Low-Resource English-to-French Machine Translation using Deep Neural Networks

Philippe Beardsell
Université de Montréal & Mila
20139766

Yassir El Mesbahi
Université de Montréal & Mila
20107171

Jérôme Labonté
Université de Montréal & Mila
20144437

Marie St-Laurent
Université de Montréal & Mila
657930

1 Introduction

1.1 Summary

Recent trends in natural language processing (NLP) and neural machine translation (NMT) indicate that state-of-the-art performance is achieved by ever larger models trained on ever growing data corpora. For NMT to reach human parity, high-resources language pairs (e.g., dozens of millions of parallel sentences) are often required [Hassan et al., 2018, Kocmi, 2020]. However, training large data-hungry models proves challenging in scenarios where the amount of parallel text data (e.g., sentence pairs) is limited, for example when translating rare languages or documents from highly specialized domains. Specific measures have recently been introduced to perform NMT successfully in so-called low-resource situations. For the current project, we trained deep neural networks to perform English-to-French NMT on a small corpus of aligned examples (11k sentence pairs). To simulate a low-resources situation, we used no additional data, except for separate unaligned corpora from the source (English) and target (French) language (474k sentences, respectively). We also used no models or embeddings that were pre-trained on data sources external to those provided.

We relied on two model architectures commonly used in NLP: the Gated-Recurrent-Units (GRU) model with an Attention module [Bahdanau et al., 2014], and the Transformer model [Vaswani et al., 2017]. We trained these models to generate formatted (capitalized and punctuated) French sentences from unformatted English sentences using the small corpus of aligned pairs. Source and target sentences were broken down into tokens by language-specific tokenizers trained on a combination of aligned and unaligned data. Models were trained with the cross-entropy loss function, and performance was evaluated using the bilingual evaluation understudy (BLEU) score [Papineni et al., 2002] computed on a 1k validation set we held-out from the provided aligned sequences. BLEU scores were calculated using the SacreBLEU implementation introduced by Post [2018].

To make-up for the sparsity of aligned pairs, we implemented two strategies using the Transformer model architecture. First, we pre-trained Word2vec monolingual embeddings [Mikolov et al., 2013] using separate language modeling tasks (one per language) and the unaligned monolingual corpora. The pre-trained embeddings were inserted into the encoder and decoder portions of a Transformer that was then trained to translate aligned sequences. We tried two approaches : fixing the embedding weights and fine-tuning them on the parallel data.

Second, we used back-translation to generate synthetic (unformatted) English sentences from the monolingual French corpus [Guzmán et al., 2019, Sennrich et al., 2015a]. The idea was to mix these synthetic pairs with the real pairs in order to augment the total number of aligned pairs in our training set. We conducted two rounds of back-translation to optimize a French -> English model and improve the quality of generated synthetic data with which we trained a final English -> French model.

IFT-6759 Project 2 Report (Winter 2020), Université de Montréal, Canada.

Our best model was a 4-layer Transformer trained on a mixture of real and synthetic parallel data obtained after two rounds of back-translation optimization. This model implemented the beam-search algorithm at evaluation. Pre-trained embeddings did not improve performance on the validation set in our experiments, and so our best model included embeddings that were trained strictly during the NMT task.

1.2 Related work

In neural machine translation (NMT), sequence-to-sequence models make it possible to generate text (i.e., word sequences) in a target language from another sequence in a different source language (e.g., Sutskever et al. [2014]). Gated recurrent neural networks, which carry sequential information forward over long sequences of tokens, have achieved some early success. However, NMT has greatly benefited from the introduction of attention mechanisms [Bahdanau et al., 2014], which give models the ability to consider all tokens at once at any step in the generation of an output sequence. Recently, Transformer models that rely strictly on attention without recurrence [Vaswani et al., 2017] have gained a prominent position in the field of machine translation [Bojar et al., 2018, 2019a,b].

While the Transformer architecture is well adapted to language translation, Transformers and other NMT models typically rely on heaps of parallel training examples to achieve SOTA performance. For example, Wu et al. [2016] from Google trained their NMT on a set of 36M sentence pairs (WMT14 English-French), while Hassan et al. [2018] used 18M English-Chinese sentence pairs. Text in many languages exists in large quantities online, but typically, the number of parallel examples is much more limited. According to Kocmi [2020], scenarios with fewer than a million training pairs can be considered "low-resource". Gu et al. [2018] qualified a corpus of 13k parallel sentences as "extremely low-resource", that is, of insufficient size to achieve reasonable translation quality with a NMT. By these criteria, the current project (11k parallel sentences) is considered an extremely low-resources scenario.

Recent advances in NMT have introduced creative solutions to compensate for small amounts of parallel data. Transfer learning has been shown to improve a model's performance, for example by pre-training word embeddings on a very large monolingual corpus using a language modeling task (e.g., Word2vec by Mikolov et al. [2013]). Pre-trained embeddings are transferred to the encoder and/or decoder part of an NMT model, which is then trained on a smaller parallel corpus (e.g., Kocmi and Bojar [2017]). Qi et al. [2018] have shown that pre-trained embeddings can improve NMT in low-resource scenarios, possibly because they capture information about word relations extracted from a larger data set.

In cases where a particular combination of languages is rare, some groups have relied on cross-lingual training. A model can be trained to translate large amounts of data from languages A->B and B->C, for example, in order to build multi-lingual representations that can improve A->C translation (e.g., Zhou et al. [2018]). Taken to the extreme, this approach can lead to zero-shot learning on language combinations for which a model has not been trained [Johnson et al., 2017].

Another approach that has recently gained in popularity is to augment the size of the parallel training corpus with synthetic data generated using back-translation [Sennrich et al., 2015a, Guzmán et al., 2019]. The idea is simple but effective: train an NMT to translate sentences from the target language "back" into the source language, and use a mixture of real and synthetic source-target pairs to train the NMT. This process can be repeated iteratively, training A->B and B->A models with synthetic data of increasing quality until convergence. Guzmán et al. [2019] reported improvements of 2.5 to 7.9 BLEU points on low-resource pairs of very distinct languages (e.g., English-Nepali) after two iterations of back-translation.

A fourth approach is to combine two NMT models into an Autoencoder framework that can exploit large unaligned corpora in both the source and the target language. For example, Cheng et al. [2016] trained two NMTs that learned to translate from source $\mapsto$ target and target $\mapsto$ source with an aligned corpus. When combined, the two NMTs also formed Autoencoders that learned to recover sequences from source $\mapsto$ source and target $\mapsto$ target using larger unaligned corpora. The authors showed that the source $\mapsto$ target half of the Autoencoder performed source $\mapsto$ target translation more accurately than a similar NMT trained strictly on aligned data or using back-translation. Artetxe et al. [2017] also introduced an Autoencoder trained entirely on unaligned source and target data in a fully unsupervised manner, although with somewhat poorer results.

For the current project, we explored two of the techniques mentioned above to improve the performance of low-resource NMT models : pre-trained embeddings, and data augmentation using back-translated synthetic data. While cross-lingual NMT has shown promising results in low-resource scenarios, this approach requires text data from at least three different languages, a type of resource that was not available for this project. We also explored whether a semi-supervised Autoencoder could help us take full advantage of the provided unaligned corpora, but found the implementation challenging within the project's short time frame.

