Neural Machine Translation by Jointly Learning to Align and Translate

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio**
Université de Montréal

Presented by Xiyang Chen

Outline

The Task

The model takes a source sentence of 1-of-K coded word vectors as input

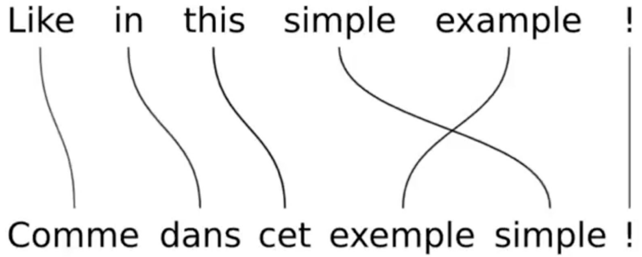
$$\mathbf{x} = (x_1, \ldots, x_{T_x}),\ x_i \in \mathbb{R}^{K_x}$$

and outputs a translated sentence of 1-of-K coded word vectors

$$\mathbf{y} = (y_1, \ldots, y_{T_y}),\ y_i \in \mathbb{R}^{K_y},$$

where K_x and K_y are the vocabulary sizes of source and target languages, respectively. T_x and T_y respectively denote the lengths of source and target sentences.

… a neural network that translates…

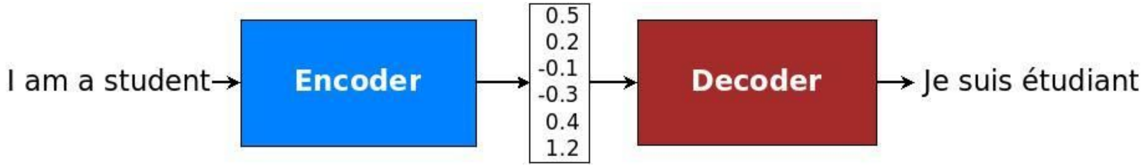

Like in this simple example !
Comme dans cet exemple simple !

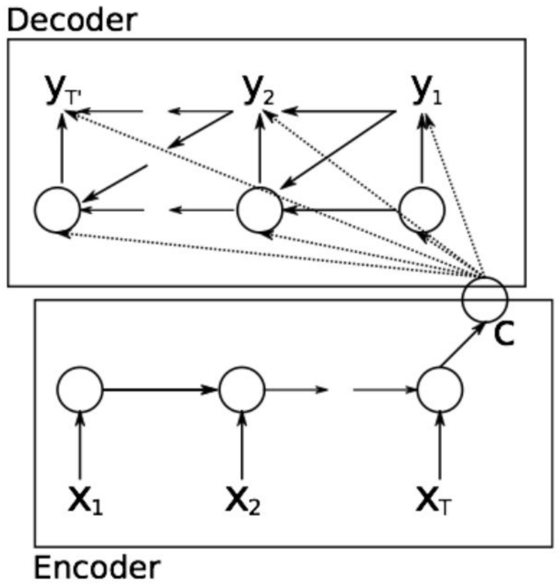
Goal: we want to estimate

$$\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$$

RNN Encoder-decoder (baseline)

- Two components





Sutskever et al. 2014
Cho et al. 2015

The Encoder (baseline)

$$\mathbf{x} = (x_1, \cdots, x_{T_x})$$

$$h_t = f(x_t, h_{t-1})$$

$h_t \in \mathbb{R}^n$ is a hidden state at time t
a.k.a. *annotations*

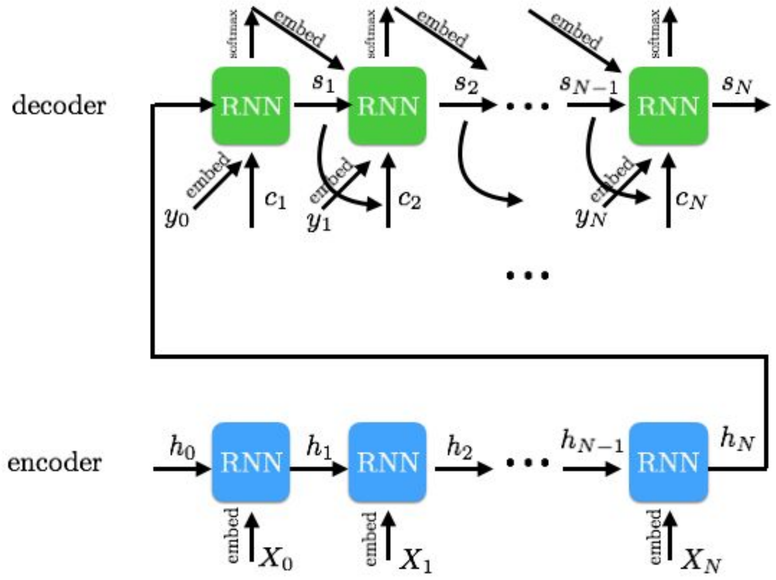
f is a nonlinear function

$$c = q(\{h_1, \cdots, h_{T_x}\})$$

c is a vector generated from the sequence

q can be a nonlinear function. For instance,

$$q(\{h_1, \cdots, h_T\}) = h_T$$



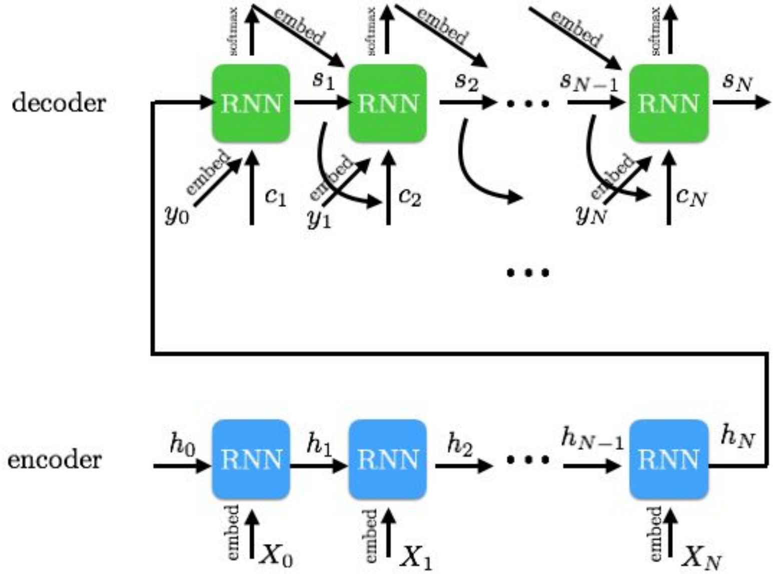
The Decoder (baseline)

$$\mathbf{y} = (y_1, \cdots, y_{T_y})$$

With an RNN, each conditional probability is modeled as

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

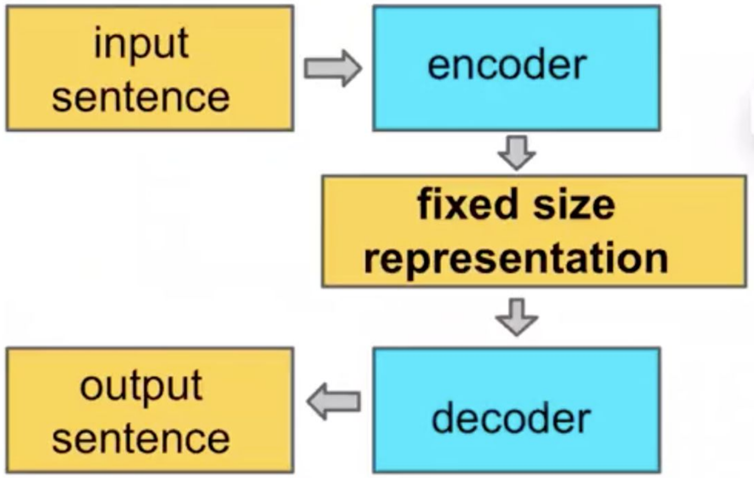
s_t is the hidden state of the RNN



On Notations

- $\boldsymbol{h}$: hidden state(s) of encoder
- $\boldsymbol{s}$: hidden state(s) of decoder
- $\boldsymbol{j}$: for encoder/input
- $\boldsymbol{i}$: for decoder/output

RNN Encoder-decoder (baseline)

