

An English to French Neural Machine Translation System

Hong Phuc Nguyen
ID 40082916

Wayne Huras
ID 40074423

Abstract

The project's goal was to successfully implement a neural machine translation model using sequence to sequence translation, gated recurrent unit, and Bahdanau's attention method. The project was successful in that the training and testing errors were very close (within 0.5%). This percentage means that the model will generalize well to unseen data.

1. Introduction

Language translation is difficult, not only for machines but for human translators as well. This is due to many factors such as words with multiple meanings and varying grammatical structures (Kay). However, translation is specifically difficult for machines. For example, even leading companies in translation such as Google can struggle with translation:

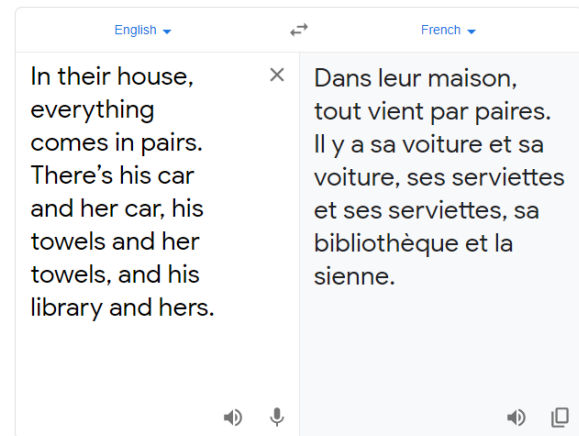


Figure 1. The Shallowness of Google Translate (Hofstadter, 2018)

In the beginning of machine translation history, scientists treated translation as a word-for-word lookup and replacement problem (Garr, 2017). This approach quickly proved ineffective as this method does not consider the complex syntax and semantics of language. This is the problem of decoding the meaning of language. A natural human translation follows the simple process of first decoding the meaning of a sentence, then translating, or re-coding, that meaning into another language. This process is quite simple for humans, however this is not the case for machines and has given birth to the

much larger problem of natural language understanding (Macherey et al., 2001).

There are many approaches to machine translation: rule-based, statistical, example-based, hybrid, and neural machine translation. Statistical machine translation (SMT) was introduced in 1949 by Warren Weaver (Weaver, 1949/1955) and was the most widely used method of machine translation. However it had its shortcomings such as heavily relying on parallel texts between two languages. This reliance led to translations where the meaning of entire sentences changed. For example, if the training set includes repetitive phrases, “train to Berlin” could be mistranslated to “train to Paris” because that phrase could be more common in the training set (*Benefits from Neural Machine Translation: Next-Level Translation for Everyone*, 2019). However in 2016, Neural machine translations (NMT) quickly overtook SMTs on the stage of machine translation. NMTs offer more fluent, humanlike translation, higher quality and accuracy, and can even take context into account (*MT Features: Neural Machine Translation*). As NMTs are paving the future of machine translation, our team decided to use the NMT approach when developing our English to French translation system.

2. Methodology & Experiment Design

Our experiment was heavily influenced by Renu Khandelwal’s “Implementing Neural Machine Translation with Attention mechanism using Tensorflow” (Khandelwal, 2020). The publication assisted our team in implementing our NMT.

Obtaining and Reading the Data

Tatoeba is a free collaborative online database which was used to obtain our training, testing, and validation data. Specifically, we used the tab-delimited bilingual sentence pairs for French and English which includes approximately 180,000 English sentences alongside the equivalent French sentences. Pandas was used to parse the raw data file into columns identified as the “source”, “targets”, and “comments”. The source being the original English sentences, the targets being the French translation, and the comments being inconsequential.

Preprocessing and Tokenizing the Data

Preprocessing data is essential for successful translation. There are many variations and writing styles that can complicate the process such as punctuation, capitalization, and spacing. The Regular Expression python library was used to format the source data to convert the text to lower-case, remove quotation marks, digits, and symbols, while

at the same time adding indicators for the start and end of the sentences.

TensorFlow (Keras) was used to tokenize the data. This essentially creates a list of every word in the given data, and then converts each sentence into a list with numerical representations of the words called tokens. This is a fundamental preprocessing step as it helps prepare the data for building the training, testing, and validation sets. However, this process does have its limitations. Words which are not included in our data will not be tokenized and therefore we will run into errors when trying to translate a new word.