In terms of optimal tokenization techniques, NMT has observed a recent shift from word-level to subword-level tokens [Kocmi, 2020]. While traditional NMT systems (e.g., Bahdanau et al. [2014]) process and generate sequences word-by-word, such models operate with a very large fixed vocabulary (e.g., 80k target words for Sutskever et al. [2014]) that cannot translate out-of-vocabulary (oov) tokens unseen at training. Sennrich et al. [2015b] showed that breaking down rare and unknown words into sequences of sub-word units that are more easily translatable using a byte pair encoding (BPE) algorithm achieves greater accuracy with smaller vocabularies. Other popular algorithms like wordpieces (which segments all words into sub-units; Wu et al. [2016]) and SentencePiece (which segments raw sentences; Kudo and Richardson [2018]) also improve translation by segmenting words into sub-components. For the current project, our source and target text encoders learned a vocabulary from a combination of the (language-specific) unaligned and aligned corpus, and byte-encoded out-of-vocabulary words, as shown in Sennrich et al. [2015b] and implemented in the following tutorial [Tensorflow, a].

2 Methods

2.1 Data analysis

The only parallel data provided for training purposes was an aligned corpus of 11k sentence pairs (where the source is unformatted English with no punctuation or capitalization and the target is punctuated and capitalized French). An additional corpus of unaligned formatted sentences (capitalized and punctuated) was also provided in each language (474k sentences per corpus). A held-out corpus of aligned data of unspecified size similar in format and type to the aligned training corpus was used by the course instructors to evaluate the final model's performance. The source of the aligned and unaligned corpora was not revealed. Sequences were saved as text files (utf-8 encoding).

During training, we split the aligned corpus between a train and a validation set (10k and 1k parallel sentences, respectively). Consecutive sentences seemed unrelated and unordered (by either length or topic), and so we simply held out the last 1k examples without reshuffling the aligned corpus. Table 1 includes statistics about sequence lengths per corpus. Length was measured in number of words and number of tokens after sequences were processed with a trained tokenizer (see section 2.2). Metrics indicate that the validation and training sets had similar distributions.

Table 1: Length of sequences per corpus

Type	Language	Size	Words per Sequence			Tokens per Sequence		
			Min	Max	Mean (SD)	Min	Max	Mean (SD)
Unaligned	English*	474k	1	180	18.61 (9.76)	1	360	21.41 (11.35)
	English	474k	1	174	18.19 (9.57)	2	393	28.68 (15.21)
	French	474k	1	218	19.17 (10.13)	2	331	28.35 (14.82)
Aligned	English* (train set)	10k	1	95	18.62 (9.84)	1	119	21.48 (11.45)
	English* (valid set)	1k	3	91	19.14 (10.05)	3	105	21.99 (11.54)
	French (train set)	10k	2	111	22.77 (11.72)	4	165	29.49 (15.06)
	French (valid set)	1k	4	112	23.48 (11.98)	5	146	30.34 (15.21)

* Unformatted text: punctuation and capital letters were removed.
Note: Max = maximal length out of all sequences; Min = minimum length out of all sequences; SD = standard deviation; valid = validation.

Table 1 also reveals that none of the corpora contained any empty sequence (length = 0). Figures 1 and 2 illustrate the distribution of sequence lengths (in words and tokens, respectively) for the unaligned French and English corpora. The overlap between distributions indicates that punctuation and capital letters systematically increased the length of tokenized sequences (while obviously having little impact on word count).

Figure 3 shows the distribution of sequence length in words (left) and tokens (right) per language for the aligned data corpus. Sequence lengths were highly correlated between matching pairs of English and French examples [Words: R = .945 (train set) and .946 (valid set); Tokens: R = .942 (train set) and .945 (valid set)]. These length distributions resemble those of the unaligned corpora, indicating reasonable similarity between the aligned and the unaligned text samples.

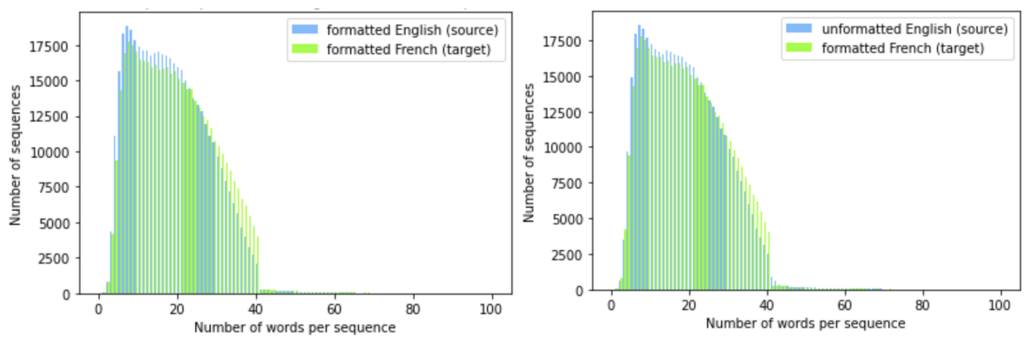


Figure 1: Distribution of the number of words per sequence in the French and English unaligned corpora. LEFT: the English corpus was unformatted (punctuation and capitalization were removed with the provided scripts) to match the aligned English corpus; RIGHT: both corpora were formatted.

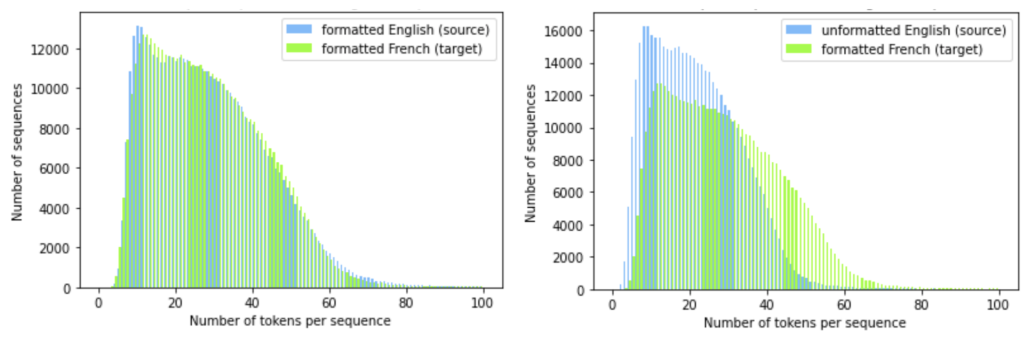


Figure 2: Distribution of the number of tokens per sequence in the unaligned French and English corpora. LEFT: the English corpus was unformatted (punctuation and capitalization were removed with the provided scripts) before being tokenized; RIGHT: both corpora were formatted when tokenized.

2.2 Preprocessing and data loading

Punctuation and capital letters were removed from the unaligned English corpus with a script provided by the course instructors to match the format of the aligned English sentences. All French sentences (aligned and unaligned) remained formatted.

Input data (source and target, or source alone) were converted into `tf.data.Dataset` type sets that were processed (tokenized) with language-specific tokenizers. For the GRU model with Attention (described in section 2.3), we used the entire unaligned data set to create two Word2Vec language models [Mikolov et al., 2013] and tokenized each word using those models [Tensorflow, b], as shown in the following tutorial [Tensorflow, c]. Out-of-vocabulary words were replaced by an "unknown" token (<unk>) and given the average word vector.

