Neural Machine Translation by Jointly Learning to Align and Translate

Reporter: Fandong Meng

Institute of Computing Technology
Chinese Academy of Sciences

mengfandong@ict.ac.cn

October 11, 2014

To appear in *proceedings of NIPS 2014*. [Download]

Overview

1. Motivation
2. RNN Encoder-Decoder
3. Learning to Align and Translate
   - Decoder: Gerenal Description
   - Encoder: Bidirectional RNN for Annotation Sequences
   - Hidden Unit that Adaptively Remembers and Forgets
4. Experiments
5. Conclusion

Motivation

Framework

Basic RNN encoder-decoder based machine translation.

- Fit a parameterized model to maximize the conditional probability of target sentence y given a source sentence x, i.e., $argmax_y p(y|x)$ using a training parallel corpus.
- Compress all the necessary information of a source sentence into a fixed-length vector.
- Decode the vector into a variable-length target sentence.

Motivation

Issues

Compressing all the necessary information of a source sentence into a fixed-length vector 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.

- To address this issue, this paper introduces an extension to the encoder-decoder model which learns to align and translate jointly.

Motivation

Distinguish from the basic encoder-decoder

It encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding.

- To generate a word in a translation, the model searches for a set of positions in a source sentence where the most relevant information is concentrated.
- The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

RNN

- Basic RNN framework

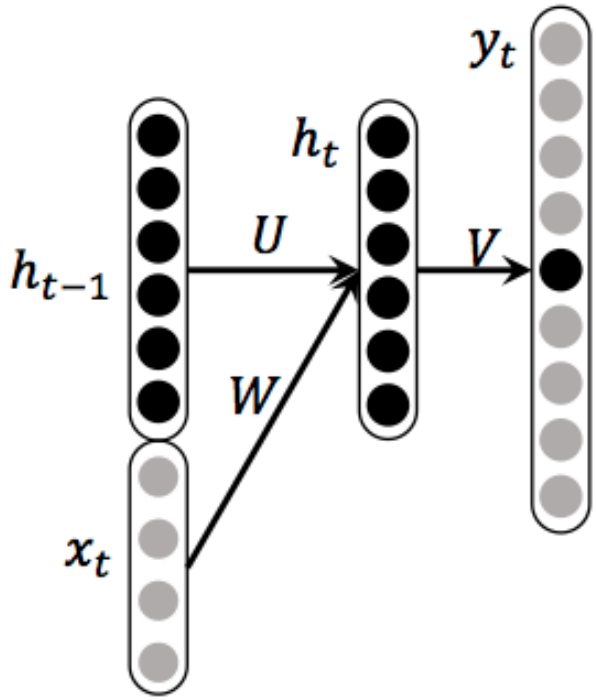

Figure : Basic RNN framework.

RNN Encoder-Decoder: Encoder

- In the Encoder-Decoder framework, an encoder reads the input sentence, a sequence of vectors $x = (x_1, ..., x_{T_x})$ into a vector c,

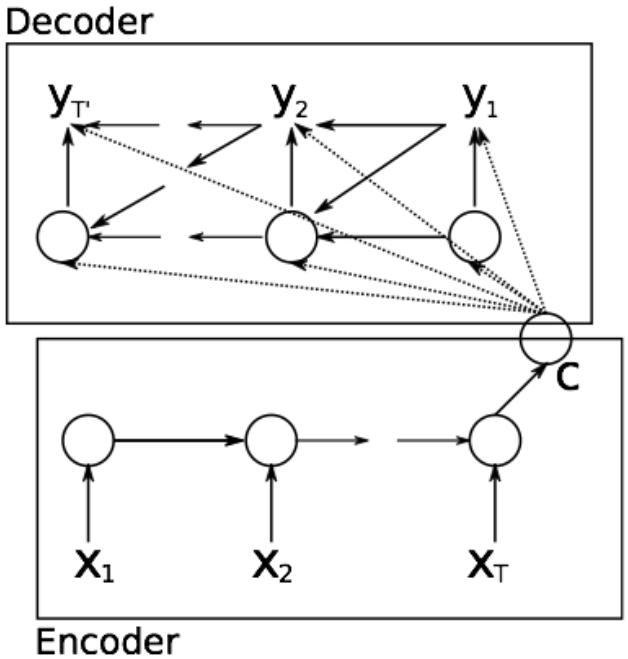

Figure : Basic encoder-decoder framework.

where $h_t = f(x_t, h_{t-1})$ and $c = q(h1, ..., h_{T_x})$, f and q are some nonlinear functions.

