Neural Machine Translation by Jointly Learning to Align and Translate

BENAISSA Tarek, BENBRAHIM Mohammed El Amine, BENSLIMANE Zahra Hafida, BENABID M'hamed Djahid

Editor: Machine Learning Avancé (2021-2022)

Abstract

As a part of Advanced Machine Learning project we re-implemented a neural machine translation, The paper introduced a model that translates one sentence from a source language into a sentence in a target language.

This model consists of 3 major parts, an encoder,reads the input sentence and encodes it into a sequence of vectors of variable length, and a decoder that outputs a sequence of word probability vectors in the target language and attention ...

1. Introduction

Neural machine translation is task that's becoming more and more essential for many industries of today's world. The overwhelming majority of the neural machine translation models are of the encoder-decoder type. Our role in this project is to reverse engineer the results obtained in the paper "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE" by Bahdanau, Cho and Bengio. The paper introduces an innovation to the classic encoder-decoder model by learning to align and translate jointly.

2. Neural Machine Translation

The global architecture of the model proposed by the paper consists of 3 main parts. the encoder, the attention model and the decoder. the graphical illustration of the model is shown below :

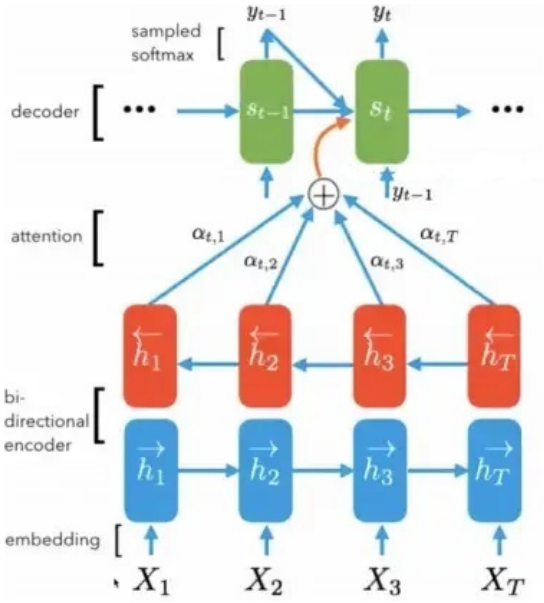

FIGURE 1 – Neural Machine Translation with attention overview

2.1 Encoder

The encoder is a recursive neural network (RNN) that reads an input sequence x starting from the first word to the last one. However, in our model we have used a Bidirectional RNN which are two RNNs one running forward over the input sequence and the other running backward which lead to output two hidden states for each input, the forward hidden state contain information from earlier in the sequence and the backward from later in the sequence, by concatenating the two hidden states it results an annotation h_j for each word.
the annotation is expressed by :

$$h_j = \begin{bmatrix} \vec{h}_j \\ \overleftarrow{h}_j \end{bmatrix}$$

Where the forward hidden state is given by :

$$\vec{h}_j = \begin{cases} (1 - \vec{z}_j) \circ \vec{h}_{j-1} + \vec{z}_j \circ \vec{h}_j & , \text{ if } j > 0 \\ 0 & , \text{ if } j = 0 \end{cases}$$

the annotation will be used for alignment and the decoder later.

2.2 Attention

One of the problems in machine translation is Alignment because it identifies which part of the input sequence can match with each word in the output, so Bahdanau attention is proposed, it is an interface between the encoder and decoder that provides the decoder with information from every encoder hidden state. With this setting, the model is able to selectively focus on useful parts of the input sequence and hence, learn the alignment between them. As explained in the previous section, the annotations that were made by the encoder are used to computed the context vector c_i which is

a weighted sum of the annotations h_i and is given by :

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

The weight a_{ij} of each hidden state h_j, which is call attention weight , is computed by

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}$$

e_{ij} is the alignment where :

$$e_{ij} = \alpha\left(s_{i-1}, h_j\right)$$

We can interpret the values of α as it tell us how much attention the model should pay to the word in the input when generating a certain output.

2.3 Decoder

The context vector that is obtained in the previous step is concatenated with the previous decoder output and fed into the Decoder RNN cell to produce a new hidden state. Then,the process repeats itself for each time step of the decoder until an ' $< end >'$ token is produced or output is past the specified maximum length. The final output for the time step is obtained by passing the new hidden state through a Linear layer, which acts as a classifier to give the probability scores of the next predicted word.

3. Datasets

The WMT' 14 dataset contains approximately 850M words, during our research the time needed for our model to train on the totallity of the dataset would be closer to 6 days. Evidently, we chose to shrink the dataset to a couple hundred thousand words, to facilitate the process and make sure our python code was correct before moving into the full dataset. We chose to concatenate the Europarl dataset with the IWSLT-17 dataset to form our mini-set