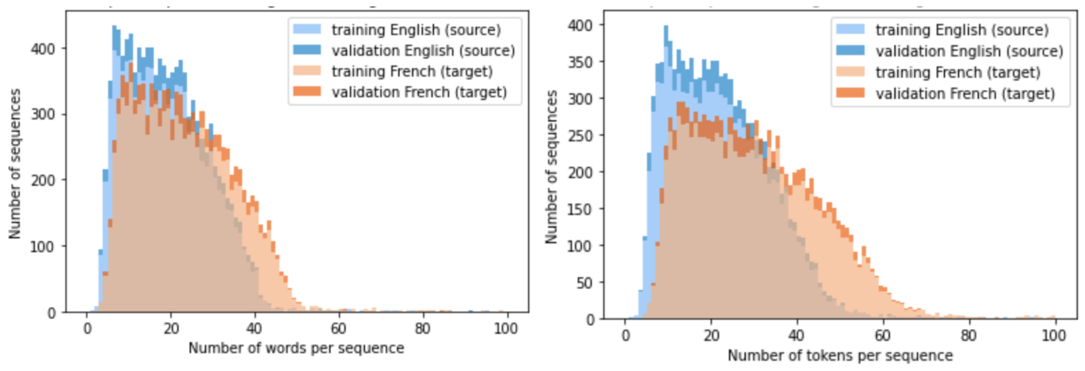


Figure 3: Distribution of the number of words (left) and tokens (right) per sequence in the aligned French (formatted) and English (unformatted) corpora. Training and validation sets are stacked per language.

For the Transformer model (described in section 2.4), language-specific tokenizers were implemented using `SubwordTextEncoder` from the `tensorflow datasets` library [Tensorflow, d] as shown in the following tutorial [Tensorflow, a], based on the byte-encoding approach introduced by Sennrich et al. [2015b]. With this approach, a vocabulary is learned on a corpus, wordpieces are stored in a vocabulary file, and out-of-vocabulary wordpieces are split into subwords with a byte-encoding algorithm. Note that this approach makes encoding fully invertible. In our case, a source and a target tokenizer were trained separately on the monolingual corpus and the portion of the aligned corpus that matched their respective language, using a set vocabulary size of 8192 tokens (the actual vocabulary size deviated slightly from this target value). As vocabulary size is likely to be data and task-related, there is no prescribed size. Our vocabulary size was chosen arbitrarily to match the one used in the Tensorflow Transformer tutorial [Tensorflow, a], and was excluded from our hyper-parameter search due to time constraints. It should be optimized in future works.

A token that indicated a sequence's start and end were added before and after each tokenized sequence. Sequences were padded to the right with zeros up to the length of a minibatch's longest sequence to have consistent length within each batch, as shown in Figure 4. Batch sizes were adjusted to be as large as possible without incurring out-of-memory errors. Batches of 64 were used unless otherwise indicated. Training data sets were cached to memory to speed-up training time.

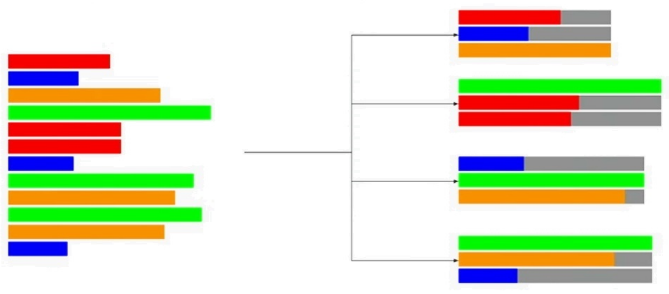
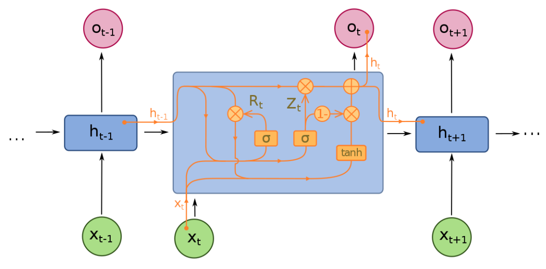


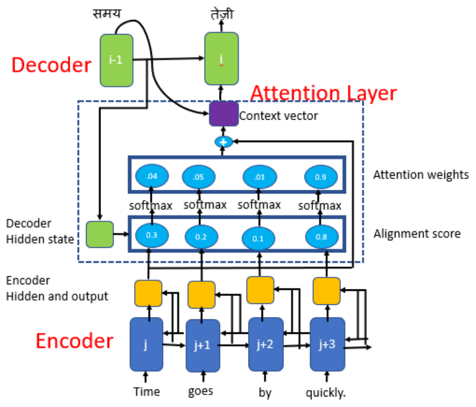
Figure 4: Dynamic padding of sequence minibatches. Source: Turol [April 2017]

2.3 GRU model with Attention

We trained a Gated-Recurrent-Units (GRU) model with Attention [Bahdanau et al., 2014] based on the following implementation [Tensorflow, c]. We used Word2Vec embeddings [Mikolov et al., 2013] of size 100 pre-trained on the unaligned monolingual data. The GRU was trained on the aligned training corpus (10k examples) without additional synthetic data. The unit size was set to 512. We observed poor performance and long training time on longer sequences, so we only used sequences under 40 words for training. Early comparisons with the Transformer architecture (described in



(a) GRU cell



(b) Attention mechanism

Figure 5: GRU model with Attention based on Bahdanau et al. [2014]. Sources : (a) Adapted from Tidwell [August 2019], and (b) Khandelwal [January 2020]

section 2.4) revealed superior results for the later model (section 3.1), and so we focused most of our efforts on training and optimizing Transformers.

2.4 Transformer model

We implemented the Transformer architecture proposed by [Vaswani et al., 2017] based on the implementation described in the following tutorial [Tensorflow, a]. Briefly, the model included an encoder and a decoder with matching numbers of stacked layers (Figure 6). The encoder processed tokenized input (source) sequences through an embedding layer, and applied positional encoding to capture the order among tokens by adding $sin(pos/10000^{2i/d_{model}})$ and $cos(pos/10000^{2i/d_{model}})$ to tokens with even and odd positions, respectively.

Encoder layers included two sub-components whose output was the size of hyper-parameter d_{model}: a multi-head self-attention sub-layer and a 2-layer position-wise fully connected feed-forward (ff) network. ReLU non-linearity was applied on the first ff layer whose size was set by hyper-parameter d_{ff}. A residual connection went around each sub-component, followed by dropout and layer normalization.

The decoder processed tokenized target or output sequences in a manner similar to the encoder (embedding and positional encoding). Embeddings were shifted right by one position compared to the encoder. Decoder layers included three sub-components : a multi-head self-attention sub-layer, a sub-layer that performed multi-head attention over the encoder output, and a 2-layer position-wise ff network. To prevent positions from attending to subsequent positions and "peak into the future", a look-ahead mask was applied to the self-attention sub-component. A residual connection went around each sub-component, followed by dropout and layer normalization.

The model's output layer was a dense layer the size of the target vocabulary that returned logits for each token. Prediction accuracy was computed by comparing these logits against target output sequences to determine whether the correct tokens were predicted. Softmax was applied to logits within the cross-entropy loss function and within the beam search generation algorithm.

