Another advantage is it not only increases translation accuracy, but also makes it easier to tell which words are translated into which in the output and thus making it possible to handle unknown words more elegantly, performing unknown word replacement. Every time the decoder chooses the unknown word in the output, we can look up the source word with the highest attention and output that word's translation from a different dictionary instead of the unknown token.

The groundbreaking Transformer model is introduced in "Attention is All you Need" [16] and is an improvement on the previous sequence transduction models that are based on complex recurrent neural networks in an encoder-decoder configuration. This new model improves Seq2Seq modelling through self-attention and positional encoding, and dispenses of the need for recurrence. Therefore, the biggest benefit comes from how the Transformer model lends itself to parallelization, which increases training efficiency. The model achieved SOTA for single-model scores on the WMT 2014 English-to-French translation task on BLEU with 41.8 after training for 3.5 days on eight GPUs, largely reducing the training costs of the best models at the time (June 2017). All in all, the translation quality is better with the Transformer model, and it requires less computation to train.

BERT [6], which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers, is the first deeply bidirectional, unsupervised language representation model. Its major contributions are in the deep bidirectional nature of its architecture and its novel pre-training tasks (Masked Language Modeling and Next Sentence Prediction). It is a language understanding model pre-trained on a very large corpus of unlabelled data, namely Wikipedia (2.5B words) and BookCorpus (800M words). It learns to represent the language and can be fine-tuned to solve a specific task. BERT [6] outperforms state-of-the-art models (at the time of its introduction) on many NLP tasks, most of them being from the GLUE benchmark. Best practices for training BERT model has been recently proposed in [13], where the roBERTa model builds on BERT and modifies key hyperparameters. It removes the next-sentence pre-training objective and trains with much larger mini-batches and learning rates. For the translation task specifically, roBERTa [13] can be pre-trained on the unaligned data and used as an embedding layer in a encoder-decoder architecture to either extract strong contextual representations of words (feature extractor) or fine-tuned on the translation task.

Back-translation converts monolingual target into the source language, providing synthetic source sentences as additional training data, as well as ensuring output layer remains sensitive to source context thus preserving quality of parameters [9]. This is useful when existing aligned examples are limited. Another paper [15] also describes an experiment whereby 1 million parrallel sentences of authentic (human translation) data and a baseline EN $\rightarrow$ DE model on the same data set with source and target sides swapped around. The latter model is used for back-translation to create

synthetic datasets. Three scenarios were investigated. Firstly, training with authentic data only that provides a baseline. Secondly, training with only synthetic, back-translated data, whose scores kept rising when increasing the amount, from 0.229 at 1 million to 0.2363 at 3.5, a 3.2% increase. Since then the model saturated. Lastly, training with combination of the authentic and synthetic data. It was discovered that increasing authentic data from 1 million to 3 increases BLEU score by 7.4%. Adding large amount of synthetic data increases score but tops out quickly, and increasing beyond certain point tails off performance, as 3.5 million such examples resulted in a dropoff of 1.2% compared to 3.

4 Methodology

4.1 Data Pre-processing

In order to optimize the translation task, we performed data preprocessing on unaligned and aligned data. For all data in both French and English, we stripped each sentence of leading and trailing whitespace and added <*start*> and <*end*> tokens so that Seq2Seq models know when to start and stop predicting. For unaligned English data, we also lowercase the sentences so that they match parallel data and made sure to remove all the punctuation. For unaligned French data we add space between the punctuation in order to easily separate the tokens. A vocab size of 20000 was selected for both English and French since it encapsulates the highest frequency words shown in Fig 1 while keeping a smaller vocabulary size. We perform an 80/20 split (train/validation) on the raw data as opposed to a split on the tokenized data. This ensures that the validation set is not included in the English input tokenizer's vocabulary to help mimicking the evaluation conditions of the hidden test set. The same pre-processing technique was used in all our models.

4.2 Word Embeddings

Word embeddings represent words being mapped to vectors of real numbers. This is realized using methods such as word-to-vectors (word2vec), global vectors (GloVe). A notable recent change is shifting from learning unconditional word vectors (where word representation is the same globally) to contextualized ones, where representation of word is dependent on the context of the sentence (roBERTa). The goal here is to exploit unaligned samples, trying to capture better information that helps improving weight initialization of embedding layers. We experimented word2vec with embedding sizes of 128 and 256 by creating embedding matrices (i.e. vocabulary of that size) and used it to initialize embedding layer weights. We also pre-trained a roBERTa model [13] on the unaligned English samples and used it as an embedding layer in a encoder-decoder architecture to either extract strong contextual representations of words (feature extractor) or fine-tune it on the translation task using the aligned samples. We first pre-trained the roBERTa model with the following hyperparameters: hidden_size of 768, num_attention_heads of 12, num_hidden_layers of 6, sequence_max_length of 512, a vocabulary size of 52 000, and a learning_rate of 1e-4 with 10 train epochs and a batch_size of 16. These results were not great as seen in experiment 03 of Table 3, so we re-pre-trained roBERTa with a decreased hidden_size of 252, a decreased max_sequence_length of 128 and a decreased vocabulary of 20 000. This improved our results as seen in experiments 04 and 05 of Table 3 and decreased training time as well.