Creating the Training and Testing Sets

Scikit learn's `train_test_split` utility was used to separate our data as follows: 80% is converted into a training set, 10% a test set, and 10% a validation set.

Encoding, Attention Layer, and Decoding

The tokenized source data is used as input to the Encoder which passes them through to an embedding layer. The embedding provides words with a context or meaning through a vector representation. The position of the word within the vector space is derived from the raw data and their relative placement within each sentence (Brownlee, 2020).

Attention layers are used to assist in prediction. Instead of encoding a single context vector, the attention layer filters the vector to help contain possibilities which are

of higher relevance. Our team used Bahdanau's attention (Bahdanau, et al., 2014) method which is able to consider all the words in the input (represented by hidden states). It accomplishes this by utilizing weighted sums of the hidden states (Hore & Chatterjee, 2019).

The Decoder uses the context vector alongside its own output via recurrent neural networks (RNN) to predict the translation. The team used a gated recurrent unit (GRU) as the RNN gating mechanism which is responsible for what is "remembered" by the RNN. GRU was chosen as it is computationally more efficient and simpler to utilize than the alternative, long short-term memory (LSTM) (Chung, et al., 2014).

Optimization and the Loss Function

Optimizing for neural networks can be tricky because we are dealing with multiple optima which can make it difficult to find the global optima (Stewart, 2020). For this reason, gradient descent is often the preferred way to optimize neural networks. Adaptive Moment Estimation (Adam) was chosen as the optimizer which is a method that computes adaptive learning rates and keeps exponentially decaying averages of past gradients (Ruder, 2016).

Sparse categorical cross entropy, a loss function provided through TensorFlow (Keras), was used to evaluate the prediction error as it is both computational and memory efficient (Khandelwal, 2020).

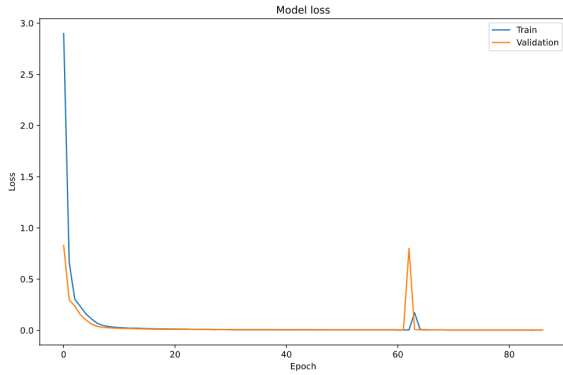


Figure 2. Model Loss per Epoch

Training the Dataset

We use gradient tape to train the dataset as TensorFlow records operations for all trainable variables. This helps maintain a high level of control over the areas where we need gradient information. In addition, teacher forcing was used as it is both a fast and efficient method for training recurrent neural networks (Brownlee, 2019). The previously mentioned GRU also helps solve the vanishing gradient problem commonly encountered while training neural networks (Khandelwal, 2018).

GPU training was beneficial and significantly reduced the overall training time by 300% compared to a CPU. Training progress went well until epoch 67 when accuracy decreased significantly. The training accuracy recovered however hit another snare at epoch 88. We experimented with, and eventually implemented early stopping which restored progress to a previous checkpoint that was within an acceptable accuracy threshold. This is a known occurrence with, however not limited to, Adam because the model starts to move

“fast” in a certain direction and can accidentally move “too far” due to the momentum.

Training Time	CPU # Epochs	GPU # Epochs
13 hours	13	35

Table 1. Training time comparison CPU vs. GPU.

Translation

After the data (a word) has gone through these steps, the numerical representation generated is converted into a translated word. This word is then used recurrently to predict the next words to form the target sentence. Example translation:

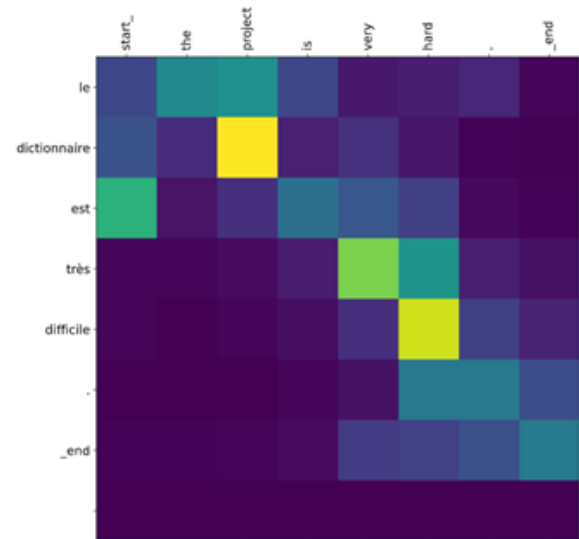


Figure 3. NMT translation of “the project is very hard” from English to French.

