# Natural Language Processing
## CSE 325/425

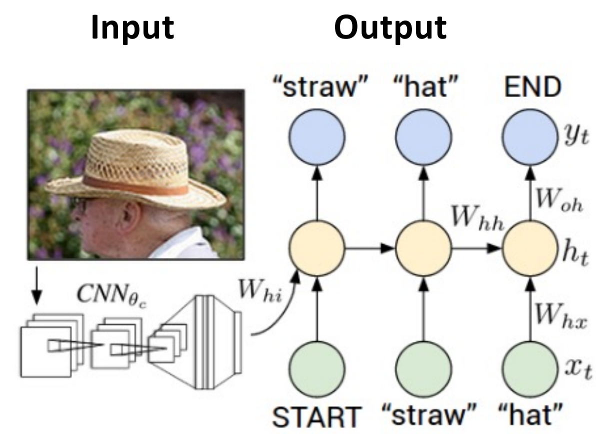
Sihong Xie

Lecture 26:
- Seq2seq model.
- Extensions.

# Applications of RNN

RNN is a language model and can be used to

- Evaluate sequences (sentence).
- Generate new sequences.
- Applications:
    - Machine translation (MT).
    - Image / video captionnig.
    - Question answering.
    - Dialogue bots.

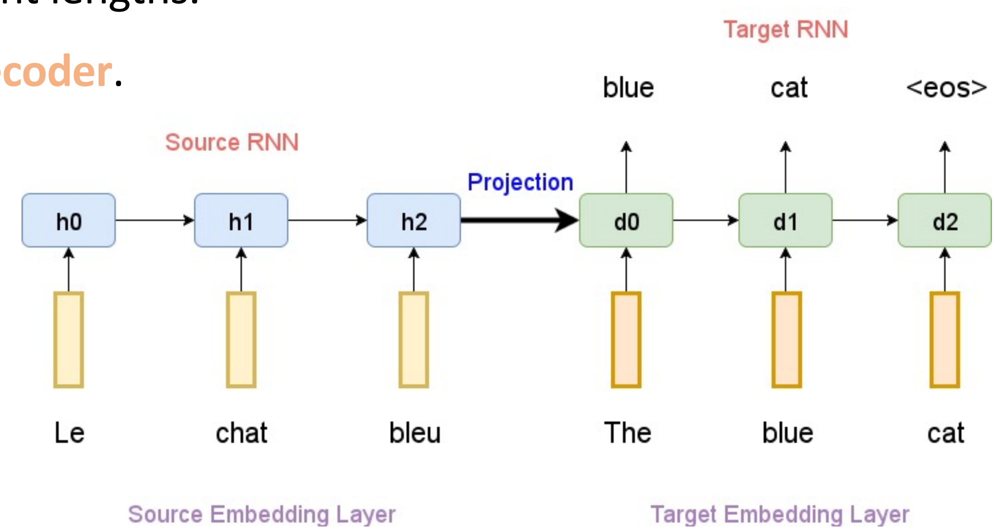



**Output:** *An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.*

**Input:** *Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.*

# Seq2seq (encoder-decoder) Model

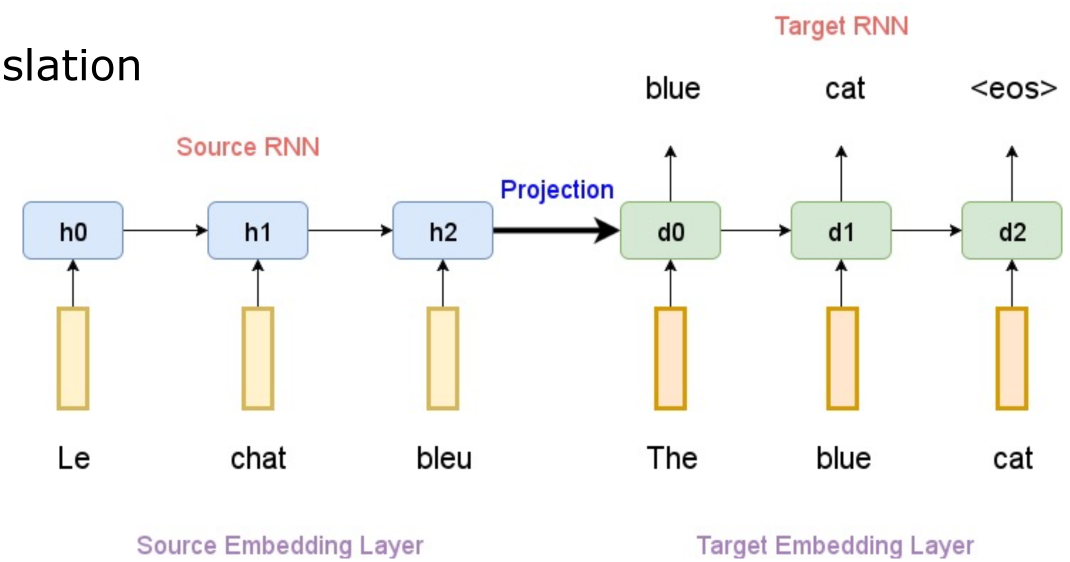
- Two RNNs to encode source sentence and predict a translation
  - Encoder: mapping from ***x*** (the input) to ***h*** (hidden units).
  - Decoder: mapping from ***h*** to ***y*** (the output).
  - Input and output can have different lengths.
  - Decouple the encoder and the decoder.