4.3 Sequence-to-Sequence Models (Seq2Seq)

Recurrent neural networks(RNN) (or its different architectures like LSTM [9], GRU[4]) have the ability to capture long-distance dependencies in language but problems like vanishing gradient for large sentences and variable input sequence arise. Seq2seq models [8] with an encoder-decoder architecture are a good choice for a translation task since they can map sequences of different lengths to each other. The encoder maps a variable-length input sentence to a fixed length vector representation. All models we experimented with are Seq2Seq models in an encoder-decoder architecture. Although, another problem of this fixed length bottleneck for longer sequences is solved using attention.

4.3.1 Encoder-Decoder Gated Recurrent Units (GRUs) with Attention

This Seq2Seq model uses GRUs [4] as its encoder and decoder, and implements Bahdanau attention [1] to alleviate bottleneck issue of information from encoder being compressed into one hidden state and passed on to the decoder. See Figure 3 for the general architecture. Out of all implementations of encoder-decoder architecture, GRUs are chosen since they form a healthy medium between capacity and efficiency. RNNs are the most efficient but stumle on long sequences due to vanishing/exploding gradients, and GRUs are more efficient than LSTMs with one fewer gate and less parameters.

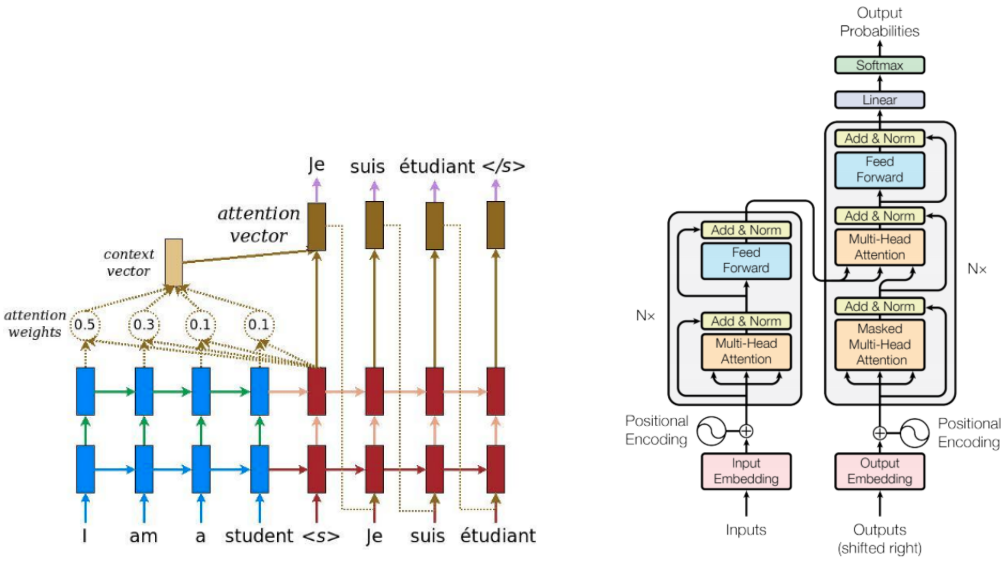

Figure 3: *(Left)*: Encode (blue) - Decoder (red) Seq2Seq model with Bahdanau attention for English to French translation task. During training, input is passed through encoder which returns an encoder output of shape $(batch\_size, max\_length, hidden\_size)$ and an encoder hidden state of shape $(batch\_size, hidden\_size)$. Both of these are passed to decoder along with decoder input (which is the <start> token). Then, the decoder returns predictions and decoder hidden state, and predictions are used to calculate loss and decoder hidden state is passed back into the model. To decide the next input to the decoder, teacher forcing is used (technique where the target word is passed as the next input to the decoder). Finally, gradients are calculated which are applied to the optimizer (Adam) and backpropagated throughout the model. Image sourced from [14].
(Right): The transformer model follows a similar pattern of Seq2Seq networks. The input sequence is passed through encoder layers that generates an output for each word/token in the sequence. The decoder attends on encoder's output and its own input (self-attention) to predict the next word. During training, this model too uses teacher forcing. Image sourced from [16]