The multi-head attention sub-layers applied Scaled Dot-Product Attention in parallel for each head to attend information from different "representation subspaces" at different positions simultaneously [Vaswani et al., 2017]. Head dimension was equal to d_{model}/num_heads. Beside logits, the model also returned a matrix of attention weights for each head in each attention sub-layer (self and encoder-decoder) in each decoder layer useful for visualization.

Our initial hyper-parameters were based on those of Guzmán et al. [2019]. We started with a big Transformer with 5 layers, 2 attention heads, $d_{model} = 512$ and $d_{ff} = 2048$. We observed a lot of overfitting with this architecture, so we gradually reduced the number of layers and conducted a search on other hyper-parameters, including dropout rate.

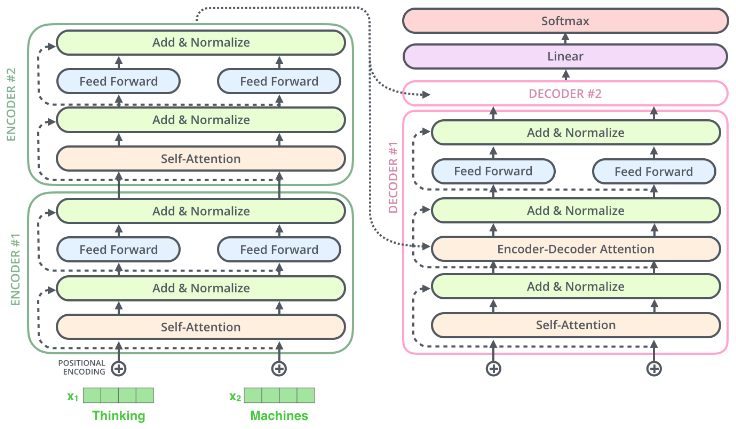


Figure 6: 2-layer Transformer model architecture. Source: Alammar [June 2018]

We tried to optimize the performance of the Transformer model in three different ways: by pre-training word embeddings on language modelling tasks, by augmenting the aligned training corpus with synthetic data using back-translation, and by using a beam search algorithm during translation generation.

Pre-trained embeddings

To initialize the weights of the encoder and decoder embedding layers, we pre-trained two Word2Vec language models [Mikolov et al., 2013] with the unaligned data (one French and one English). We tested the C-BOW and the Skip-grams version of the gensim Word2Vec implementation [gensim, November 2019] (window size = 5). For embeddings to be compatible with the vocabulary of the tokenizers, we first tokenized the unaligned data, and then replaced each token by its corresponding sub-word. For example, the sentence "a traditional midwife is a professional" becomes "a_ traditional_ mid wife_ is_ a_ profession al". This approach generates a model with a vocabulary made of the tokenizer sub-words. We expected that sub-words with similar meanings would have similar embeddings.

We inserted pre-trained embeddings into the encoder and decoder of an English $\mapsto$ French Transformer whose hyper-parameters had achieved reasonable performance during step 2 of the back-translation procedure (Table 2): 4 hidden layers, 4 attention heads, d_{model} = 512, dropout = 0.1 and batch size = 32. This model was trained on the aligned corpus augmented with 20k of "first iteration" synthetic data. We tried fixing embedding weights and fine-tuning them on the parallel data. Pre-trained embeddings were compared against embeddings initialized with a random uniform distribution. Results can be seen in Table 3.

Synthetic data augmentation

Table 2: Iterative Steps of the Back-Translation Procedure

Step	Source	Target	Synthetic Input	Synthetic output	H-param search results
1	French $\mapsto$	English	None	Syn_Fr_1	Table 4
2	English $\mapsto$	French	Syn_Fr_1	Syn_En_1	Table 5
3	French $\mapsto$	English	Syn_En_1	Syn_Fr_2	Table 6
4	English $\mapsto$	French	Syn_Fr_2	None	Table 7

Table numbers indicate where hyper-parameter search results are summarized for each step in section 3.3. In all cases, English was unformatted while French was formatted.

We trained a Transformer with synthetic data using a four-step back-translation procedure inspired by Guzmán et al. [2019]. Steps are summarized in Table 2. At each step, the best model was identified through an hyper-parameter search whose results are summarized in section 3.3.

Step 1. We trained a French to English transformer using the parallel training corpus. Initial model parameters were based on Guzmán et al. [2019], and were adjusted according to performance on the validation set. With the best identified model, we generated 120k unformatted English sentences from the unaligned French corpus. We used these synthetic pairs to augment the size of the training parallel corpus at step 2.

Step 2. We trained an English to French model on a mixed parallel corpus made of real and synthetic data (generated at step 1). With the best model, we generated 474k formatted French sentences, which we combined with the real parallel corpus to augment it at step 3.

Step 3. We trained a second French to English model using a mixed corpus of real and synthetic data (generated at step 2). This round of hyper-parameter search was more limited as it was informed by successful combinations observed at previous steps. With the best identified model, we generated 474k unformatted English sentences. Based on Guzmán et al. [2019], we expected these translations to be of higher quality than those generated at step 1, so that our final model could benefit from being trained on more closely matching pairs.

Step 4. We trained a final English to French model on a mixed corpus of real and synthetic parallel data (generated at Step 3). The best identified model was the one submitted as best model for the completion of this project. We used this model to optimize the beam search algorithm's beam size to generate translations.

Beam Search data generation

We implemented a beam search algorithm to optimize the accuracy of the sequences generated by our best Transformer model trained on mixed (real and synthetic) aligned data. We followed a procedure described by Wu et al. [2016], and developed our own algorithm based on the T2T [January 23, 2020] implementation: at each step (token) during sentence generation, a number of candidate sentences equal to the beam size is maintained and expanded. The final sequence with the highest likelihood is selected among the candidates. We normalized the likelihood of candidate sequences proportionally to their length (α around 0.6) based on recommendations from Wu et al. [2016]. Beam search implementation was conducted in parallel with the iterative training procedure described above. Once the best Transformer was identified and trained, we evaluated the influence of beam size on the quality of the sentences generated based on accuracy and BLEU score (see section 3.4).

2.5 Training and Evaluation

Models were built and trained in Tensorflow 2.1.0. using Python 3.7. With the exception of a few smaller models trained locally on a lower number of examples, most models were trained on Calcul Québec's Helios3 cluster server using NVIDIA K80 GPUs [ComputeCanada].

Models were trained with the cross-entropy loss function, and performance was evaluated on the 1k validation set we held out from the aligned training corpus. We used the validation bilingual evaluation understudy (BLEU) score [Papineni et al., 2002] calculated with the SacreBLEU implementation from Post [2018] to evaluate model performance. We also used validation accuracy as a convenient proxy for BLEU, as we found that both metrics were correlated when vocabulary size was fixed. For all models, the reported BLEU scores are the ones obtained for the training epoch that obtained the highest validation accuracy, as computing BLEU after each epoch was impractical.

We used the Adam optimizer [Kingma and Ba, 2014]. For Transformer models, like Vaswani et al. [2017], we used Adam with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, and a custom learning rate scheduler that followed the following formula (`warmup_steps=` 4000):

$$lrate = d_{model}^{-0.5} \cdot min(step\_num^{-0.5},\ step\_num * warmup\_steps^{-1.5})$$

Models were trained for up to 30 epochs, which was sufficient to overfit the training data in most cases, except for models training on the largest amount of synthetic data (see 2.5). For each model, scores and weights were selected from the epoch where the highest validation accuracy was observed.