RNN Encoder-Decoder: Decoder

- The decoder is often trained to predict the next word y_t given the context vector c and all the previously predicted words $\{y_1, ..., y_{t-1}\}$.
- In other words, the decoder defines a probability over the translation $\mathbf{y}$ by decomposing the joint probability into the ordered conditionals:

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

where $\mathbf{y} = \{y_1, ..., y_{T_y}\}$. With an RNN, each conditional probability is modeled as

$$p(y_t | \{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \tag{2}$$

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

RNN Encoder-Decoder

Context vector c is a constant vector!

Learning to Align and Translate

Introduce two parts in detail.

- Decoder: Gerenal Description
- Encoder: Bidirectional RNN for Annotation Sequences

Decoder: Gerenal Description

Derived in steps.

- The conditional probability is

$$p(y_i|\{y_1,...,y_{i-1}\},\mathbf{x}) = g(y_{i-1},s_i,c_i) \tag{3}$$

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

$$s_i = f(s_{i-1},y_{i-1},c_i). \tag{4}$$

- Here the probability is conditioned on a **distinct** context vector c_i for each target word y_i, which is different from basic RNN decoder.

Decoder: Gerenal Description

Derived in steps.

- In $s_i = f(s_{i-1}, y_{i-1}, c_i)$, the context vector c_i depends on a sequence of annotations $(h_1, ..., h_{Tx})$ to which an encoder maps the input sentence.
- Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the $i-th$ word of the input sequence.
- How to compute c_i?

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \tag{5}$$

Decoder: Gerenal Description

Derived in steps.

- In $c_i = \sum_{j=1}^{T_x} a_{ij} h_j$, the weight a_{ij} of each annotation h_j is computed by

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{6}$$

where $e_{ij} = a(s_{i-1}, h_j)$, a is the so called alignment model, can be jointly trained with all the other components of the proposed system.

Decoder: Gerenal Description

Derived in steps.

- The probability a_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next hidden state s_i and generating y_i.

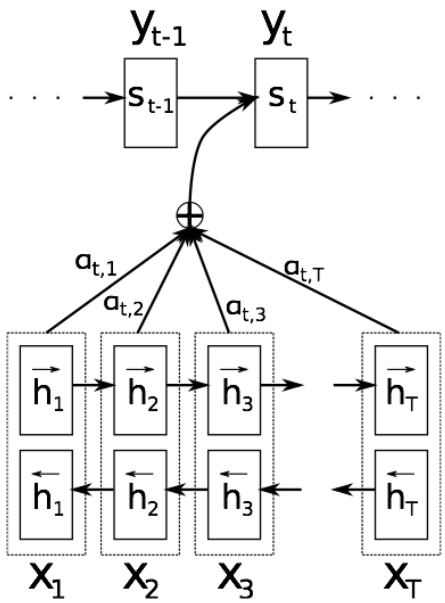

Figure : The graphical illustration of the proposed model trying to generate the $t-th$ target word y given a source sentence $(x_1, ..., x_T)$.

Decoder: Gerenal Description

Derived from top to down.

- $p(y_i|\{y_1,...,y_{i-1}\},\mathbf{x}) = g(y_{i-1}, s_i, c_i)$
- $s_i = f(s_{i-1}, y_{i-1}, c_i)$
- $c_i = \sum_{j=1}^{T_x} a_{ij} h_j$
- $a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$
- $e_{ij} = a(s_{i-1}, h_j)$

What are left?

- $a(s_{i-1}, h_j)$
- h_j

Decoder: Gerenal Description

Alignment model

- Use a single-layer multilayer perceptron.

$$a(s_{i-1}, h_j) = v_a^T tanh(W_a s_{i-1} + U_a h_j) \tag{7}$$

where v_a, W_a, U_a are weight matrices, can be jointly trained with all the other components of the proposed system.

Decoder: Gerenal Description

Derived from top to down.