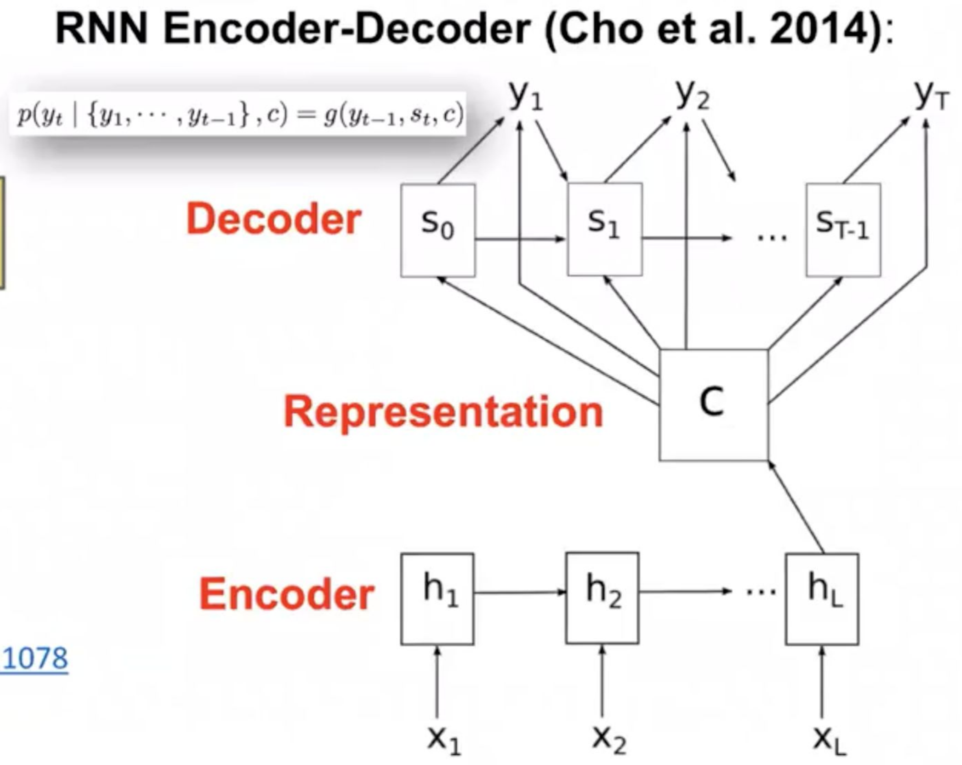


(Ñeco&Forcada, 1997)

(Kalchbrenner et al., 2013)

(Cho et al., 2014) https://arxiv.org/abs/1406.1078

(Sutskever et al., 2014)



RNN Encoder-Decoder (Cho et al. 2014):
$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$
y_1
y_2
y_T
Decoder
s_0
s_1
...
s_{T-1}
Representation
c
Encoder
h_1
h_2
...
h_L
x_1
x_2
x_L

Better than BoW

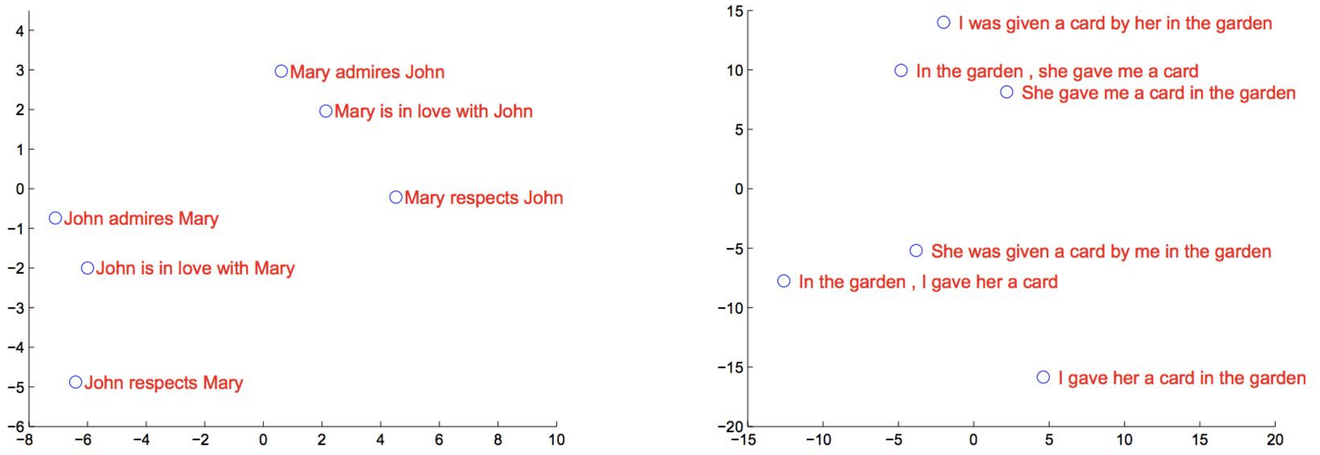


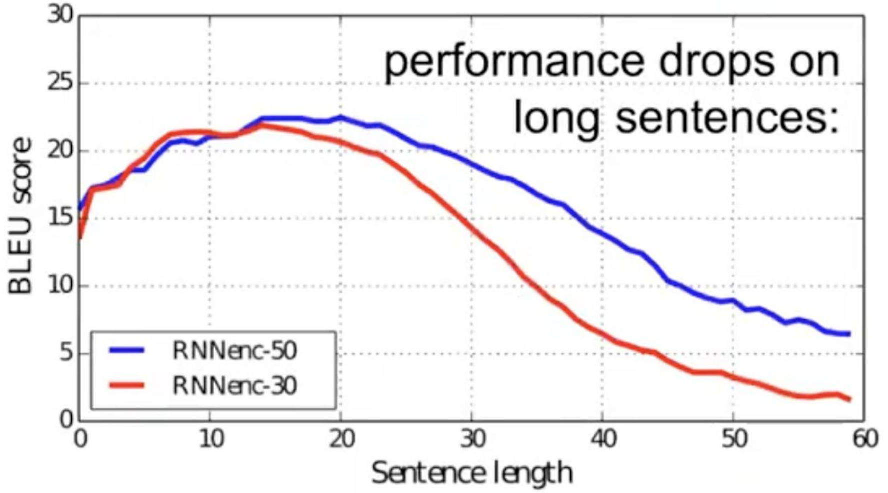
Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

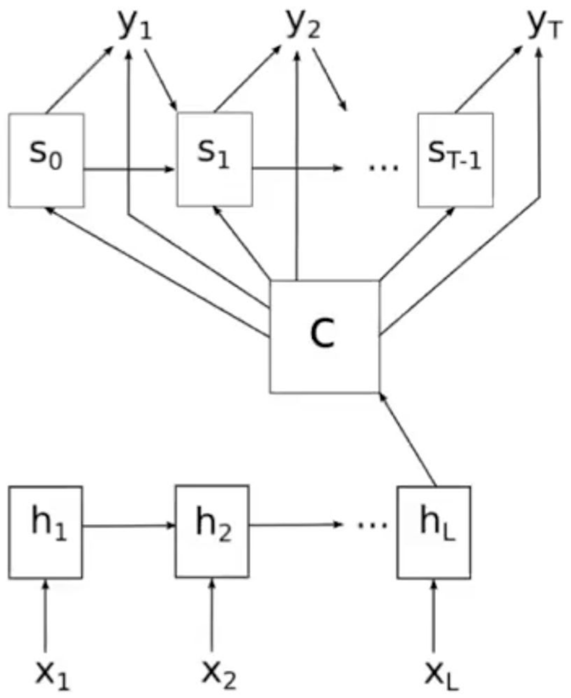
(Sutskever et al., 2014)

- For translation, seq2seq models had achieved performance close to SotA in English-to-French translation tasks

Issues with RNN Encoder-Decoder

- has to remember the whole sentence Even with LSTM/GRU
- fixed size representation can be the bottleneck
- humans do it differently Need to 'attend' to each individual word





Main Contributions of Badanau *et al*.