3 Results and Discussion

3.1 Model architecture selection

An early comparison between the best GRU with Attention (hyper-parameters detailed in section 2.3) and a 2 layer, 2 head Transformer model ($d_{model} = 512$, $d_{ff} = 2048$) trained on real aligned data (no synthetic data) with 10% dropout revealed a superior BLEU score for the Transformer (BLEU = 8.88) compared to the GRU (BLEU = 5.01). Based on these results, we selected the Transformer architecture for our best model.

3.2 Trained Embeddings

Table 3 shows how using pre-trained Word2Vec embeddings to initialize the embedding layer in the Transformer encoder and decoder affects performance. All other hyper-parameters were fixed across models (see section 2.4). When embedding training was allowed, embedding weights were fine-tuned while training the Transformer on the aligned corpus, otherwise embedding weights were fixed. Results were much poorer with pre-trained embeddings than with the random initialization. For this reason, we used random embedding initializations to optimize our best model.

Table 3: Experiments using pre-trained Word2Vec embedding

Embedding Initialization	Allow embedding training	Accuracy	Bleu Score
Random Uniform	True	**0.0922**	**8.90**
Word2Vec-CBOW	True	0.0718	2.84
Word2Vec-CBOW	False	0.0699	2.63
Word2Vec-Skip-Gram	True	0.0748	3.94
Word2Vec-Skip-Gram	False	0.738	3.57

3.3 Data Augmentation with Back-Translation

Table 4: Hyperparameter search for French -> English model (first iteration)

layers	d_{model}	d_{ff}	heads	dropout	Accuracy	BLEU Score
1	512	2048	2	0.1	0.0695	
2	512	1024	2	0.2	0.0711	
	512	2048	2	0	0.0664	
	512	2048	2	0.1	0.0708	
	512	2048	2	0.2	**0.0718**	**11.47**
	512	2048	2	0.3	0.0704	
	512	2048	4	0.2	0.0704	
3	256	1024	2	0.1	0.0686	
	256	1024	2	0.25	0.0659	
	256	1024	2	0.5	0.0508	
	512	1024	2	0.1	0.0687	
	512	1024	4	0.1	0.0671	
	512	2048	2	0.2	0.0694	
4	256	1024	2	0.1	0.0658	
	512	2048	2	0.1	0.0648	
5	512	2048	2	0.3	0.0490	

Note: no synthetic data were used at training. Best model results are shown in bold.

Results from the hyper-parameter search conducted at Step 1 to identify the best French $\mapsto$ English model are summarized in Table 4. We initially started with a big model (5 layers, 2 heads) but observed a lot of over-fitting so we gradually reduced the number of layers. We also conducted a search on the dropout rate, the number of heads, d_{ff} and d_{model}. At Step 1, we used validation prediction accuracy as a proxy for the BLEU score as its computation was not yet integrated into our training pipeline (we observed strong correlations between the two scores). The best model (2 layer, 2 attention heads, ($d_{ff} = 2048$, $d_{model} = 512$) achieved a validation BLEU score of 11.47 (accuracy = 7.18%). With this model, we generated 120k English synthetic examples which we used to train models at Step 2.

Table 5 summarizes the hyper-parameter search conducted a Step 2. From this point onward, we used $d_{model} = 512$ and $d_{ff} = 2048$, which had achieved the best scores at Step 1. Without synthetic data, smaller models (2 layers, 2 heads) were superior to larger ones, while the 4 layer, 4 head model achieved the highest scores when the training set was sufficiently augmented. Better results were systematically achieved with larger amounts of synthetic data. The best model was trained with 120k synthetic examples (the total amount generated at Step 1) and obtained a BLEU score of 11.17. With this model, we generated 474k synthetic French examples (the maximal amount possible from the unaligned English corpus, since more data appeared helpful) to train models at Step 3.

Table 5: Hyperparameter search for English -> French model (first iteration)

layers	n_synth	d_{model}	d_{ff}	heads	dropout	Accuracy	BLEU Score
2	0	512	2048	2	0.1	0.0854	8.88
	10k	512	2048	2	0.1	0.0910	10.01
	20k	512	2048	2	0.1	0.0934	10.50
	40k	512	2048	2	0.1	0.0948	10.05
	120k	512	2048	2	0.1	0.0933	10.98
4	0	512	2048	4	0.1	0.0757	7.22
	10k	512	2048	4	0.1	0.0879	9.09
	20k	512	2048	4	0.1	0.0939	10.25
	40k	512	2048	4	0.1	0.0973	10.66
	120k	512	2048	4	0.1	**0.0979**	**11.17**
5	0	512	2048	2	0.1	0.0672	4.17
	10k	512	2048	2	0.1	0.0730	4.28
	20k	512	2048	2	0.1	0.0779	4.48
	40k	512	2048	2	0.1	0.0785	4.65
	120k	512	2048	2	0.1	0.0793	5.15
6	0k	512	4096	8	0.1	0.0648	3.86

Note: n_synth = number of synthetic examples included in parallel training set.
Best model results are shown in bold.

Table 6 shows results from the hyper-parameter search done at Step 3. Parameter choices were informed by successful results from Step 2. Best scores (BLEU = 17.51) were achieved with a 4 layer, 8 head model trained on the entire synthetic data corpus (474k examples). We also tried a larger model (6 layers, 8 head) to test whether a larger capacity would do well with a larger training set, but stopped its training due to poor results. With the best model, we generated 474k English examples from the unaligned French corpus to train our final model at Step 4.

Results from the limited hyper-parameter search conducted at Step 4 are shown in Table 7. For comparison purposes, we trained a 4 layer, 4 head model on 120k synthetic examples and obtained a BLEU score of 14.40. This score was substantially higher than the BLEU score of 11.17 obtained with similar parameters at Step 2, demonstrating the benefits of training models on higher-quality synthetic data obtained with multiple back-translation iterations. Our best final model (4 layers, 8 heads) was trained on 474k synthetic examples with a dropout rate of 0.1 We experimented with higher dropout rates (> 0.1) but found that they hurt performance, suggesting that sufficient regularization is provided by the intrinsic noisiness of the synthetic data.

Table 6: Hyperparameter search for French -> English model (2nd iteration)

layers	n_synth	d_{model}	d_{ff}	heads	dropout	Accuracy	BLEU Score
4	20k	512	2048	4	0.1	0.0847	12.44
	120k	512	2048	4	0.1	0.1013	15.90
	474k	512	2048	4	0.1	0.1099	16.90
	120k	512	2048	8	0.1	0.1005	15.54
	474k	512	2048	8	0.1	**0.1105**	**17.51**
6	474k	512	2048	8	0.1	0.0862	(Stopped)

*Training was restarted from a checkpoint after 5 epochs, for a total of 5 + 12 = 17 training epochs.
Note: n_synth = number of synthetic examples included in the parallel training set.
Best model results are shown in bold.

Table 7: Hyperparameter search for English -> French model (2nd iteration, final model)

layers	n_synth	d_{model}	d_{ff}	heads	dropout	Accuracy	BLEU Score
4	120k	512	2048	4	0.1	0.1027	14.40
4	474k	512	2048	8	0.0	0.1032	14.72
	474k	512	2048	8	0.1	**0.1040**	**15.49**
	474k	512	2048	8	0.2	0.10	14.65
	474k	512	2048	8	0.3	0.0727	4.85
	474k	512	2048	8	0.4	0.0689	4.28

Note: n_synth = number of synthetic examples included in the parallel training set.
Best model results are shown in bold.