- $p(y_i|\{y_1,...,y_{i-1}\},\mathbf{x}) = g(y_{i-1}, s_i, c_i)$
- $s_i = f(s_{i-1}, y_{i-1}, c_i)$
- $c_i = \sum_{j=1}^{T_x} a_{ij} h_j$
- $a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$
- $e_{ij} = a(s_{i-1}, h_j)$
- $a(s_{i-1}, h_j) = v_a^T tanh(W_a s_{i-1} + U_a h_j)$

What is left now?

- h_j

Encoder: Bidirectional RNN for Annotation Sequences

A BiRNN consists of forward and backward RNNs.

- The forward RNN $\overrightarrow{f}$ reads the input sequence as it is ordered (from x_1 to x_{T_x}) and calculates a sequence of forward hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_{T_x})$.
- The backward RNN $\overleftarrow{f}$ reads the input sequence as it is ordered (from x_{T_x} to x_1) and calculates a sequence of forward hidden states $(\overleftarrow{h}_1, ..., \overleftarrow{h}_{T_x})$.
- Therefore, $h_j = [\overrightarrow{h}_j^T; \overleftarrow{h}_j^T]$

Encoder: Bidirectional RNN for Annotation Sequences

Take a look at the annotation h again.

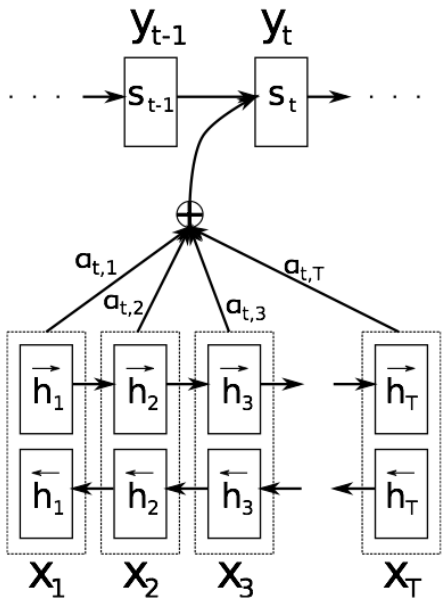

Figure : The graphical illustration of the proposed model trying to generate the $t-th$ target word y given a source sentence $(x_1, ..., x_T)$.

Hidden Unit that Adaptively Remembers and Forgets

A new type of hidden unit.

In $s_i = f(s_{i-1}, y_{i-1}, c_i)$, f may be as simple as an element-wise logistic sigmoid function and as complex as a long short-term memory (LSTM) unit. This paper used a new type of hidden unit (Cho et al., 2014)) that has been motivated by the LSTM unit but is much simpler to compute and implement. Let us describe how the activation of the jth hidden unit is computed:

- The hidden state s_i of the decoder given the annotations from the encoder is computed by

$$s_i = (1 - z_i) \circ s_{i-1} + z_i \circ \widetilde{s}_i \tag{8}$$

where

$$\widetilde{s}_i = tanh(WEy_{i-1} + U[r_i \circ s_{i-1}] + Cc_i)$$
$$z_i = sigmod(W_z Ey_{i-1} + U_z s_{i-1} + C_z c_i)$$
$$r_i = sigmod(W_r Ey_{i-1} + U_r s_{i-1} + C_r c_i)$$

Hidden Unit that Adaptively Remembers and Forgets

A new type of hidden unit.

Look at the follow example, we just take h as s and take X as Y.

$$
\begin{aligned}
s_i &= f(s_{i-1}, y_{i-1}, c_i) = (1 - z_i) \circ s_{i-1} + z_i \circ \widetilde{s}_i \\
\widetilde{s}_i &= tanh(WEy_{i-1} + U[r_i \circ s_{i-1}] + Cc_i) \\
z_i &= sigmod(W_z Ey_{i-1} + U_z s_{i-1} + C_z c_i) \\
r_i &= sigmod(W_r Ey_{i-1} + U_r s_{i-1} + C_r c_i)
\end{aligned}
$$