Data is pre-processed via method described in the Data Pre-processing section. Then it is tokenized using the Keras Tokenizer or the roBERTa Tokenizer. Keras Tokenizer [1] takes raw texts and turns it into space-separated sequences of words, which are split into tokens (words) and indexed (mapped to an index). For roBERTa [2], we trained a Byte-pair encoding (BPE) tokenizer from vocabulary built during pre-training on unaligned data. We chose a byte-level BPE (rather than a WordPiece tokenizer like in BERT [6]) as it builds its vocabulary from an alphabet of single bytes, so all words will be decomposable into tokens (there will be no unknown <unk> tokens). The Encoder has an embedding layer which turns indices into dense vectors of fixed size. The Encoder also has a GRU layer where initialization for linear transformation of recurrent state is glorot_uniform. The Decoder has an embedding layer, a GRU layer (with same recurrent_initialization as Encoder) along with a Dense layer. The embedding layer for both the Encoder-Decoder is initialized uniformly and is of shape (vocabulary_size and embedding_dimension) which are both hyperparameters. Note that we also experimented with switching out the embedding layer with a roBERTa model [13] pre-trained on the unaligned data in order to capture the strong contexual word representations [13]. We tried feature extraction and fine-tuning of the roBERTa model, and all results can be seen in Table 3. Optimizer used in this Seq2Seq model is Keras implementation of Adam algorithm [12] where its default hyperparameters were used ($learning\ rate = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-7}$). Loss used is Keras implementation of Sparse Categorical Cross-Entropy [3] which computes cross-entropy loss

[1] `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer`
[2] `https://huggingface.co/transformers/model_doc/roberta.html#robertatokenizer`
[3] `https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy`

between labels (gold French word) and predictions (predicted French translated word from model). All hyperparameters and their values are presented in Tables 2 and 3.

4.3.2 Encoder-Decoder Transformers

Transformer models handle variable-sized input using stacks of self-attention layers. This provides an advantage over RNNs as outputs can be calculated in parallel and it can learn long-range dependencies which is essential in translation task. However, to incorporate the temporal nature of our task, we need to add positional embedding to the model, to give information about relative position of words in the sentence. Look ahead masks are also created to mask future tokens in the sequences as the entire sentence is given as input. Multi-head attention has four parts: Linear layers and split into heads, Scaled dot-product attention, Concatenation of heads and Final linear layer. Each multi-head attention block gets three inputs; Q(query), K(key), V(value). These are put through linear (Dense) layers and split up into multiple heads. Instead of one single attention head, Q, K, and V are split into multiple heads because it allows the model to jointly attend to information at different positions from different representational spaces. Attention function used by the transformer takes three inputs: Q, K, V and equation used to calculate the attention weights is $softmax(\frac{Q \cdot K^\top}{\sqrt{d_k} \cdot V})$

This transformer model (as shown in Figure 3) follows the same Seq2Seq model where input sentence is passed through encoder layers that generates an output for each word/token in the sequence. Then decoder attends on encoder's output and its own input (self-attention) to predict the next word. The Encoder consists of Input Embedding, Positional Embedding, and 4 encoder layers. The input is put through an embedding which is summed with positional encoding. Output of this summation is the input to the encoder layers and output of the encoder is the input to the decoder. Each encoder layer consists of two layers: Multi-head self-attention and a fee-forward neural network. Each of these sub layers has a residual connection around it followed by a normalization layer. Residual connections help in avoiding vanishing gradient problem in deep networks. Similarly, the decoder consists of Output Embedding, Positional Encoding, 4 decoder layers. The target is put through an embedding which is summed with the positional encoding. Output of this summation is the input to decoder layers and output of the decoder is the input to the final linear layer. Input embedding used in both encoder and decoder was extracted from a word2vec model trained on unaligned samples for 5 epochs with a window size of 3 for both languages. Although the transformer is an auto-regressive model and makes predictions using its own past prediction. Teacher forcing is used for training where true output is passed to the next time step regardless of what the model predicts at the current time step. Cross entropy loss was used and padding in the sequence was not used to calculate it. We used Adam optimizer with a custom learning rate, expressed as $d_{model}^{0.5} \cdot min(step\ number^{0.5}, step\ number \cdot warmup\ steps^{-1.5})$ [16].