3.4 Beam Search algorithm

Table 8: Beam size optimization for final model

Beam size	batch size	time spent (s)	BLEU Score
1	32	159	15.49
2	32	120	16.11
3	32	174	16.22
4	32	182	16.29
5	32	209	16.54
6	32	249	16.59
7	32	314	16.68
8	16	351	16.70
9	16	389	**16.71**
10	16	386	16.69
11	16	436	16.67
12	16	428	16.70

Note: optimization was conducted with the best model identified in Table 7.
Best results are shown in bold.

Table 8 shows BLEU scores obtained with our best model using different beam sizes, with the α (length normalization) parameter fixed to 0.6 based on recommendations from Wu et al. [2016]. Batch size was reduced when using a larger beam size for memory reason, which should not have impacted the BLEU score. Larger beam size requires more memory usage and longer run time, which prevented us from using beam search to generate synthetic data, unfortunately. The optimal results for our model were achieved with a beam of size 9, after which results stopped improving. The final model submitted for this project used a beam size of 7 to be sure it will translate all test sentences within the 30 minutes time limit.

Table 9: Beam search α parameter optimization for final model

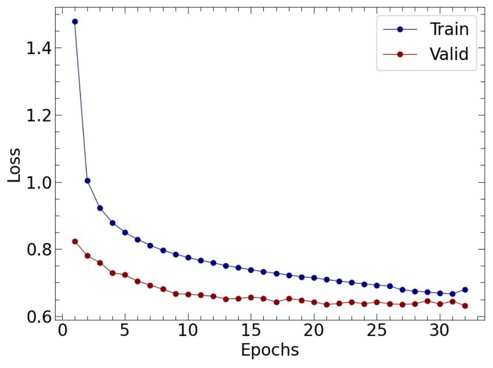
α	time spent (s)	BLEU Score
0.3	394	16.71
0.4	403	16.71
0.5	397	**16.75**
0.6	389	16.71
0.7	393	16.74
0.8	395	16.74

Note: Beam size was fixed to 9 and batch size to 16

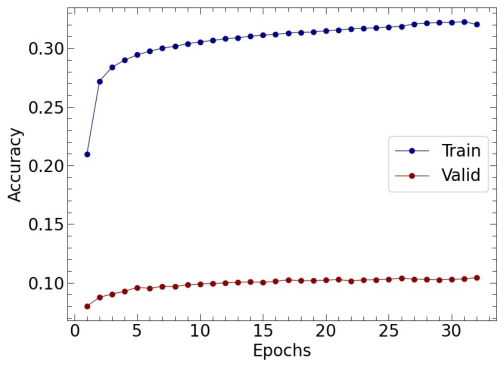
We also experimented with the α parameter used to normalize the length of different sequences during beam search. Results (see Table 9) were inconclusive as α seems to have very little impact on the validation BLEU score. Although marginally better results were obtained with $\alpha = 0.5$, we chose to keep 0.6 as our final value since similar results where obtained with $\alpha = 0.7$ and 0.6, which were within the range proposed by Wu et al. [2016].

3.5 Best model visualization

Figure 7 describes the training curves for our best model. Both loss and accuracy values indicate that



(a) Train/validation loss



(b) Train/validation accuracy

Figure 7: Training curves for best model

despite having ran for 30 epochs, the model continued to learn (validation accuracy and loss kept improving, though slowly, without over-fitting). Learning was slow and long (approximately 1 hour per epoch, 30 hours for 30 epochs). Also, the training loss values are higher than the validation's. The underlying reason could be the regularization (dropout), as it applied during training but not during validation. Translated samples obtained with our best model are included in Appendix A.

Figure 8 shows the attention given to each token in the last layer of our best model for a sample sentence. We observe that the translation is quite accurate, and that attention from at least one of the heads is always very high for the corresponding token in the source language.

4 Conclusion

4.1 Successful model features and future steps

Consistent with the Transformer's current dominance in NLP and growing prominence in NMT [Bojar et al., 2018, 2019a,b], our early experiments revealed superior output quality and training time

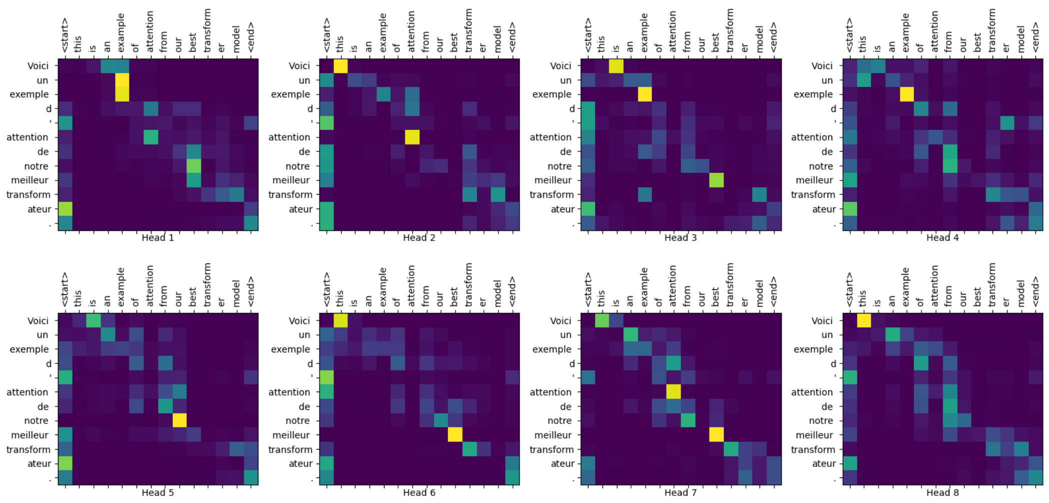


Figure 8: Attention weights in the last layer of our best model. Top is the source sentence and left is the translation.

efficiency for the Transformer model compared to the GRU with Attention. Unlike GRUs, which are sequential models, Transformers allow for more parallelization by eschewing recurrence, which reduces training time [Vaswani et al., 2017].

We also found that augmenting the size of the training corpus with back-translated synthetic data improved translation quality substantially, as shown by Sennrich et al. [2015a]. In fact, greater amounts of synthetic data systematically lead to better BLEU validation scores without reaching a point of convergence, as observed by Hoang et al. [2018]. In the final two steps of our back-translation procedure, the best models were those trained with the full back-translated unaligned corpus. While a greater proportion of synthetic data dilutes the average quality of aligned training sequences, it is interesting to note that greater amounts of data resulted in superior results despite this quality trade-off. As a next step, we could repeat the iterative back-translation process using the maximal possible amount of synthetic data at every steps.

Our best results were obtained after two back-translation iterations, similarly to Guzmán et al. [2019] and Hoang et al. [2018]. Synthetic data quality improved by several BLEU points between iterations. As a future step, it would be interesting to explore whether model performance improves further after a third iteration, since our base score started so low due to our very small aligned corpus. The project's short timeline also constrained the scope of the hyper-parameter search we conducted with the back-translation procedure. Within the parameters we tested, we found that higher capacity models became more appropriate as the size of the training corpus increased. Future work could explore more thoroughly how training set size and quality influences optimal model dimensions.