- Propose a novel architecture for NMT
- The **encoder**: a **bidirectional RNN**
  - the hidden state should encode information from both the previous and following words.
- The **decoder: proposed an extension (attention) model**
  - attention mechanism: a **weighted sum of the input hidden states**

Learning to Align and Translate (attention)

- Basic encoder–decoder:
  - encodes a whole input sentence into a single fixed-length vector
- Encoder-decoder with attention:
  - Encodes the input sentence into a (variable-length) sequence of vectors
  - While decoding the translation, chooses a subset of these vectors adaptively

Learning to Align and Translate (attention)

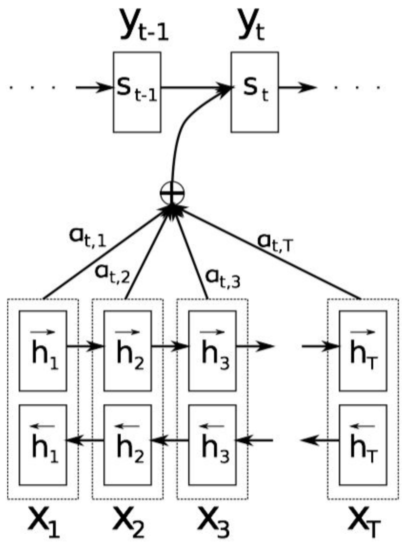


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

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

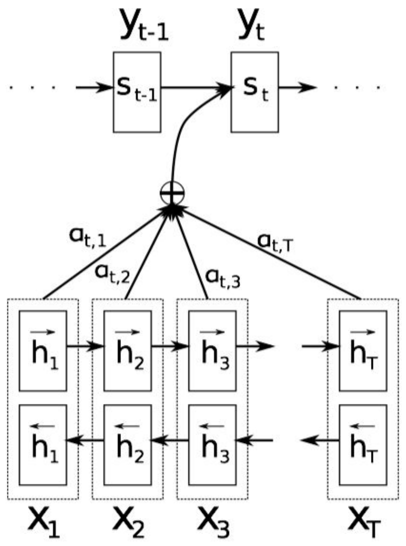
s_i is an RNN hidden state for time i

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

Notice that the context vector varies at each time step.

In other words, probability is conditioned on a distinct contect vector c_i for each target word y_i

Learning to Align and Translate (attention)



y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
x_1 x_2 x_3 x_T

Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

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

For example, in the experiments:

$$s_i = (1 - z_i) \circ s_{i-1} + z_i \circ \tilde{s}_i$$
$$\tilde{s}_i = \tanh\left(W E y_{i-1} + U\left[r_i \circ s_{i-1}\right] + C c_i\right)$$
$$z_i = \sigma\left(W_z E y_{i-1} + U_z s_{i-1} + C_z c_i\right)$$
$$r_i = \sigma\left(W_r E y_{i-1} + U_r s_{i-1} + C_r c_i\right)$$

$\circ$ is an element-wise multiplication,

z_i - updadate gates.
r_i - reset gates.

Expected Annotation

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

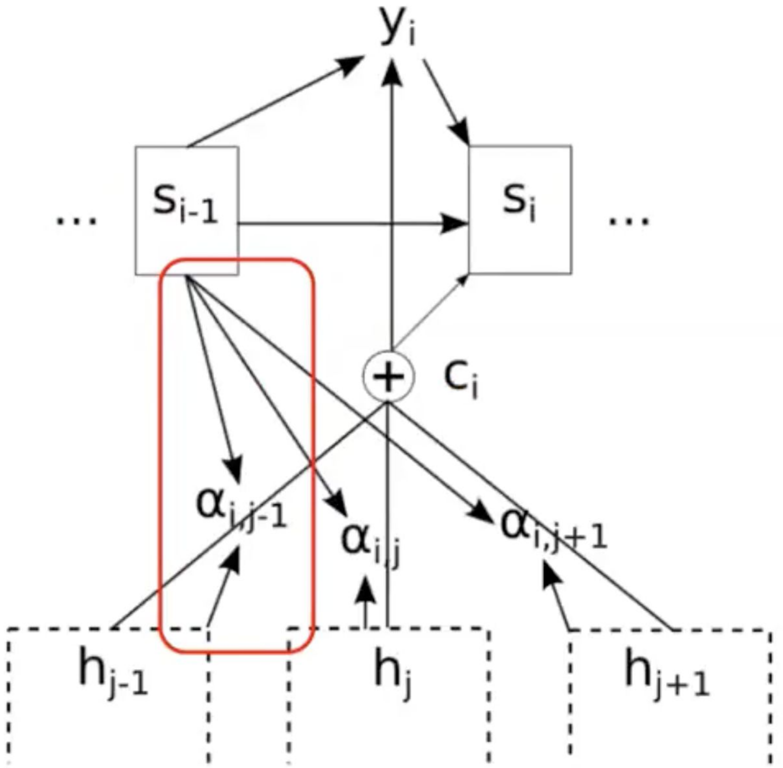
(a.k.a. context or expected annotation for output *i*)

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

(i.e. softmax; a.k.a. weight for each input annotation)

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

(a.k.a. *alignment model*, or *score* on how well position ***i*** (**of output**) and position ***j*** (**of input**) match)



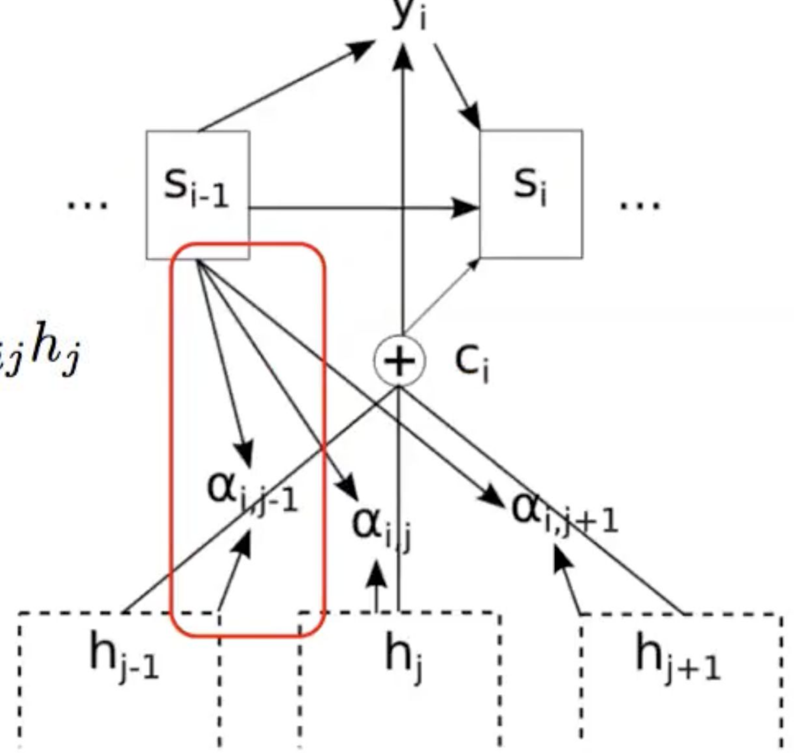
Alignment Model

$$\boxed{e_{ij} = v^T \tanh(W s_{i-1} + V h_j)} \quad (1)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{L} \exp(e_{ik})} \quad (2)$$

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

- nonlinearity (tanh) is crucial!
- simplest model possible
- Vh_j is precomputed => quadratic complexity with low constant



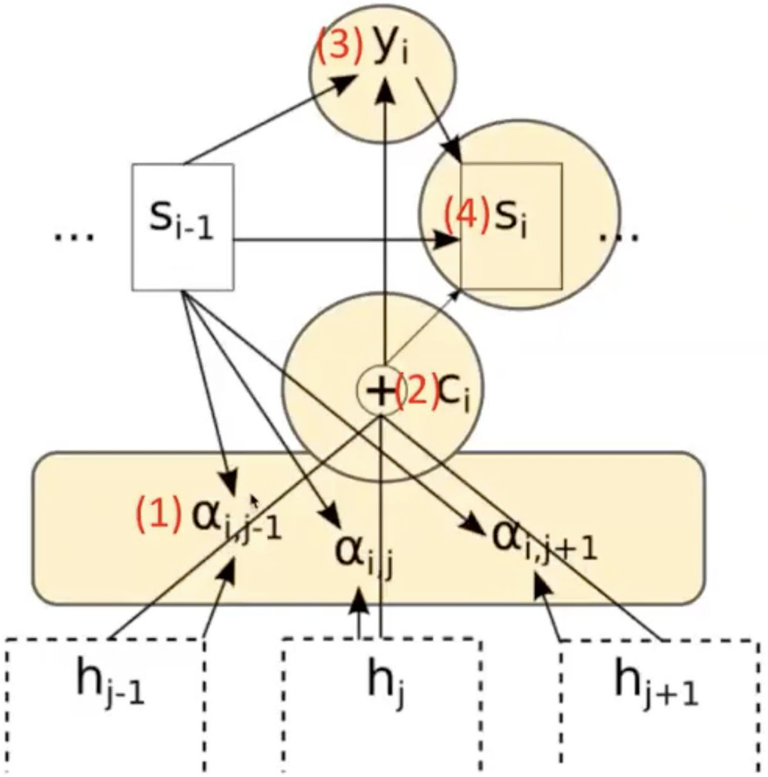
Step i:

(1) compute alignment

(2) compute context

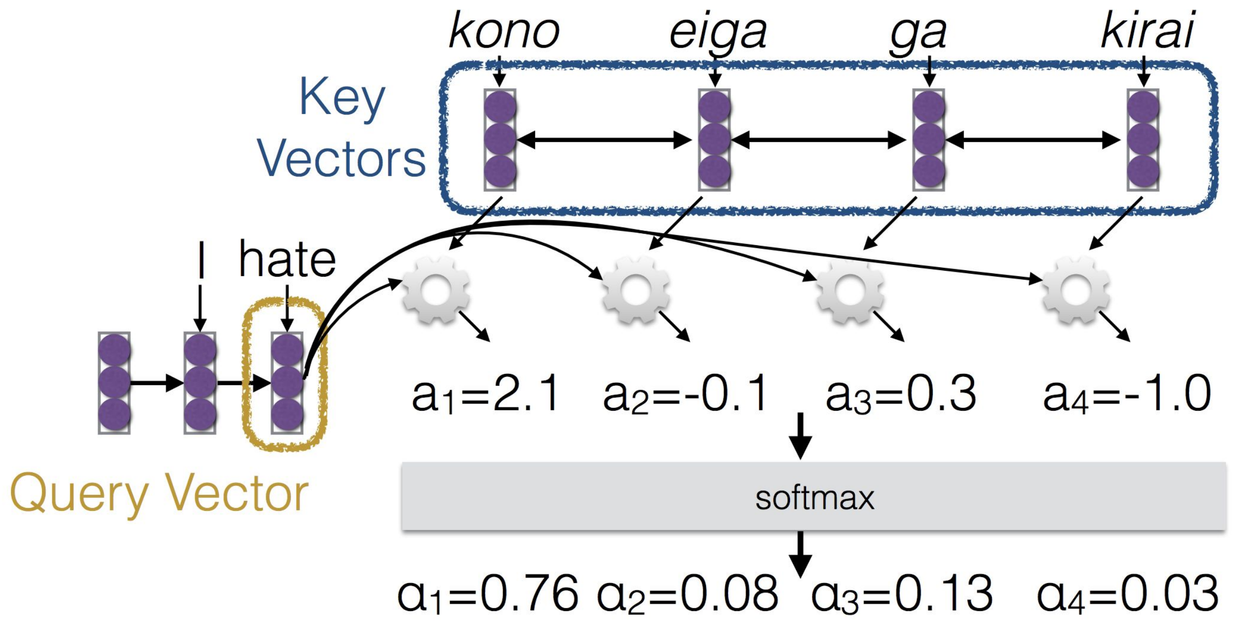
(3) generate new output

(4) compute new decoder state

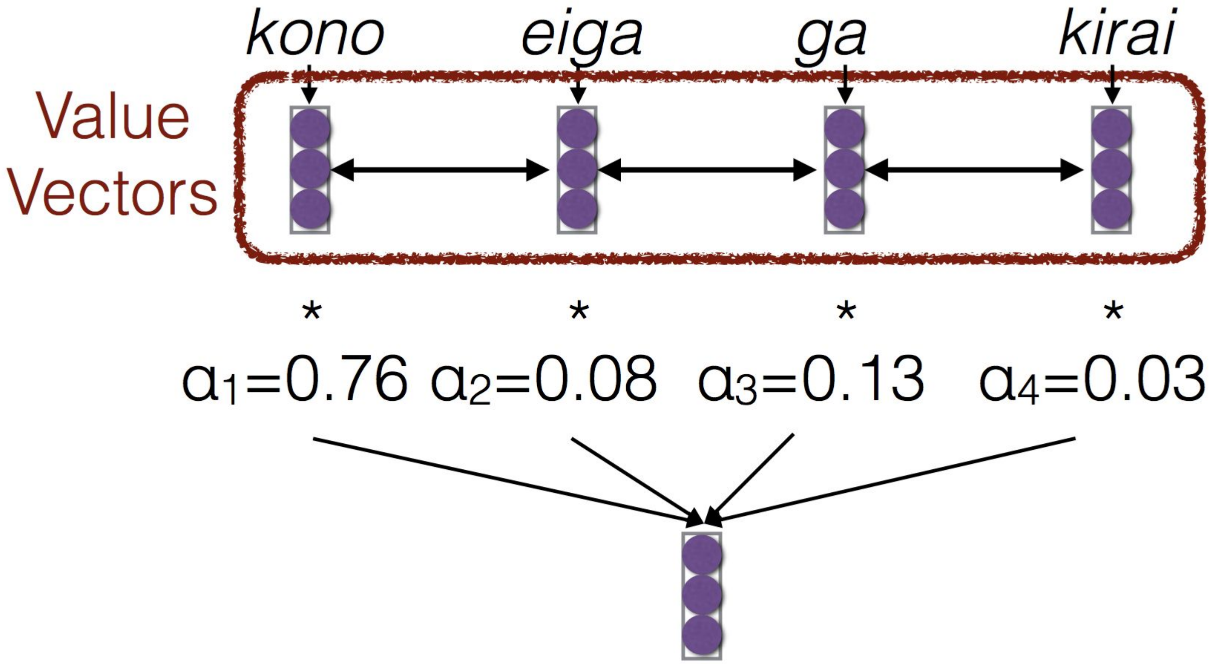


Alternative Interpretation (Q, K, V)

- Use “query” vector (decoder state) and “key” vectors (all encoder states)
- For each query-key pair, calculate weight
- Normalize to add to one using softmax



- Combine together value vectors (usually encoder states, like key vectors) by taking the weighted sum



- Use this in any part of the model you like

Intuitions

- [Our model] does not attempt to encode a whole input sentence into a single fixed-length vector. Instead, it encodes the input sentence into *a sequence of vectors* and *chooses a subset* of these vectors adaptively while decoding the translation.
- Each time the proposed 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
- Interpreted in another way, the attention mechanism is simply giving the network access to its internal memory, which is the hidden state of the encoder.
- With this new approach the *information can be spread* throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly.

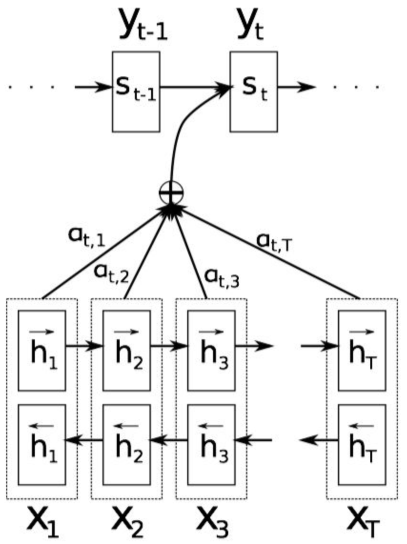
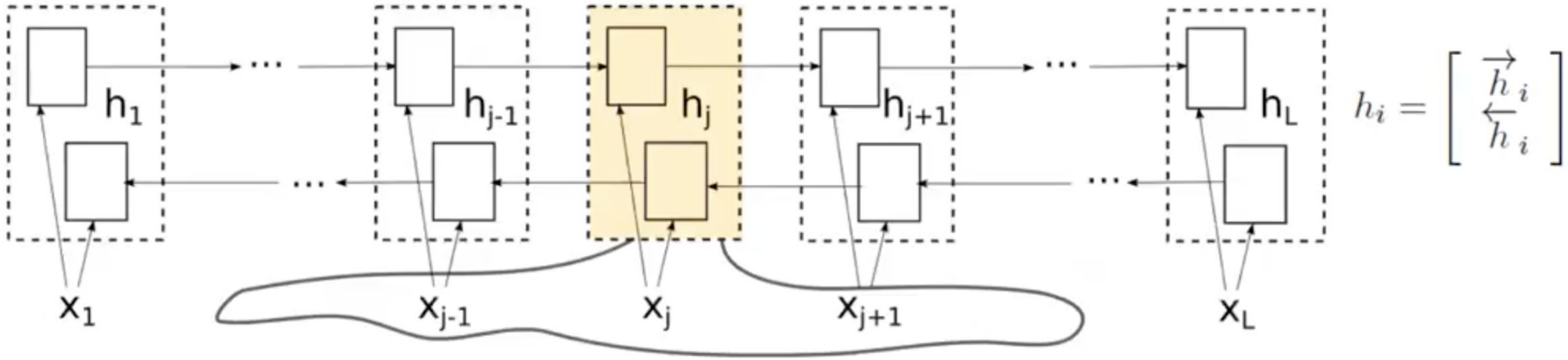


