[Review] Neural Machine Translation by jointly learning to align and translate ICLR 2015

Hankyo Jeong

August 8, 2024

Seoul National University

Abstract

- Machine translation
  - statistical MT $>>$ neural MT
- Encoder - Decoder
  - encode a source sentence into a fixed-length vector from which a decoder generates a translation. $>>$ bottleneck
- Automatically (soft)search source sentence
  - What are relevant to prediction to target word?
  $=$ soft alignment
  $=$ attention

Introduction

- Neural Machine Translation
- (basic) Encoder - Decoder
- Issue
  - fixed-vector representation >> bottleneck
  - necessary information of source sentence must be compressed
  - long input sentence: low performance
- (proposed) Encoder - Decoder: learns to align and translation jointly.
  - each time the model generates a word in a translation, it (soft)searches for a set of positions in a source sentence where the most relevant information is concentrated.
  - Then predicts the target word based on the context vectors associated with these source positions and previously generated target words.
  - encodes the input sentences into a sequence of vectors and choose a subset of these vectors adaptively while decoding the translation.

Background: NEURAL MACHINE TRANSLATION

- Translation: $argmax_y P(\mathbf{y}|\mathbf{x})$

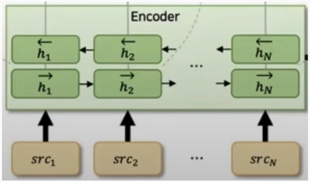
▶ RNN Encoder-Decoder
- $\mathbf{x} = (x_1, ..., x_{Tx})$ sequence of vectors (input)
- $c = q(\{h_1, ..., h_{Tx}\})$: fixed-length representation generated from the sequence of the hidden states.
- $h_t = f(x_t, h_{t-1})$: $h_t \in R^n$ is a hidden state at time t, f and q are some nonlinear functions, for instance 'Sutskever et al. (2014)' used f as LSTM and $q(\{h_1, ..., h_{Tx}\} = h_{Tx}$
- Decoder: predict next word given the context vector and previously predicted words.

$$p(y) = \prod_{t=1}^{T} p(y_t|\{y_1, ..., y_{t-1}, c)$$
$$p(y_t|\{y_1, ..., y_{t-1}, c) = g(y_{t-1}, s_t, c)$$

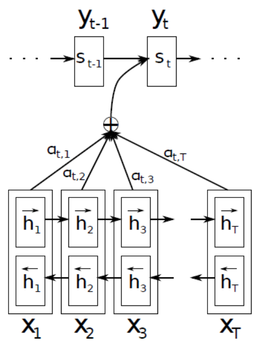
where g is a nonlinear, potentially multi-layered, function that outputs the prob of y_t and s_t is the hidden state of RNN.

LEARNING TO ALIGN AND TRANSLATE

- Encoder: Bidirectional RNN



- Decoder: searches through a source sentences



ENCODER: BIDIRECTIONAL RNN FOR ANNOTATING SEQUENCES

- RNN: reads input sequence in order from x_1 to x_{Tx} (forward)
- BiRNN: forward RNN($\overrightarrow{f}$) + backward RNN($\overleftarrow{f}$)
  - forward RNN:
  calculates a sequence of forward hidden states $((\overrightarrow{h}_1, ..., \overrightarrow{h}_{Tx})$.
  - bakcward RNN:
  calculates a sequence of backward hidden states $((\overleftarrow{h}_1, ..., \overleftarrow{h}_{Tx})$.
- Obtain annotation for each word x_j by concatenating the forward hidden state and backward hidden state. $h_j = [\overrightarrow{h}_j^T, \overleftarrow{h}_j^T]^T$
  - In this way the annotation h_j contains the summaries of both the preceding words and the following words.

DECODER: GENERAL DESCRIPTION

- Each conditional probability defined as:

  $$p(y_i|\{y_1, ..., y_{i-1}, x) = g(y_{i-1}, s_i, c_i)$$

  where s_i is an RNN hidden state for time i, computed by

  $$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

  conditioned on a distinct context vector c_i for each target word y_i

- c_i depends on a sequence of annotations $(h_1, ..., h_{Tx})$ computed

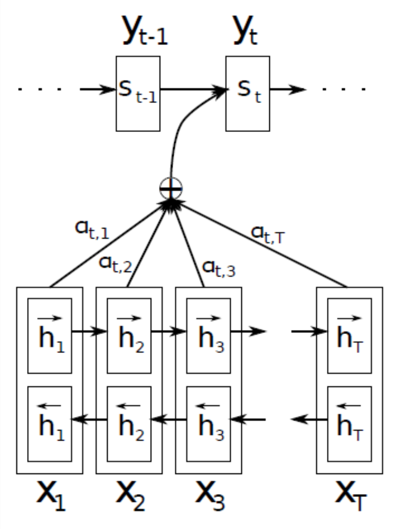
  $$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

  The weight $\alpha_{ij}$of each annotation h_j is computed by

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

  where $e_{ij} = a(s_{i-1}, h_j)$ is an alignment model which scores how well the inputs around position j and the output at position i match.

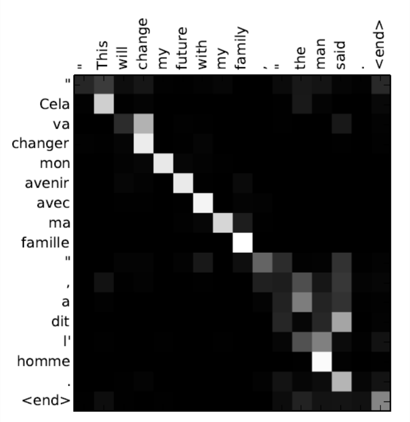
DECODER: GENERAL DESCRIPTION



EXPERIMENT SETTINGS

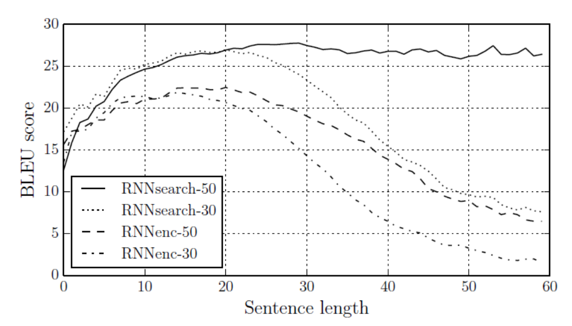
- English-to-French translation
- Train two types of models
  - Conventional RNN encoder-decoder vs Proposed RNNsearch
- encoder and decoder of the RNN have 1000 hidden units each, single maxout hidden layer for each target word.
- SGD, Adadelta

RESULTS



Each cell: attention score α_{ij}, balck: 0, white: 1

RESULTS



proposed model is robust than basic RNN model