4.4 Iterative Back Translation

An effective way to handle low amount parallel samples is making synthetic samples using back-translation on unaligned data, which in turn can be used in combination with the existing aligned data for training[7]. Table shows adding synthetically created parallel samples increases BLEU score. We incremented the back translation data slowly to observe the trends during validation. Considering synthetic data is generated by an imperfect model, we tried to limit amount of data being added, as it may negate the benefits from original aligned data and degrade performance at some point . We tried different data mixing ratios starting from 1:1 where both aligned samples and synthetically generated samples are of same amount. Eventually, we realized that synthetic data does not degrade performance even if it becomes largely dominant. One can also simultaneously improve the quality of synthetically generated parallel samples by doing back translation iteratively[10]. We created quality synthetic samples based on the model, which itself was trained using synthetic samples in its training data. This process can be iterated multiple times to produce quality results.

Models	Embedding Size	Hidden Size	Batch Size	Layers
GRU	variable	512	64	1
T_S	128	512	128	4
T_M	256	1024	64	5
T_L	768	2048	16	5

Table 2: Different models sizes experimented. Note that when the GRU model fine-tunes roBERTa in its encoder, the Batch Size decreases to 16.

ID	Models	Embedding-Size	Ratio	BLEU
01	GRU	Glorot-100	-	2.68
02	GRU	W2V-256	-	3.40
03	GRU	roBERTa_FE-768	-	0.72
04	GRU	roBERTa_FE-252	-	1.80
05	GRU	roBERTa_FT-252	-	2.40
06	T_S	-	-	7.20
07	T_M	-	-	7.48
08	T_S	W2V	-	8.38
09	T_M	W2V	-	8.40
10	T_M	roBERTa_FE	-	6.70
11	T_L	roBERTa_FE	-	3.32
12	T_S	W2V	BackTrans - 1:1	10.56
13	T_S	W2V	BackTrans - 1:4	11.47
13	T_S	W2V	BackTrans - 1:8	12.03
14	T_S	W2V	Re-BackTrans - 1:8	13.05
15	T_M	W2V	BackTrans - 1:1	10.64
16	T_M	W2V	BackTrans - 1:4	11.78
16	T_M	W2V	BackTrans - 1:8	12.13
17	T_M	W2V	Re-BackTrans - 1:8	**13.70**

Table 3: BLEU on 20% validation set across various models and settings. roBERTa_FE stands for roBERTa used as a Feature Extractor (weights frozen) and roBERTa_FT stands for roBERTa being fine-tuned. Ratio is the amount of original aligned data and the synthetically generated data. W2V stands for word2vec and the embedding layers were allowed to train even after initializing with embeddings from word2vec.

5 Results and Discussion

5.1 Encoder-Decoder (GRUs) with Attention

The Seq2Seq model with GRUs as the encoder-decoder took time to run (10 minutes per epoch on 1 GPU). The minimum number of epochs to stop training this model was determined to be 20. This number was determined sufficient to test future models, since increase in BLEU does not justify the longer training time as can be seen in Table 4. Other Embedding and Hidden Sizes tested were 256 and 1024 respectively, but these proved to be too large since training would take about 20 minutes per epoch. In Experiments 03, 04, and 05 3 we tried to take advantage of roBERTa's strong contextualized representation of words in order to help with translation task. Two roBERTa models were pre-trained on unaligned data (one each for English and French). In Experiment 03 3 we incorporated English roBERTa model in the Encoder of Seq2Seq model as a feature extractor (no fine-tuning, weights are frozen), but unfortunately it decreased our BLEU score significantly. One hypothesis is that in roBERTa's implementation, models are pre-trained on approximately 6.4 million examples (WikiPedia and Bookcorpus) [13] which is significantly more data than ours at 474k. We simply do not have enough training data to capture good contextualized representations of the words. To investigate whether the problem was roBERTa model being too large (768 hidden size) or vocabulary being too large (52,000), we re-pre-trained roBERTa on unaligned data with a smaller size of 252 and decreased the vocabulary to 20 000, and the score can be seen in experiment 04 3. Since we changed both values at once, it is hard to disentangle whether it was the decrease in capacity or vocabulary that contributed to the better score. In future works, we recommend to investigate these separately. We also tried fine-tuning the 252 hidden roBERTa model in the encoder with 20,000-vocabulary to see if this would improve score, and it did as seen in Experiment 05 3. Note that when fine-tuning, we decreased batch size to 32 to avoid out-of-memory issues, which increased training time and is a limitation of fine-tuning pre-trained models. As expected, fine-tuning roBERTa gave us better scores than using it as a feature extractor since the loss from the encoder-decoder model is backpropagated through the model and further optimizes the whole training procedure.