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

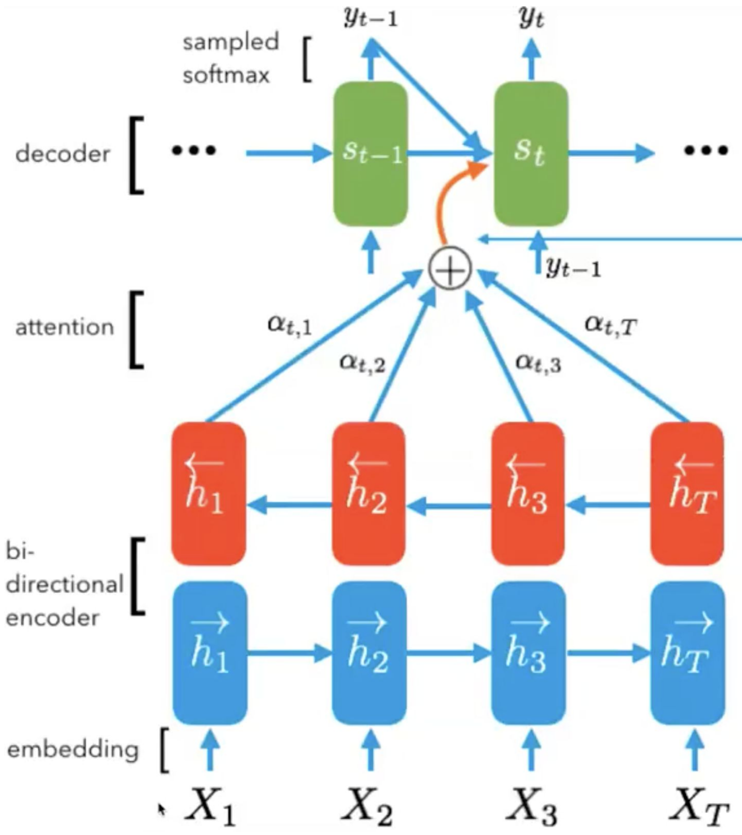
Bidirectional RNN

Bidirectional RNN: h_j contains x_j together with its context (..., x_{j-1}, x_{j+1}, …).

(h_1, …, h_L) is the new *variable-length* representation instead of *fixed-length* c.


h_1
h_{j-1}
h_j
h_{j+1}
h_L
x_1
x_{j-1}
x_j
x_{j+1}
x_L
$h_i = \begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix}$

All put together



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

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

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

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

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

where a is a feed forward neural network.

$$h_j = [\rightarrow h_j; \leftarrow h_j]_{concat}$$

Note that the model becomes RNN Encoder–Decoder if we fix c_i to $\overrightarrow{h}_{T_x}$.

Experiments: English to French

Model:

- RNN Search, 1000 units

Baseline:

- RNN Encoder-Decoder, 1000 units
- Moses, a SMT system (Koehn et al. 2007)

• 1000 hidden units
• Word embedding dimensions: 620
• Beam size 12

Data: http://www.statmt.org/wmt14/translation-task.html (ACL WMT '14)

- English to French translation, 348 million words,
- 30000 words + UNK token for the networks, all words for Moses

Training:

- Minimize mean log P(y|x,θ) w.r. θ
- log P(y|x,θ) is differentiable w.r. θ => usual methods

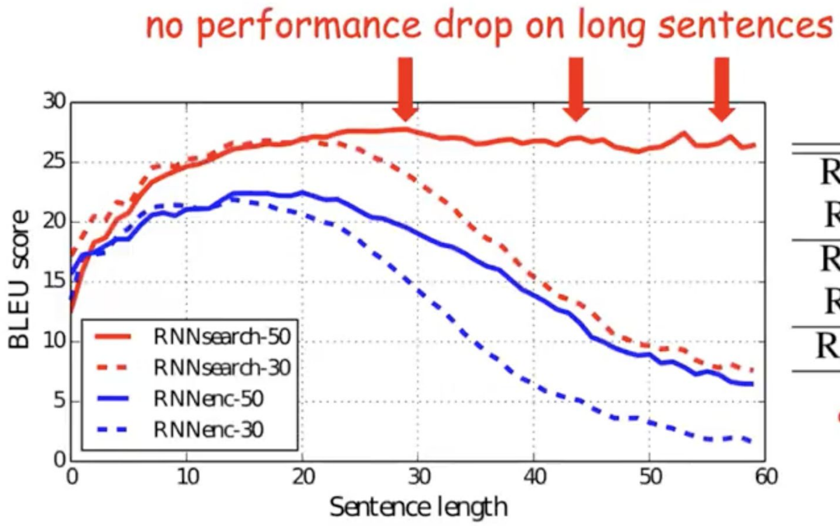
• SGD w/ Adadelta
• mini-batch: 80 sentences

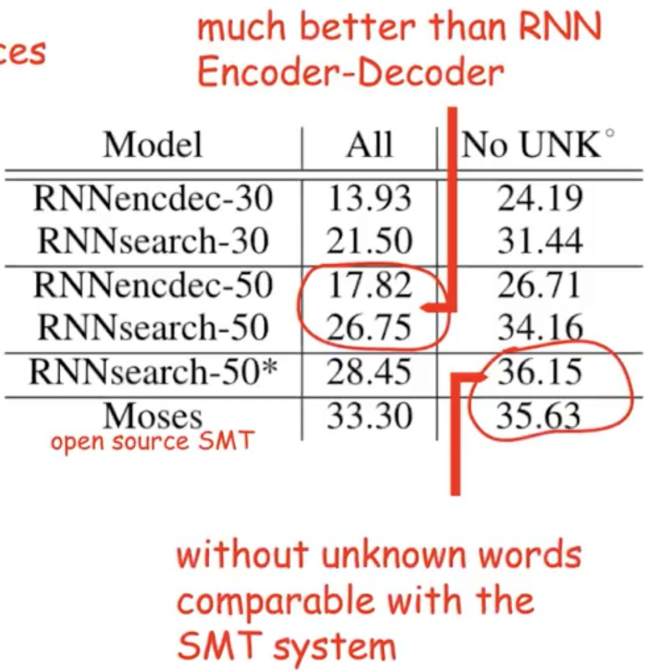
We trained each model for approximately 5 days.

- Beam search + BLEU

Results

- Achieves, *with a single model*, a translation performance comparable, or close, to the conventional phrase-based system
- The improvement is more apparent with longer sentences, but can be observed with sentences of any length.