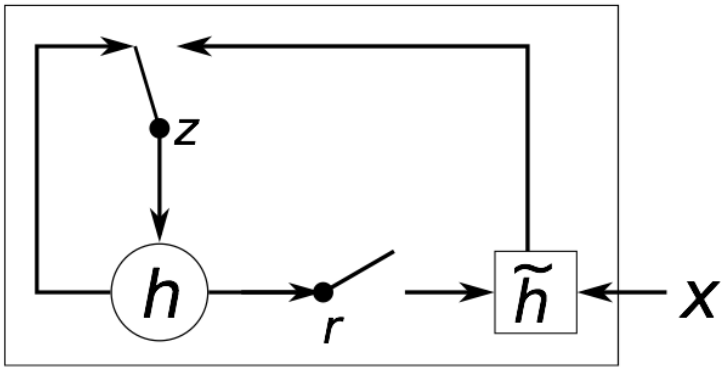

Figure : An illustration of the proposed hidden activation function. The update gate z selects whether the hidden state is to be updated with a new hidden state $\widetilde{h}$. The reset gate r decides whether the previous hidden state is ignored.

Experiments

Setup

- Training Set
  - WMT 2014 English-French parallel corpora
  - http://www.statmt.org/wmt14/translation-task.html
- Development Set
  - news-test-2012 and news-test-2013
- Test Set
  - news-test-2014
- Other Details
  - data selection
  - 30,000 most frequent words
  - map other words to [UNK]

Experiments

Results with respect to the lengths of the sentences.

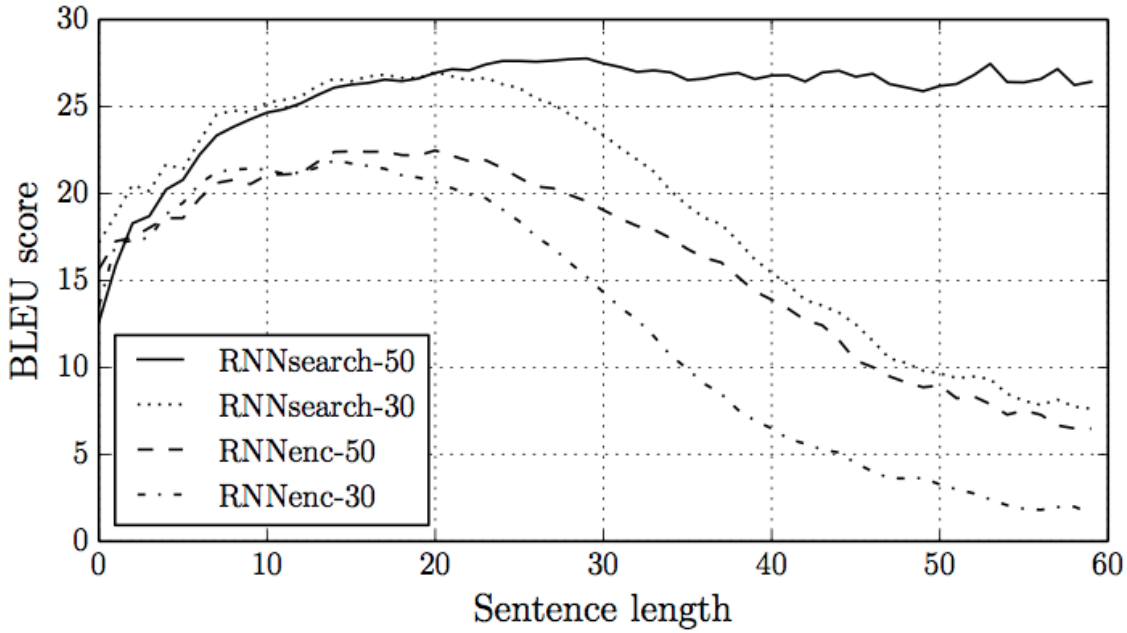

Figure : The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Experiments

Main Results

Model	All	No UNK°
RNNenc-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNenc-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50⋆	28.45	36.15
Moses	33.30	35.63

Figure : BLEU scores of the trained models computed on the test set. The second and third columns show respectively the scores on all the sentences and, on the sentences without any unknown word in themselves and in the reference translations. Note that RNNsearch-50⋆ was trained much longer until the performance on the development set stopped improving. We disallowed the models to generate [UNK] tokens when only the sentences having no unknown words were evaluated (last column).

Conclusion

- Extend the basic encoderCdecoder by letting a model search for a set of input words, or their annotations computed by an encoder, when generating each target word. This novel architecture makes it better for long sentence translation.
- One of challenges left for the future is to better handle unknown, or rare words.

The End! Thanks!