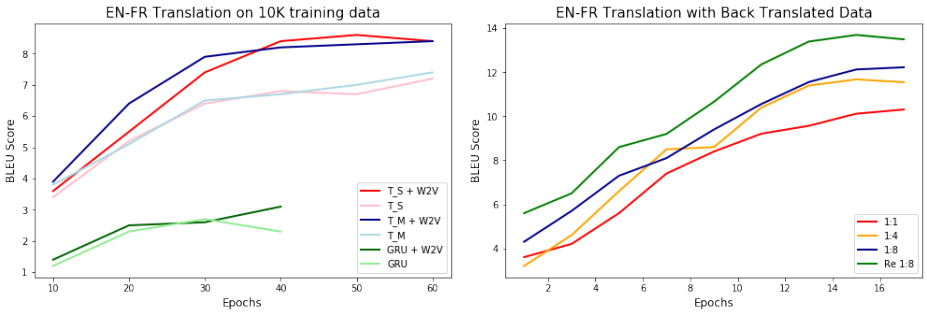

Figure 4: *(Left)*: Comparing BLEU scores for different models trained only on the original aligned samples. Scores calculated on the validation set with a 80-20 split.
(Right): Comparing BLEU scores with different ratios of synthetically generated back-translated data on T_M model.

5.2 Encoder-Decoder - Transformers

Table 3 shows results of using different transformer model sizes, with and without using word2vec embeddings. Word2vec model was trained on 400k unaligned samples for both languages. Models using word2vec performed marginally better than those without it. Moreover, higher BLEU scores were achieved quickly when weights were initialized using word2vec. When these embedding layers were kept trainable, we saw a noticeable improvement in scores. We also trained a roBERTa model on unaligned data and used it as a feature extractor to extract embeddings for each token. We expected it to outperform the word2vec model but the results were quite the opposite due to the small training size used to train it. As explained in previous section, due to a lack of resources, we did not experiment with fine-tuning the roBERTa model. The model with an embedding size of 256, a hidden size of 1024 and a batch size of 64 performed the best with a BLEU score of 8.4 on validation data created using 80/20 split. To obtain synthetic aligned data, we trained a French-English model on 11k samples and attained BLEU score of 8.9. A similar blue score was not expected albeit it is comparatively easier to translate French inputs with proper casing to English outputs without either. This implies that missing punctuations in inputs for original English to French task would not be a major challenge for the model, therefore we never built a model aiming to improve punctuations.

90000 synthetic parallel samples were created using French-English model and used with existing aligned samples to train original English-French task. We tried to limit usage synthetic data to avoid it negating benefits from existing, perfect aligned examples. We obtained good results with BLEU score of 10 when the ratio of data was 1:1. To investigate effects of synthetic data, we kept adding it to the mix and scores kept improving. Surprisingly, we did not reach the expected tipping point where results deteriorate; adding data further improved results, albeit such improvement is not linear to addition amount. We also tried iterative back-translation to further improve French-English model to generate quality synthetic samples. The English-French model with BLEU score of 10 was used to generate synthetic samples for French-English model, which was then used to generate samples for English-French model. This process was repeated twice to improve the quality of generated sample, and it increased BLEU scores. The best model yielded a score of 13.7 which was trained using synthetic samples generated from an French-English model with BLEU score of 15.3. The ratio of original aligned samples to that of synthetically generated was 1:8. We believe that adding more data by iteratively back-translating on top of each other can improve the score further.

6 Conclusion

In this project, we investigated performance of a English to French NMT system using models with differing architectures, including GRUs with attentions and transformer models. Low availability of aligned examples requires us to utilize unaligned data with large amount of back-translated synthetic data to increase traing size, thus improving scores. Despite low quality, the model learns to construct sentences well and produced plausible results. We also showed while translation performance improves with additional synthetic data, performance tends to saturate when balanced is tipped too far in favour of synthetic data. However, we could not find a point where performance tips off. Iterative back-translation was also explored where system improved with quality of synthetic data.

There are a few areas in which we would like to investigate further to improve performance. To begin with, Beam Search decoder would be preferred over Greedy since it considers multiple best options based on beam width using conditional probability. This is especially useful when we have synthetic samples in the training data. In addition, a multi-task system could be useful where BERT is trained on masked modelling and simultaneously improved by joining it with a decoder for translation task. We could also experiment with adding and fine tuning roBERTa in the decoder of GRU seq2seq model. Last but not least, we could explore various other tokenizers such as byte-processed ones to better handle OOV words. Since byte-level tokenizers start building its vocabulary from an alphabet of single bytes, all words will be decomposable into tokens and the absence of the unknown <unk> token might improve our scores. Averaging weights from various checkpoints of the model could also prove to be beneficial. Training the seq2seq model with GRU's as the encoder-decoder took a lot more time than the Transformer encoder-decoder architecture, which proved to be one of the limitations of the GRU model.