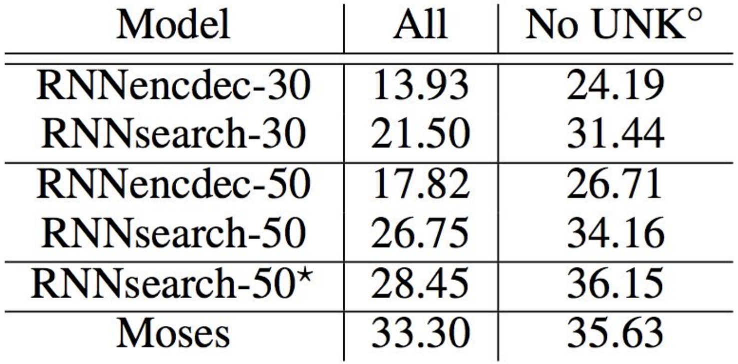
much better than RNN Encoder-Decoder

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses open source SMT	33.30	35.63

without unknown words comparable with the SMT system

Results

- RNNencdec (baseline) vs. RNNsearch (current)

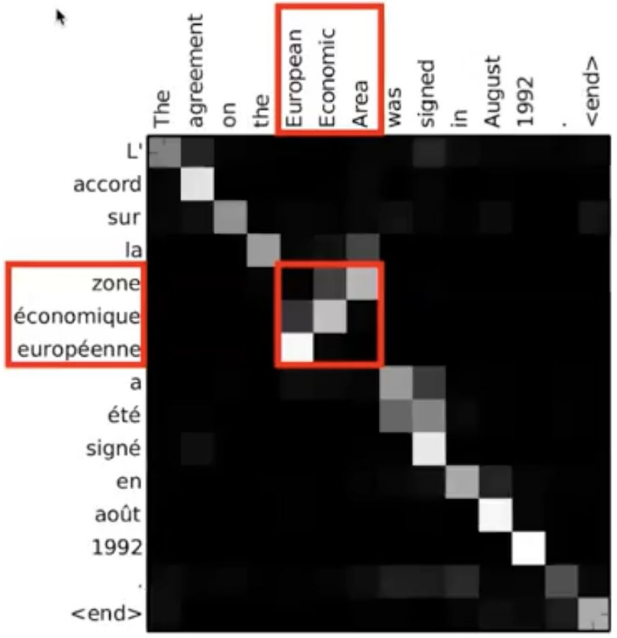


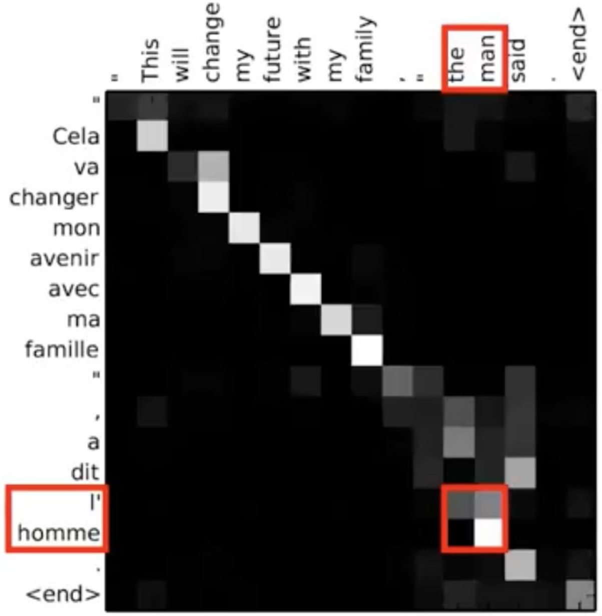
Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50⋆	28.45	36.15
Moses	33.30	35.63

Table 1: 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).

Qualitative Analysis - Alignment

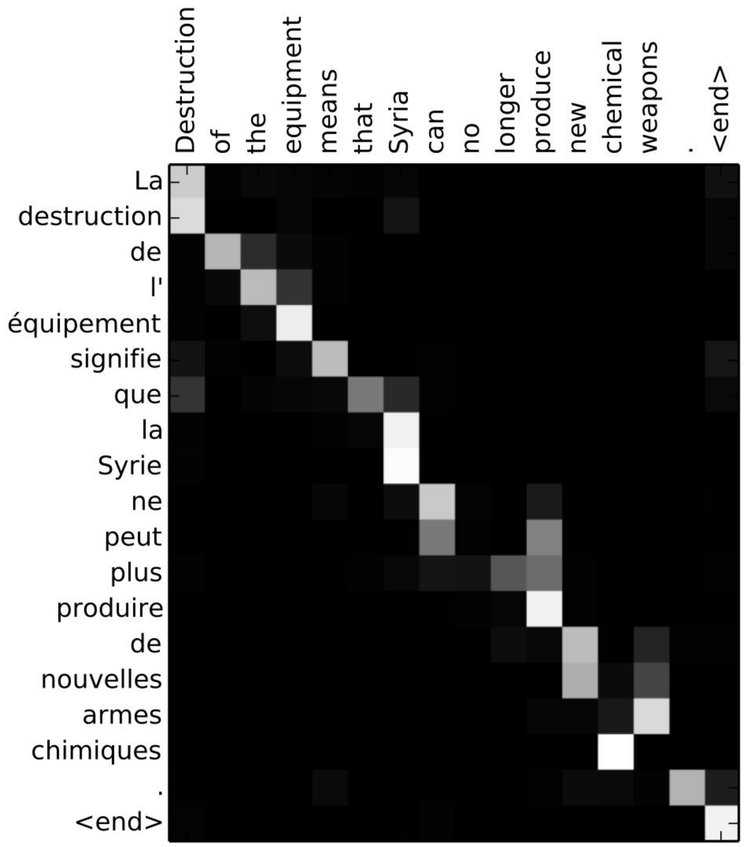
- Generates linguistically plausible (soft-)alignment between a source sentence and the corresponding target sentence.
- naturally deals with source and target phrases of different lengths

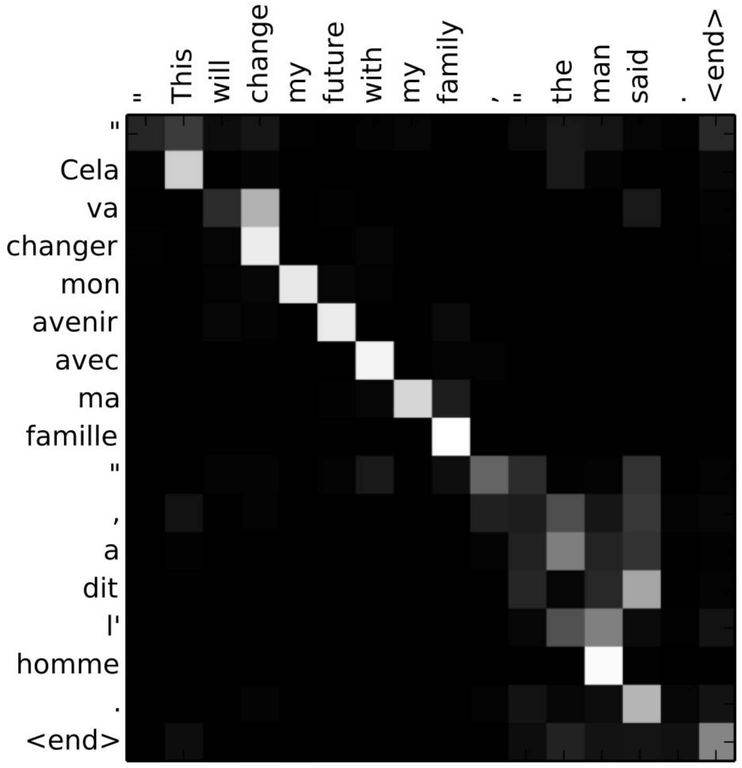




English-French: largely monotonic with a number of non-trivial, non-monotonic alignments

Qualitative Analysis - Alignment





Qualitative Analysis - Alignment

An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.

New Model
RNNsearch-50

Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.

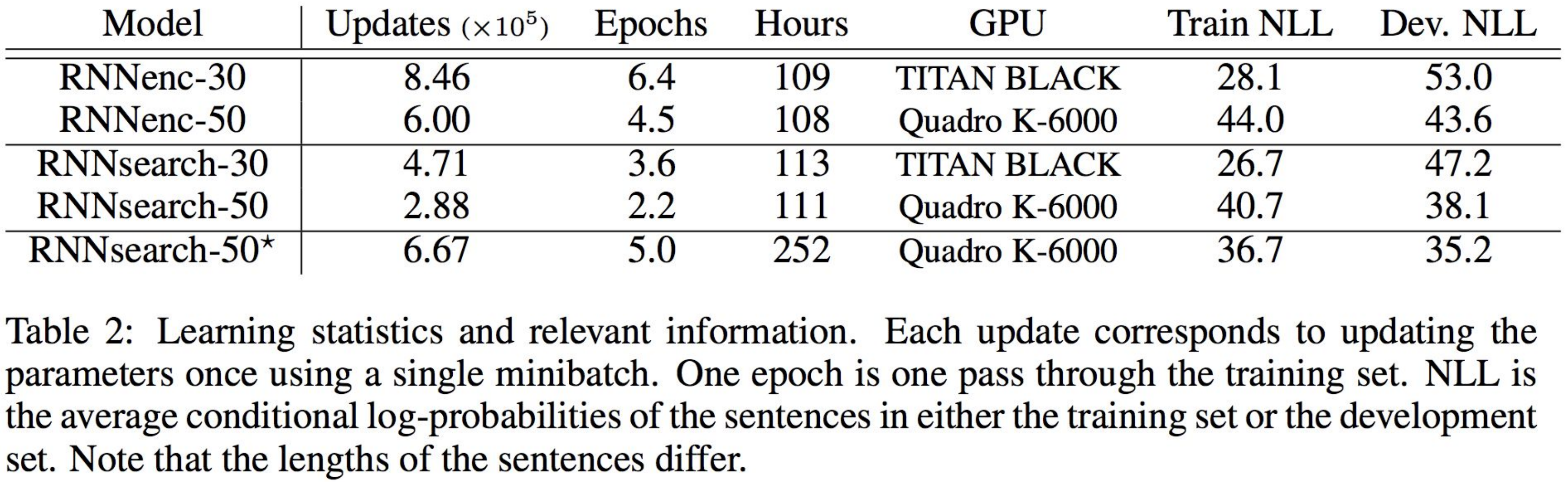
correct!