Beam search improved the quality of our best model's output compared to a completely greedy generation algorithm. Bahdanau et al. [2014] and Vaswani et al. [2017] have used beam search successfully, among others. Like Wu et al. [2016], we used a length normalization (or length penalty) approach, although its impact on model performance was unclear. Our implementation of the beam search was conducted in parallel with back-translation. Edunov et al. [2018] have shown that back-translation conducted with beam search is well adapted to low-resource settings. We observed that our 2nd iteration French $\mapsto$ English model obtained better BLEU scores with a beam size of 9, and tried to generate English synthetic data with beam search (as done by Guzmán et al. [2019] and Sennrich et al. [2015a]). Unfortunately, beam search slowed down translation and forced us to reduce the minibatch size to avoid memory errors, and it took too long to generate all 474k synthetic examples. With additional computing power and training time, we could repeat back-translation with beam search to improve the overall quality of the synthetic training data. We could also use a larger beam size, which improved our validation BLEU score but was impractical to run.

Interestingly, we found that dropout > 0.1 hindered performance when training our final model, possibly because the noise from the synthetic data was sufficient to regularize the model. Edunov et al. [2018] showed that adding noise to the beam search during back-translation improved performance in high- but not low-resource (80k examples) settings, likely because poor quality back-translation is overpowered by the noise.

While Word2Vec embeddings pre-trained on enormous corpora can improve low-resource NMT performance in some cases [Qi et al., 2018, Kocmi and Bojar, 2017], our unaligned data set may have been too small to pre-train embeddings, as our pre-trained embeddings were harmful to our NMT models. According to Kocmi [2020], pre-trained embeddings are not commonly used in NMT, partially due to a shift from word-level to subword-level tokenization (pre-trained embeddings are typically trained on tokenized words, making vocabularies incompatible). In our case, the embedding vocabulary was built to match our byte-encoding tokenization. It is possible, however, that embeddings pre-trained on a monolingual task rather than for translation may have been a poor fit for the NMT task, as words trained in similar contexts are not necessarily synonymous, and many tokens unique to the unaligned corpus would not be fine-tuned at training.

As a future step for this project, an exciting new development in low-resource NMT is the emergence of semi-supervised (e.g., Cheng et al. [2016]) and unsupervised (e.g., Artetxe et al. [2017]) learning to maximize the use of abundant unaligned corpora. We dedicated significant efforts to implement an approach introduce by Cheng et al. [2016], for which they combined two NMT models (Bahdanau et al. [2014]'s RNNSEARCH, in their case) into and Autoencoder. The NMTs were respectively trained to translate from source $\mapsto$ target and target $\mapsto$ source with an aligned corpus (see schema in Appendix B), and then combined to form two Autoencoders that learned to recover sequences from source $\mapsto$ source and target $\mapsto$ target with unaligned corpora. The source $\mapsto$ target portion of the Autoencoder formed an NMT whose performance was enhanced by the additional unsupervised training on the unaligned data.

To train their model simultaneously on two translation (back and forward) and two autoencoding tasks, Cheng et al. [2016] used a complex loss function with four objectives. We first attempted to implement each task sequentially (translation tasks followed by autoencoding) to get the necessary building blocks in place to then implement the simultaneous training. We also used Transformers as our NMTs rather than RNNs. Unfortunately, implementing this framework proved very challenging: we struggled to back-propagate signal between two Transformers, and to implement the Autocoencoder without encuring out-of-memory errors. As a next step for this project, we would like to pursue our implementation of the Autoencoder framework with Transformers, given the superior results reported by Cheng et al. [2016] with this framework over NMTs trained stricly with parallel data or using back-translation.

4.2 Challenges and limitations

The biggest challenge we faced with the current project was obviously to train models with extremely low resources without having access to pre-trained models or external data. For example, large additional corpora in other languages would have made it possible to attempt cross-lingual training.

The project's short time-frame also limited our capacity to explore hyper-parameters, to train models and to implement more complex frameworks. Training a model on the largest possible amount of synthetic data required around 24h for 17 epochs with a single GPU, which limited our capacity to test different models, especially when using iterative back-translation. Our capacity to run long jobs was also restricted, and more epochs could have improved the performance of models that did not overfit the training set, including our best final model. Additional computing capacity (e.g., having access to more than one GPU at a time) could have also facilitated this process and supported our implementation of the Autoencoder.

4.3 Closing Remarks

We performed low-resource NMT using a Transformer model, and succesfully optimized performance with beam search and iterative back-translation. We found pre-trained embeddings unhelpful in our context. Future steps should involve more extensive hyper-parameter search, experimenting with additional rounds of back-translation, and implementing semi-supervised learning to maximize our use of the unaligned data made available for the project.

References

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567, 2018. URL http://arxiv.org/abs/1803.05567.

Tom Kocmi. Exploring benefits of transfer learning in neural machine translation, 2020. URL https://arxiv.org/abs/2001.01622.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL https://arxiv.org/abs/1409.0473.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

Matt Post. A call for clarity in reporting BLEU scores. *CoRR*, abs/1804.08771, 2018. URL http://arxiv.org/abs/1804.08771.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL http://arxiv.org/abs/1301.3781.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. Two new evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english. *CoRR*, abs/1902.01382, 2019. URL http://arxiv.org/abs/1902.01382.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709, 2015a. URL http://arxiv.org/abs/1511.06709.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors. *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, Belgium, Brussels, October 2018. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Marco Turchi, and Karin Verspoor, editors. *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Association for Computational Linguistics, Florence, Italy, August 2019a. URL http://www.aclweb.org/anthology/W19-53.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Marco Turchi, and Karin Verspoor, editors. *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Association for Computational Linguistics, Florence, Italy, August 2019b. URL http://www.aclweb.org/anthology/W19-54.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1032. URL `https://www.aclweb.org/anthology/N18-1032`.

Tom Kocmi and Ondrej Bojar. An exploration of word embedding initialization in deep-learning tasks. *CoRR*, abs/1711.09160, 2017. URL `http://arxiv.org/abs/1711.09160`.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2084. URL `https://www.aclweb.org/anthology/N18-2084`.

Zhong Zhou, Matthias Sperber, and Alexander Waibel. Massively parallel cross-lingual learning in low-resource target language translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 232–243, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6324. URL `https://www.aclweb.org/anthology/W18-6324`.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5(0):339–351, 2017. ISSN 2307-387X. URL `https://transacl.org/ojs/index.php/tacl/article/view/1081`.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1185. URL `https://www.aclweb.org/anthology/P16-1185`.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *CoRR*, abs/1710.11041, 2017. URL `http://arxiv.org/abs/1710.11041`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015b. URL `http://arxiv.org/abs/1508.07909`.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226, 2018. URL `http://arxiv.org/abs/1808.06226`.

Tensorflow. Transformer model for language understanding, a. URL `https://www.tensorflow.org/tutorials/text/transformer`.

Tensorflow. tf.keras.preprocessing.text.tokenizer, b. URL `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer`.

Tensorflow. Neural machine translation with attention, c. URL `https://www.tensorflow.org/tutorials/text/nmt_with_attention`.