References

[1] Dzmitry et al. Bahdanau. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. URL: http://arxiv.org/abs/1409.0473.

[2] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944966.

[3] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." In: *EMNLP*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. ACL, 2014, pp. 1724–1734. ISBN: 978-1-937284-96-1. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2014.html#ChoMGBBSB14.

[4] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[5] KyungHyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259 (2014). arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

[6] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[7] Sergey Edunov et al. "Understanding Back-Translation at Scale". In: *CoRR* abs/1808.09381 (2018). arXiv: 1808.09381. URL: http://arxiv.org/abs/1808.09381.

[8] Ilya Sutskever et al. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[9] Rico Sennrich et al. "Improving Neural Machine Translation Models with Monolingual Data". In: *CoRR* abs/1511.06709 (2015). arXiv: 1511.06709. URL: http://arxiv.org/abs/1511.06709.

[10] Vu Cong Duy Hoang et al. "Iterative Back-Translation for Neural Machine Translation". In: URL: https://www.aclweb.org/anthology/W18-2703.

[11] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[12] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. URL: http://arxiv.org/abs/1412.6980.

[13] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[14] *Neural machine translation with attention*. https://www.tensorflow.org/tutorials/text/nmt_with_attention. Accessed: 2020-02-27.

[15] Alberto Poncelas et al. "Investigating Backtranslation in Neural Machine Translation". In: *CoRR* abs/1804.06189 (2018). arXiv: 1804.06189. URL: http://arxiv.org/abs/1804.06189.

[16] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

7 APPENDIX

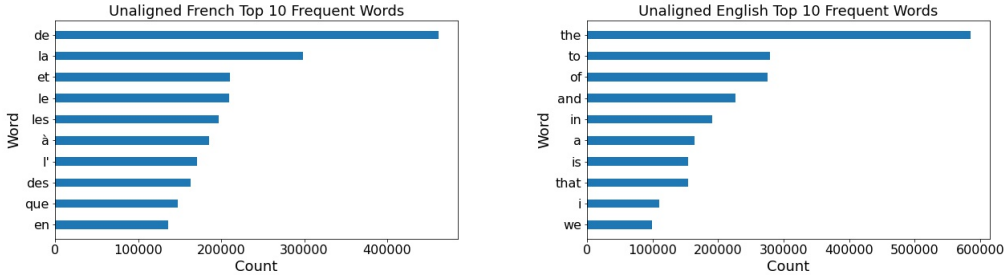

Figure 5: Top-10 Most Frequent Words in Unaligned Data, French (*Left*) and English (*Right*)

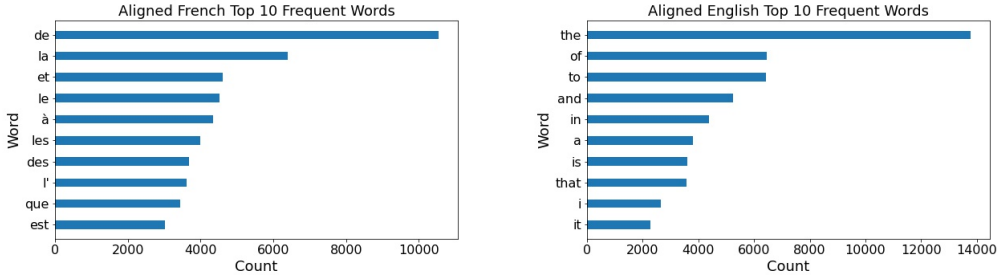

Figure 6: Top-10 Most Frequent Words in Aligned Data, French (*Left*) and English (*Right*)

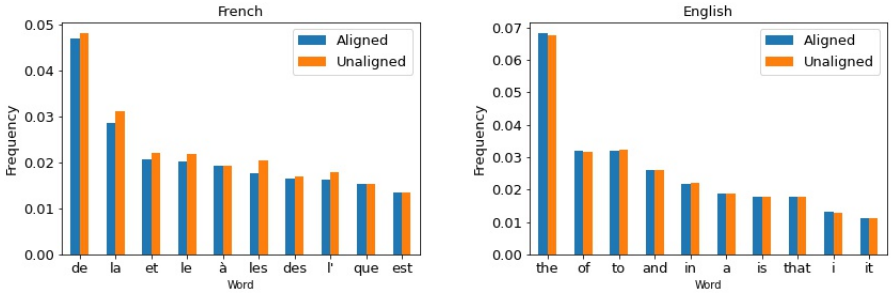

Figure 7: Top 10 Frequent Words in Each Language, French (*Left*) and English (*Right*)