Encoder-Decoder
RNNencdec-50

.... d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.

[based on his state of health???]

Learning statistics



Model	Updates ($\times 10^5$)	Epochs	Hours	GPU	Train NLL	Dev. NLL
RNNenc-30	8.46	6.4	109	TITAN BLACK	28.1	53.0
RNNenc-50	6.00	4.5	108	Quadro K-6000	44.0	43.6
RNNsearch-30	4.71	3.6	113	TITAN BLACK	26.7	47.2
RNNsearch-50	2.88	2.2	111	Quadro K-6000	40.7	38.1
RNNsearch-50*	6.67	5.0	252	Quadro K-6000	36.7	35.2

Table 2: Learning statistics and relevant information. Each update corresponds to updating the parameters once using a single minibatch. One epoch is one pass through the training set. NLL is the average conditional log-probabilities of the sentences in either the training set or the development set. Note that the lengths of the sentences differ.

Related Work

Our alignment model is an *attention mechanism.*

- First differentiable attention model for handwriting synthesis: (Graves et al. 2013)
    - monotonic alignment only
    - predicts shifts instead of selecting location
- Non-differentiable attention mechanism for image classification: (Mnih et al. 2014)

Later Work

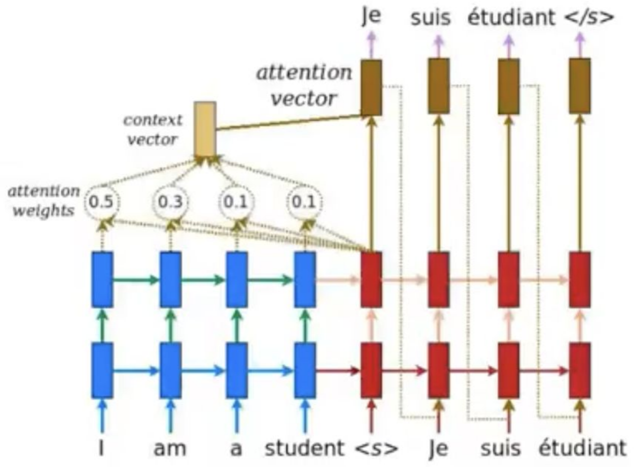
Effective Approaches to Attention-based Neural Machine Translation (Luong, 2015)

$$\alpha_{ts} = \frac{\exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \qquad \text{[Attention weights]} \qquad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \qquad \text{[Context vector]} \qquad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \qquad \text{[Attention vector]} \qquad (3)$$

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases} \qquad (4)$$



Summary

- Novel approach to neural machine translation
    - No fixed size representation
    - Linguistically plausible alignments
- Applicable to many other structured input/output problems
- Some design choices may be due to practical consideration
    - Later works do more analysis on the tradeoffs

Discussions

- How much improvement comes from attention? How much comes from bidirectional-ness? How much from gated RNN's?
  - Luong et al. shows that unidirectional LSTMs could perform just as well
- What about other score (energy) functions?
  - Later works show that results can vary based on types of attention (local vs. global, etc)
- Why is softmax needed? Is it always desirable?

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

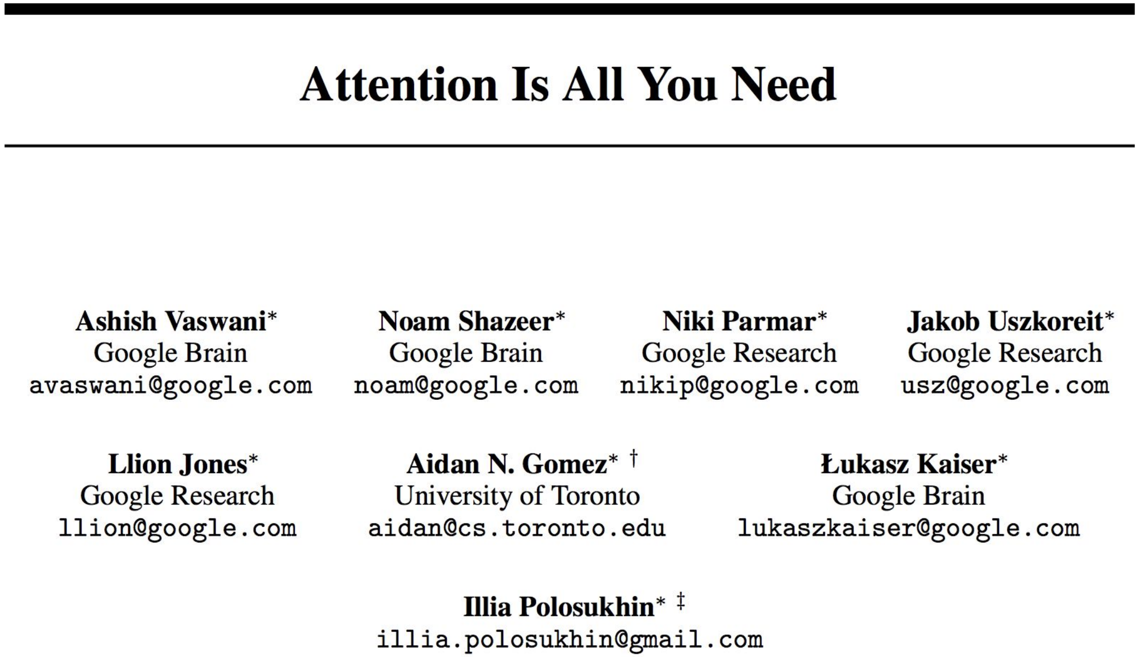
Luong et al. 2015.

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

$$e_{ij} = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right)$$

Next Monday (Oct 22)