Sequence to Sequence Learning with Neural Networks, NIPS, 2015 (using LSTM)
Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2015 (using RNN)

# Seq2seq (encoder-decoder) Model

Machine translation




**Encoder** is a regular RNN (source RNN):

- $$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

**Decoder** is another regular RNN (target RNN):

- $$\boldsymbol{d}^{(t)} = g(\boldsymbol{d}^{(t-1)}, \hat{\boldsymbol{y}}^{(t)}; \boldsymbol{\theta})$$

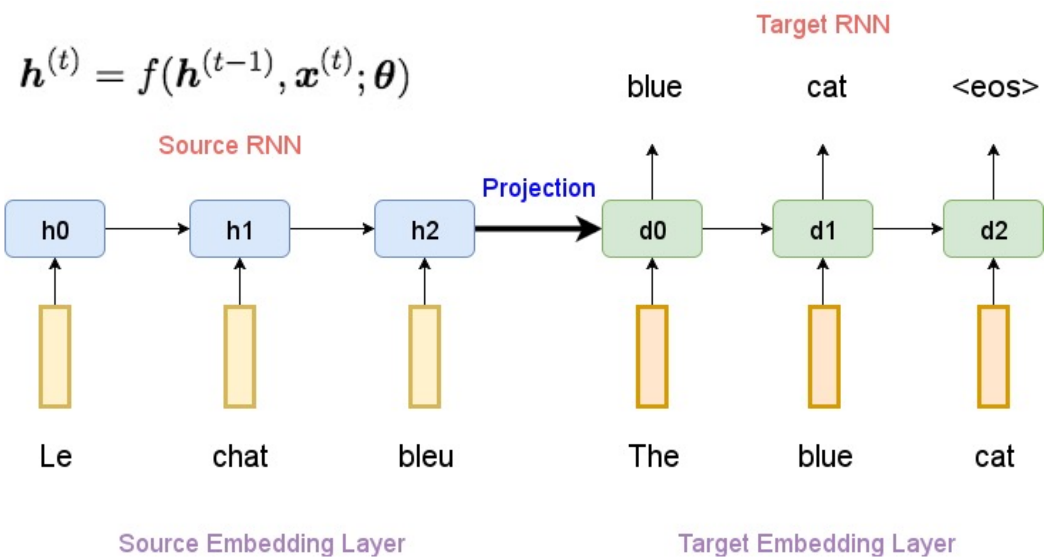
source: https://recordnotfound.com/Seq2Seq-PyTorch-MaximumEntropy-149015

# Seq2seq training and prediction

Given a (input, output) pair $([x_1, \dots, y_T], [y_1, \dots, y_{T'}])$

MLE: $$p(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots, y_{t-1})$$

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$



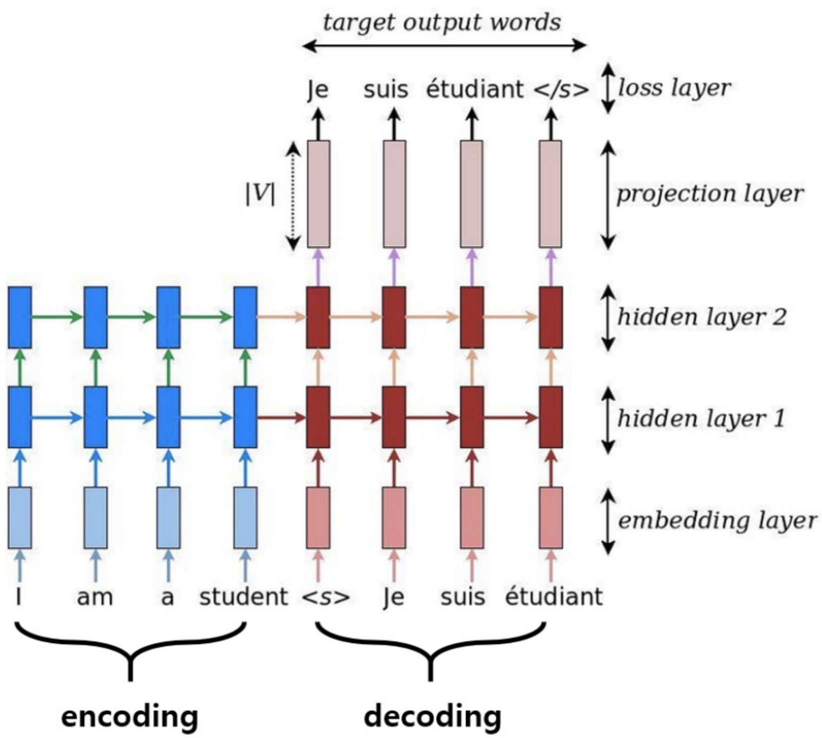

$$p(y_t | v, y_1, \dots, y_{t-1}) = \mathrm{softmax}(o^{(t)})$$

$$o^{(t)} = \boldsymbol{V}\boldsymbol{d}^{(t)}$$

$$\boldsymbol{d}^{(t)} = g(\boldsymbol{d}^{(t-1)}, \hat{\boldsymbol{y}}^{(t)}; \boldsymbol{\theta})$$