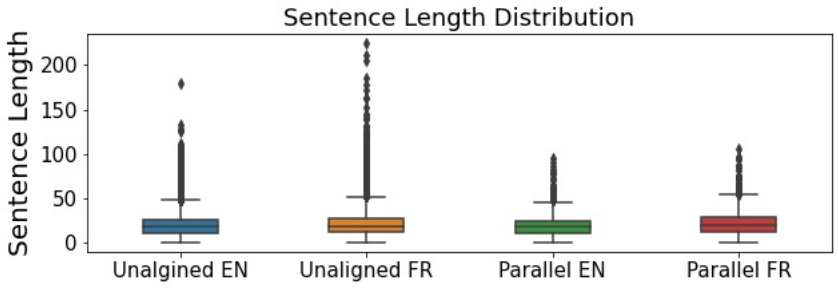

Figure 8: Sentence Lengths Distributions

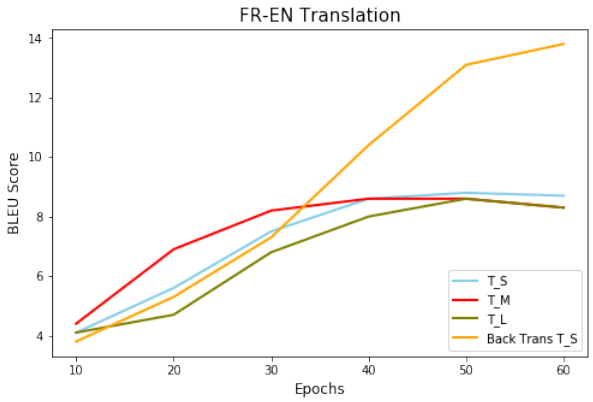

Figure 9: **French-English Translation Model** built for generating back translated synthetic parallel samples from unaligned data. Back-Trans T_S model is build using back translated samples generated from en-fr model. This was repeated twice to improve quality of synthetic samples.

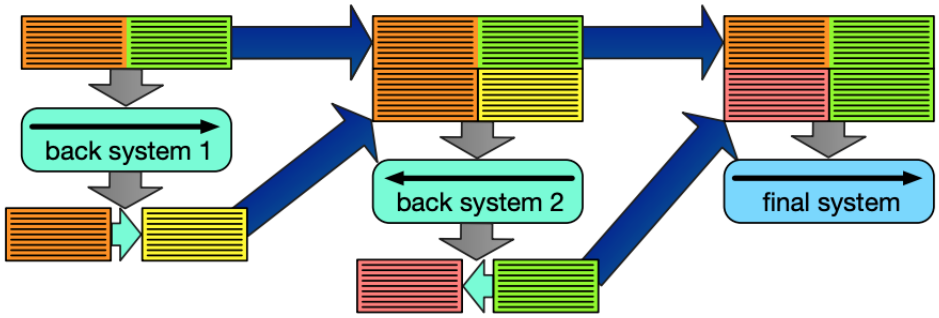

Figure 10: **Re-Back Translation:** After training a system with back-translated data (back system 2 above), it is used to create a synthetic parallel corpus for the final system. Image sourced from [10]

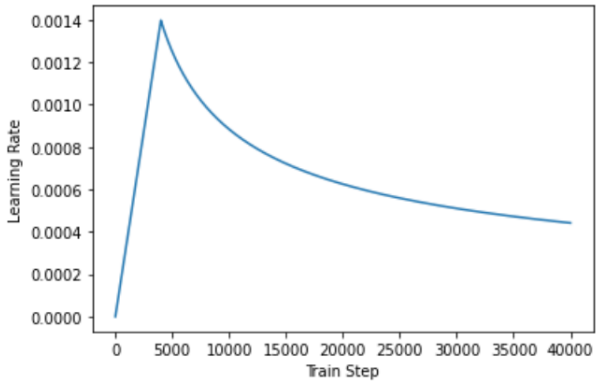

Figure 11: **Learning Rate Schedule** used in the transformer model with Adam Optimizer.