Joseph Palermo from Dessa will be presenting



Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

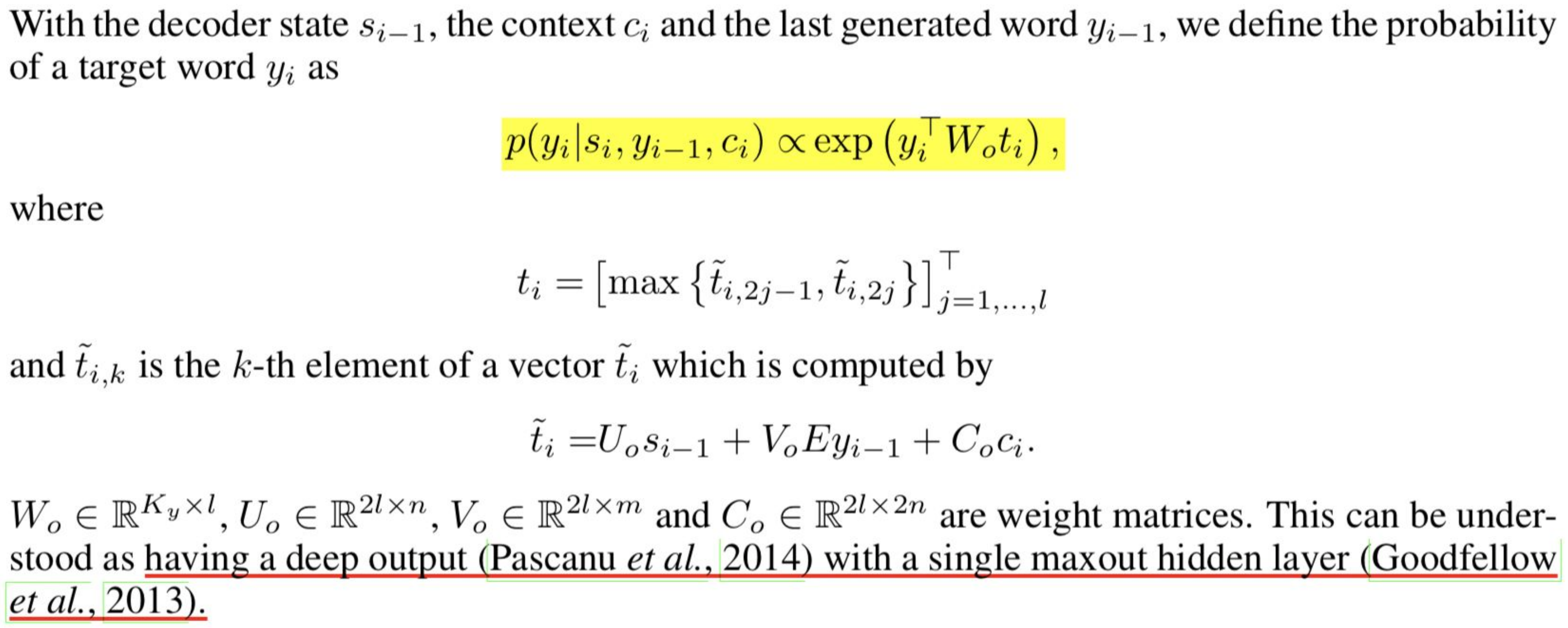
Illia Polosukhin* ‡
illia.polosukhin@gmail.com

References & Links

- Luong et al. 2015. Effective Approaches to Attention-based Neural Machine Translation
- Sutskever et al. 2014. Sequence to Sequence Learning with Neural Networks
- Cho et al. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches
- Neural Machine Translation and Sequence-to-Sequence Models: A Tutorial
- Attention and Memory in Deep Learning and NLP (blog post)
- Fandong Meng's report
- CMU Neural Nets for NLP 2017: Attention (Video & slides)
- YouTube Video by ChangWook Jun
- Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout net- works. In Proceedings of The 30th International Conference on Machine Learning, pages 1319– 1327.
- Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2014). How to construct deep recurrent neural networks. In Proceedings of the Second International Conference on Learning Representations (ICLR 2014).

Backup Slides

Congratulations, you have discovered the Hidden Portal!



With the decoder state s_{i-1}, the context c_i and the last generated word y_{i-1}, we define the probability of a target word y_i as

$$p(y_i | s_i, y_{i-1}, c_i) \propto \exp\left(y_i^\top W_o t_i\right),$$

where

$$t_i = \left[\max\left\{\tilde{t}_{i,2j-1}, \tilde{t}_{i,2j}\right\}\right]_{j=1,\ldots,l}^\top$$

and $\tilde{t}_{i,k}$ is the k-th element of a vector $\tilde{t}_i$ which is computed by

$$\tilde{t}_i = U_o s_{i-1} + V_o E y_{i-1} + C_o c_i.$$

$W_o \in \mathbb{R}^{K_y \times l}$, $U_o \in \mathbb{R}^{2l \times n}$, $V_o \in \mathbb{R}^{2l \times m}$ and $C_o \in \mathbb{R}^{2l \times 2n}$ are weight matrices. This can be understood as having a deep output (Pascanu *et al.*, 2014) with a single maxout hidden layer (Goodfellow *et al.*, 2013).

```
def encode(self, cell_fw, cell_bw):
    enc_outputs, (output_state_fw, output_state_bw) = tf.nn.bidirectional_dynamic_rnn(
        cell_fw,
        cell_bw,
        self.encoder_inputs_emb,
        dtype=tf.float32,
        sequence_length=self.source_length,
        time_major=True
    )
    enc_state = tf.concat([output_state_fw, output_state_bw], axis=1)
    enc_outputs = tf.concat(enc_outputs, axis=2)
    enc_outputs = tf.reshape(enc_outputs, [-1, self.emb_dim * 2])
    enc_outputs = tf.split(enc_outputs, self.num_steps, 0)
    return enc_outputs, enc_state
```

https://github.com/pemywei/attention-nmt/blob/master/seq2seq.py

```python
def decode(self, cell, init_state, enc_outputs, loop_function=None):
    outputs = []
    prev = None
    state = init_state
    for i, inp in enumerate(self.decoder_inputs_emb):

        if loop_function is not None and prev is not None:
            with tf.variable_scope("loop_function", reuse=True):
                inp = loop_function(prev, i)
        if i > 0:
            tf.get_variable_scope().reuse_variables()
        c_i = self.attention(state, enc_outputs)
        inp = tf.concat([inp, c_i], axis=1)
        output, state = cell(inp, state)
        # print output.eval()
        outputs.append(output)
        if loop_function is not None:
            prev = output
    return outputs
```

```python
def attention(self, prev_state, enc_outputs):
    """
    Attention model for Neural Machine Translation
    :param prev_state: the decoder hidden state at time i-1
    :param enc_outputs: the encoder outputs, a length 'T' list.
    """
    e_i = []
    c_i = []
    for output in enc_outputs:
        atten_hidden = tf.tanh(tf.add(tf.matmul(prev_state, self.attention_W), tf.matmul(output, self.attention_U)))
        e_i_j = tf.matmul(atten_hidden, self.attention_V)
        e_i.append(e_i_j)
    e_i = tf.concat(e_i, axis=1)
    # e_i = tf.exp(e_i)
    alpha_i = tf.nn.softmax(e_i)
    alpha_i = tf.split(alpha_i, self.num_steps, 1)
    for alpha_i_j, output in zip(alpha_i, enc_outputs):
        c_i_j = tf.multiply(alpha_i_j, output)
        c_i.append(c_i_j)
    c_i = tf.reshape(tf.concat(c_i, axis=1), [-1, self.num_steps, self.hidden_dim * 2])
    c_i = tf.reduce_sum(c_i, 1)
    return c_i
```

```python
# encode and decode
enc_outputs, enc_state = self.encode(self.enc_lstm_cell_fw, self.enc_lstm_cell_bw)
if is_training:
    self.dec_outputs = self.decode(self.dec_lstm_cell, enc_state, enc_outputs)
else:
    self.dec_outputs = self.decode(self.dec_lstm_cell, enc_state, enc_outputs, self.loop_function)
# softmax
self.outputs = tf.reshape(tf.concat(self.dec_outputs, axis=1), [-1, self.hidden_dim * 2])
self.logits = tf.add(tf.matmul(self.outputs, self.softmax_w), self.softmax_b)
self.prediction = tf.nn.softmax(self.logits)

self.y_output = tf.reshape(self.y_outputs, [-1])
self.y_output = tf.one_hot(self.y_output, depth=self.target_vocab_size, on_value=1.0, off_value=0.0)

self.target_weight = tf.reshape(self.target_weights, [-1])

cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=self.logits, labels=self.y_output)
self.cross_entropy_loss = tf.reduce_mean(tf.multiply(self.target_weight, cross_entropy))

# Gradients and SGD update operation for training the model.
params = tf.trainable_variables()
self.optimizer = tf.train.GradientDescentOptimizer(self.learning_rate)

gradients = tf.gradients(self.cross_entropy_loss, params)
clipped_gradients, _ = tf.clip_by_global_norm(gradients, self.max_gradient_norm)
self.updates = self.optimizer.apply_gradients(zip(clipped_gradients, params), global_step=self.global_step)
```

Automatic Evaluation: Bleu Score

N-Gram precision

$$p_n = \frac{\sum_{n\text{-gram} \in hyp} count_{clip}(n\text{-gram})}{\sum_{n\text{-gram} \in hyp} count(n\text{-gram})}$$

Bounded above by highest count of n-gram in any reference sentence

brevity penalty

$$\mathbf{B} = \begin{cases} e^{(1-|ref|/|hyp|)} & \text{if } |ref| > |hyp| \\ 1 & \text{otherwise} \end{cases}$$

Bleu score: brevity penalty, geometric mean of N-Gram precisions

$$\textbf{Bleu=}\quad B \cdot \exp\left[\frac{1}{N}\sum_{n=1}^{N} p_n\right]$$