4. Conclusion

The team was able to successfully implement, train, test, and validate an open source neural machine translation model. The test and training error are very close (within 0.5%) meaning the model generalizes well to unseen data. This was proved by our validation set using two metrics offered by Tensorflow (Keras): categorical accuracy and exact accuracy. In the future we could be more innovative and experiment with different architectures, train on more robust data, and expand to different languages.

Test Loss	0.00202540494501590
Test Masked Categorical Acc.	0.9988803863525391
Test Exact Matched Acc.	0.9716096520423889

Table 2. Model Accuracy

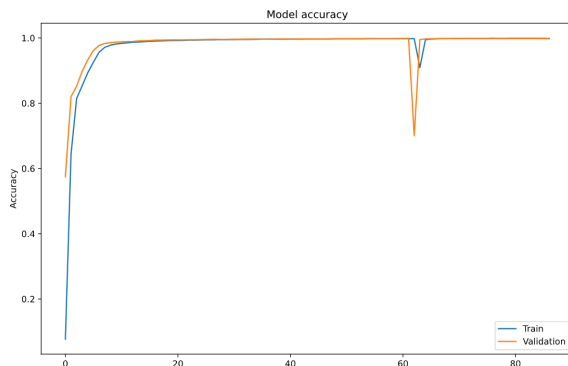


Figure 4. Model Accuracy per Epoch

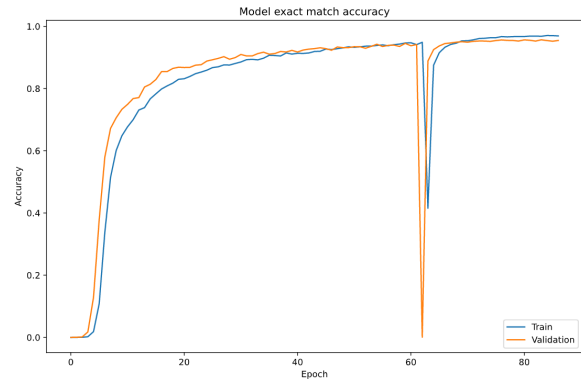


Figure 5. Model Exact Match Accuracy per Epoch

Appendix: Examples - Success & Failures

Example Failure Results	
English	He brought his work home with him
Model	Il sest fait le travail à la maison avec lui
Google Translate	Il a ramené son travail à la maison avec lui
English	The cat played in the snow all day long
Model	Le chat dans la rivière
Google Translate	Le chat a joué dans la neige toute la journée

Example Successful Results	
English	They are going to the store in his car
Model	Ils vont au magasin dans la voiture
Google Translate	Ils vont au magasin dans sa voiture
English	He went to the attic.
Model	Il est allé au grenier
Google Translate	Il est allé au grenier

Appendix: Additional Features

Subword Segmentation was used to develop an understanding of rare and novel words. This form of tokenization does not penalize the error rate when the model encounters unrecognized words such as names. The team found the quality of this feature was relatively low. Example of subword segmentation:

English Input: the United States is never freezing during November, but is sometimes rainy in winter .

French Output: les états-unis ne sont jamais glaciales en novembre , mais il est parfois pluvieux en hiver .

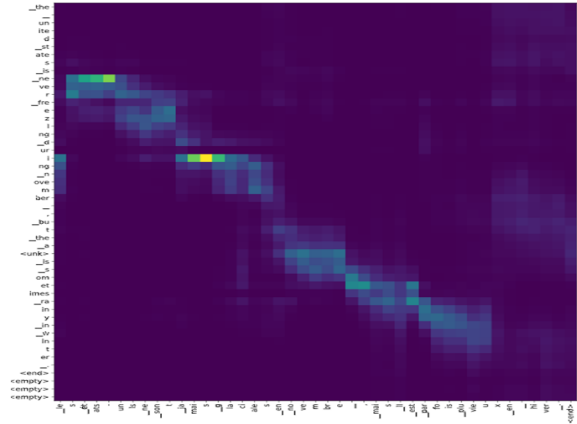


Figure 6. Subword Segmentation - rare and novel words.

