# Title

Neural Machine Translation by Jointly Learning to Align and Translate

# Author

Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio

# Abstract

The models proposed recently for neural machine translation often belong to a family of **encoder–decoders** and **encode a source sentence into a fixed-length vector from which a decoder generates a translation**. Author proposed **attention mechanism,** allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly.

# Issue

A potential issue with this encoder–decoder approach 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**, especially those that are longer than the sentences in the training corpus.

# Method

## RNN Encoder–Decoder

In the Encoder–Decoder framework, an encoder reads the input sentence, a sequence of vectors , into a vector . The most common approach is to use an RNN such that

and

where is a hidden state at time , and is a vector generated from the sequence of the hidden states. and are some nonlinear functions.

The decoder is often trained to predict the next word given the context vector and all the previously predicted words . In other words, the decoder defines a probability over the translation by decomposing the joint probability into the ordered conditionals:

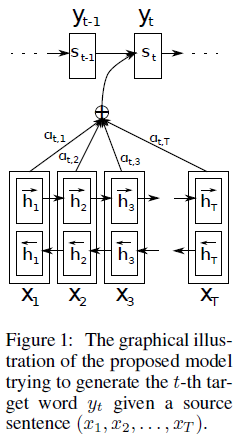
where . With an RNN, each conditional probability is modeled as

where is a nonlinear, potentially multi-layered, function that outputs the probability of , and is the hidden state of the RNN.

## Attention (Learning to Align and Translate)

The new architecture consists of a **bidirectional RNN** as an encoder and a decoder that emulates searching through a source sentence during decoding a translation.

### Decoder: General Description

In a new model architecture, we define each conditional probability as:

where is an RNN hidden state for time , computed by

Here the probability is conditioned on a distinct context vector for each target word . The context vector is, then, computed as a weighted sum of these annotations :

The weight of each annotation is computed by

where

is **an alignment model which scores how well the inputs around position and the output at position match**. The score is based on the RNN hidden state (just before emitting yi) and the j-th annotation of the input sentence.

### Encoder: Bidirectional RNN for Annotating Sequences

A BiRNN consists of **forward and backward RNN’s**. The forward RNN reads the input sequence as it is ordered (from to ) and calculates a sequence of forward hidden states . The backward RNN reads the sequence in the reverse order (from to ), resulting in a sequence of backward hidden states .

We obtain an annotation for each word by concatenating the forward hidden state and the backward hidden state , i.e., . In this way, **the annotation**  **contains the summaries of both the preceding words and the following words**.