Tensorflow. tfds.features.text.subwordtextencoder, d. URL https://www.tensorflow.org/datasets/api_docs/python/tfds/features/text/SubwordTextEncoder.

Sophia Turol. The magic behind google translate: Sequence-to-sequence models and tensorflow, April 2017. URL https://www.altoros.com/blog/the-magic-behind-google-translate-sequence-to-sequence-models-and-tensorflow/.

Darek Tidwell. Character level text generation with lstm rnn in tensorflow: Alice in wonder land, August 2019. URL https://darektidwell.com/character-level-text-generation-with-lstm-rnn-in-tensorflow-alice-in-wonder-land/.

Renu Khandelwal. Attention: Sequence 2 sequence model with attention mechanism, January 2020. URL https://towardsdatascience.com/sequence-2-sequence-model-with-attention-mechanism-9e9ca2a613a.

Jay Alammar. The illustrated transformer, June 2018. URL http://jalammar.github.io/illustrated-transformer/.

gensim. models.word2vec – word2vec embeddings, November 2019. URL https://radimrehurek.com/gensim/models/word2vec.html.

T2T. /tensor2tensor/utils/beam_search.py, January 23, 2020. URL https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/beam_search.py.

ComputeCanada. Helios. URL https://docs.computecanada.ca/wiki/H%C3%A9lios.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2703. URL https://www.aclweb.org/anthology/W18-2703.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1045. URL https://www.aclweb.org/anthology/D18-1045.

A Sample translations from best model on validation set examples

Input:the members concerned have never listened to nonetheless they voted in favour of a document bearing no relation to truth or reality

Output: Les députés n'ont néanmoins jamais écouté, ils ont néanmoins voté en faveur d'un document de vérité ou de réalité.

Input:in fact this crisis is also a crisis for democracy

Output: En fait, cette crise est également une crise pour la démocratie.

Input:a risky thing to do because they were street kids

Output: Un risque, parce qu'ils étaient descendus dans les rues.

Input:what stage are we at

Output: Quels stades sommes-nous ?

Input:the rapporteur is right to note that the transatlantic relationship was in fact the keystone of our security as long as it was under threat from the east - west divide

Output: Le rapporteur a-t-il raison de noter que la relation transatlantique était en effet la preuve de notre sécurité d'approvisionnement, alors qu'elle était sous la menace de l'Occident ?

Input:is there anyone else around

Output: Est-ce qu'il y a quiconque ?

Input:and quarks have electric charges in thirds

Output: Et les protéines animales ont des répercussions dans les pays tiers.

Input:greenwich village is a place which especially attracts the young

Output: Windows est un endroit qui attire notamment l'attention.

Input:my father disapproved of my going to the concert

Output: Mon père a désapprouvé ma conduite.

Input:so we sucked up

Output: Donc nous sommes tels.

Input:i did n't sleep well

Output: Je n'ai pas dormi."

Input:since i had a slight fever i stayed in bed

Output: Depuis que j'avais une mouche, je suis resté en lit.

Input:i regret that the report by the committee on petitions refers us only to the european personnel selection office and not to the european ombudsman who until now has not carried out an ex officio scrutiny of his department

Output: Je regrette que le rapport de la commission des pétitions ne nous présente pas seulement à la sélection des députés européens, et non pas jusqu'à aujourd'hui en tant que médiateur européen, qui n'a pas effectué d'enquête sur ses travaux.

Input:in writing - the maltese islands are the southern frontier of the eu

Output: par écrit. - (EN) Les Maldives."."

Input:tom did n't think that anyone would recognize him

Output: Tom ne pensait pas que quiconque reconnaîtrait lui-même.

Input:will china find a way to reverse its economic slowdown

Output: La Chine trouvera un moyen de corriger son ralentissement économique.

Input:i am extremely unhappy with the world trade talks which threaten the european food policy and i said the same to commissioner mandelson at a meeting we had yesterday

Output: Je suis extrêmement enthousiaste avec le commerce mondial qui menace la politique alimentaire européenne et je l'ai dit même au commissaire Mandelson lors d'une

---

Input:we are using this resolution to ask the new commissioners to give priority to the aspects of their portfolios relating to the roma and to stop pursuing the current policy which is long - winded but devoid of any real action

Output: Nous utilisons cette résolution pour demander aux nouveaux commissaires de donner la priorité aux aspects concrets de leur feuille de route, notamment en ce qui concerne les Roms et d'arrêter de fumer, qui est la politique actuelle, mais à long terme.

Input:mr schulz compared you to the leopard

Output: Monsieur Schulz, vous comparez le pavé.

Input:a key thing is that the major violence genes it 's called the mao - a gene

Output: Une chose clé est que la violence majeure, c'est-à-dire le Massachusetts.

Input:to this end the scope of the world trade organization should be extended through the negotiation of future global arrangements that reduce subsidies and both tariff and non - tariff barriers

Output: Pour ce faire, le champ d'application de l'Organisation mondiale du commerce, devrait être étendu à la négociation d'une solution globale qui réduise les émissions de gaz à effet de serre et non pas aux deux.

Input:are there any objections to this oral amendment

Output: Y a-t-il des objections à cet amendement oral ?

Input:so because the fusiform is intact the chap can still recognize his mother and says oh yeah this looks like my mother but because the wire is cut to the emotional centers he says but how come if it ’s my mother i do n’t experience a warmth or terror as the case may be right

Output: Donc, parce que l’IRA est intacte, les cochons peuvent encore reconnaitre sa mère et dire, "Oh, cela ressemble à ma mère à la femme, mais parce que le théâtre de l’émotion, ou peut-être est-ce que je viens peut-être de le reconnaître,

---

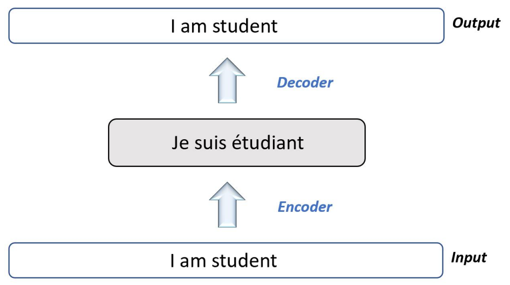
Input:in its annual report for 2010 the european commission highlighted the challenges facing europe as a result of the recent economic and financial crisis in terms of equality between men and women

Output: Dans son rapport annuel pour 2010, la Commission européenne a mis en avant les défis auxquels l’Europe est confrontée en termes économiques récents et en termes d’égalité entre les hommes et les femmes.

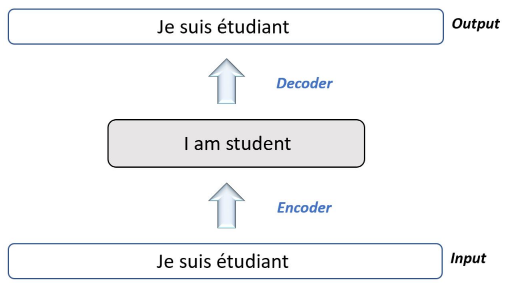
Input:light from the sun wind and the tides are forms of this energy

Output: La lumière du soleil et le vent sont des formes d’énergie de ce type.

B Schema of the semi-supervised Autoencoder framework



(a) Source to source auto-encoder



(b) Target to target auto-encoder

Figure 9: Auto-encoder NMT framework. Image adapted from Cheng et al. [2016]