ATTENTION MAPS

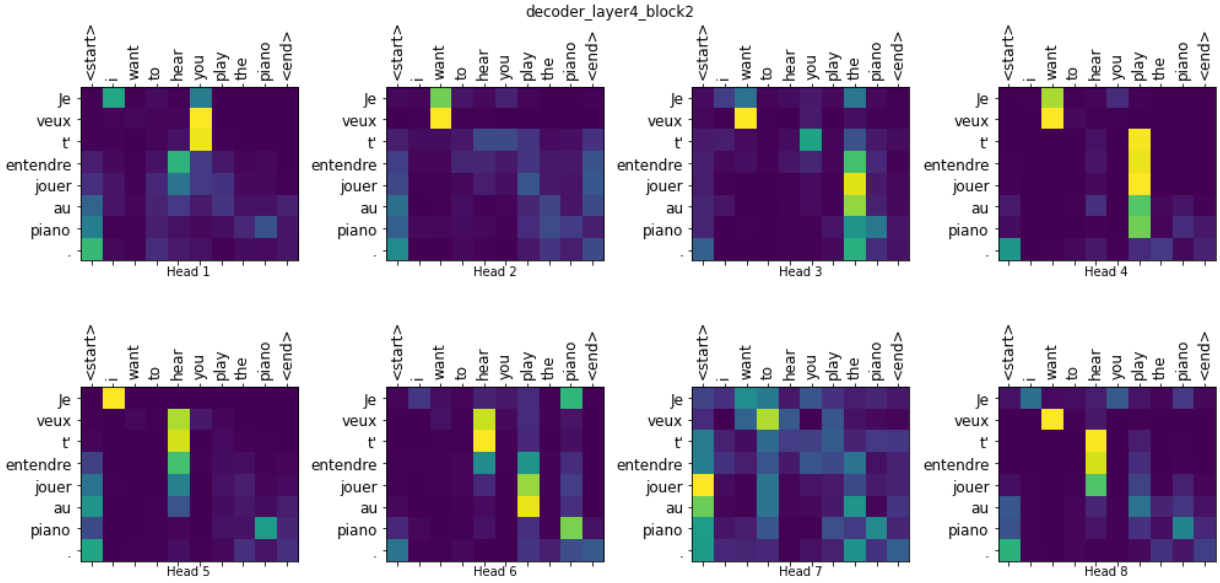

Figure 12: Attention map for second block of the last layer of Decoder

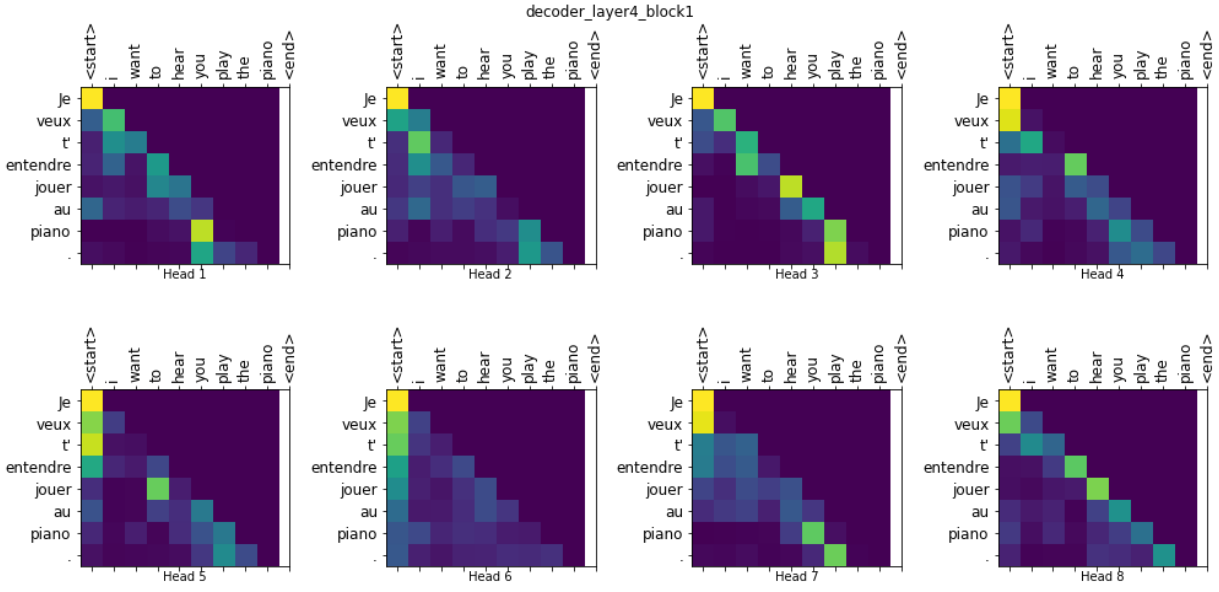

Figure 13: Attention map for first block of the last layer of Decoder