Appendix: Alternative Methods

The team tested using a **bidirectional LSTM** for encoding and decoding. The translation accuracy improved, however there was a significant time increase of 26%.

	LSTM	GRU
Time Per Epoch	1260 sec	1000 sec

Table 3. LSTM vs. GRU encoding and decoding time per epoch.

Appendix: Other Notes and Figures

Cell	Params	Time
GRU	342x128	1050s/epoch
LSTM	353x128	1260/epoch

Table 4. Hyperparameter Optimization (Kamsetty et al., 2020)

Beam	newstest2013	Params
B1	20.66 ± 0.31 (21.08)	66.32M
B3	21.55 ± 0.26 (21.94)	66.32M
B5	21.60 ± 0.28 (22.03)	66.32M
B10	21.57 ± 0.26 (21.91)	66.32M
B25	21.47 ± 0.30 (21.77)	66.32M
B100	21.10 ± 0.31 (21.39)	66.32M
B10-LP-0.5	21.71 ± 0.25 (22.04)	66.32M
B10-LP-1.0	21.80 ± 0.25 (22.16)	66.32M

Table 5. Tuning Results, Beam Search Translation Method (Chang, 2018)

Dim	Time sec/epoch	Params	Accuracy
128	1000	128x342	0.965
512	1200	512x342	0.970
1024	1250	1024x342	0.983

Table 6. Embedded hidden sizes and accuracy and training time.

Embedding hidden size 1024 gives the best accuracy however increases training time significantly. We decided to use size 128 which still provided acceptable accuracy.

References

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate (cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation)

Benefits from Neural Machine Translation: Next-Level Translation for Everyone. (2019, September 8).
<https://pangeamt.com/ai-weekly/neural-machine-translation-next-level-translation-for-everyone>.

Brownlee, J. (2019, August 14). *What is Teacher Forcing for Recurrent Neural Networks?*
<https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/>.

Brownlee, J. (2020, September 2). *How to Use Word Embedding Layers for Deep Learning with Keras.* <https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/>.


Chang, V. (2018, November 11). *Hyperparameter tuning for NMT.*
<https://medium.com/@vina.wt.chang/hyperparameter-tuning-for-nmt-af44fbcecccb>.

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling (cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop)

Garr, B. L. (2017, May 4). *Why is Machine Translation Soooo Hard?*
<https://medium.com/@briangarr/why-is-machine-translation-soooo-hard-a0a4983084fd>.

Hofstadter, D. (2018, January 30). *The Shallowness of Google Translate.*
<https://www.theatlantic.com/technology/archive/2018/01/the-shallowness-of-google-translate/551570/>.

Hore, P., & Chatterjee, S. (2019, November 28). *Attention Mechanism In Deep Learning: Attention Model Keras.*
<https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning>.

Kamsetty, A., Fricke, K., & Liaw, R. (2020, August 25). *Hyperparameter Optimization for  Transformers: A guide.*
<https://medium.com/distributed-computing-with-ray/hyperparameter-optimization-for-transformers-a-guide-c4e32c6c989b>.

Kay, M. *Machine Translation*. <https://www.linguisticsociety.org/resource/machine-translation>.

Khandelwal, R. (2018, November 1). *Recurrent Neural Network-RNN*.
<https://medium.com/datadriveninvestor/recurrent-neural-network-rnn-52dd4f01b7e8>.

Khandelwal, R. (2020, February 17). *Implementing Neural Machine Translation with Attention using Tensorflow*.
<https://towardsdatascience.com/implementing-neural-machine-translation-with-attention-using-tensorflow-fc9c6f26155f>.

Macherey, K., Och, F. J. & Ney, H. (2001). Natural Language Understanding Using Statistical Machine Translation. European Conference on Speech Communication and Technology (EUROSPEECH) (pp. 2205--2208), September, Aalborg, Denmark.

MT Features: Neural Machine Translation.
<https://www.tilde.com/products-and-services/machine-translation/features/neural-translation>.

Ruder, S. (2016, January 19). *An overview of gradient descent optimization algorithms*.
<https://ruder.io/optimizing-gradient-descent/>.

Stewart, M. (2020, July 29). *Neural Network Optimization*.
<https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0>.

Weaver, W. (1949/1955). Translation. In W. N. Locke & A. D. Boothe (ed.), *Machine Translation of Languages* (pp. 15--23) . MIT Press .