- During prediction, previous word in the translation is predicted rather than provided:
- This is call “Teacher forcing”

# Deeper seq2seq model




- Deeper RNNs to capture more non-linearity (**BERT** can go up to 24 layers).
- Lower level RNN captures lower level and short-term dependencies.
- Higher level RNN captures long-range dependencies.
- **Encoder** and **decoder** can have different depths.

Image courtesy of: https://towardsdatascience.com/seq2seq-model-in-tensorflow-ec0c557e560f

# seq2seq (encoder-decoder) model

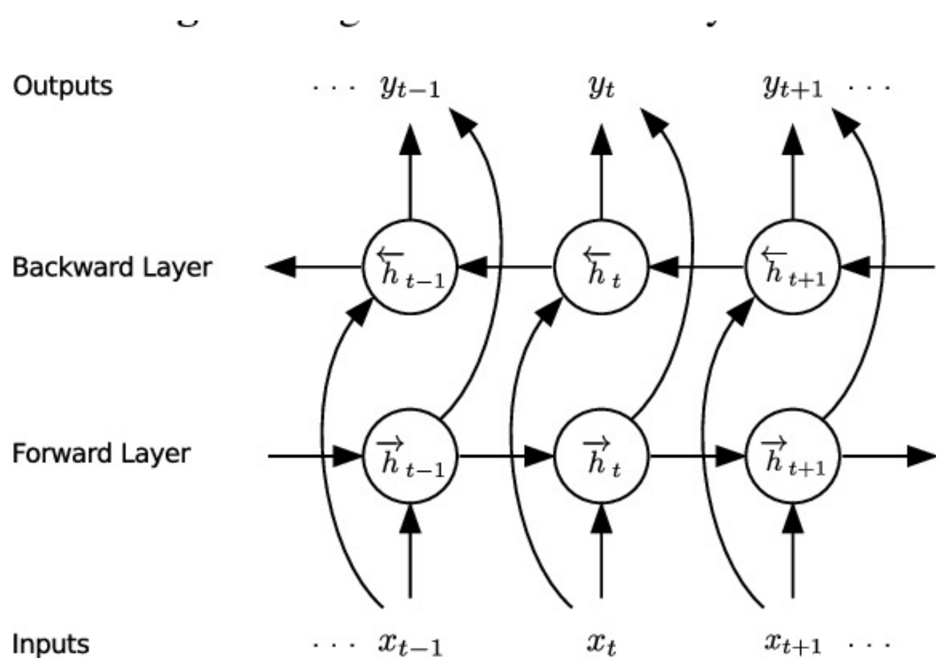



Image courtesy of : "Hybrid speech recognition with Deep Bidirectional LSTM

Languages are inherently bidirectional

***" the movie was terribly exciting ! "***

*negative*

$$\overrightarrow{\boldsymbol{h}}_t = \overrightarrow{f}(\overrightarrow{\boldsymbol{h}}_{t-1}, \boldsymbol{x}_t)$$

*positive*

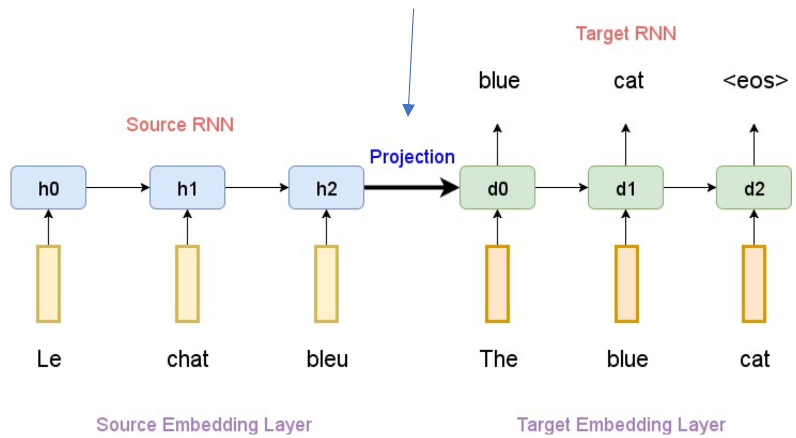
$$\overleftarrow{\boldsymbol{h}}_t = \overleftarrow{f}(\overleftarrow{\boldsymbol{h}}_{t+1}, \boldsymbol{x}_t)$$

Concatenation to combine both contexts:

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t]$$

# Attention models

This **fixed-length** vector has to provide the information that all target positions need, regardless of the length of the source sequence. Asking for too much!




Neural machine translation by jointly learning to align and translate, ICLR, 2015

RNN tends to have **sequential recency**

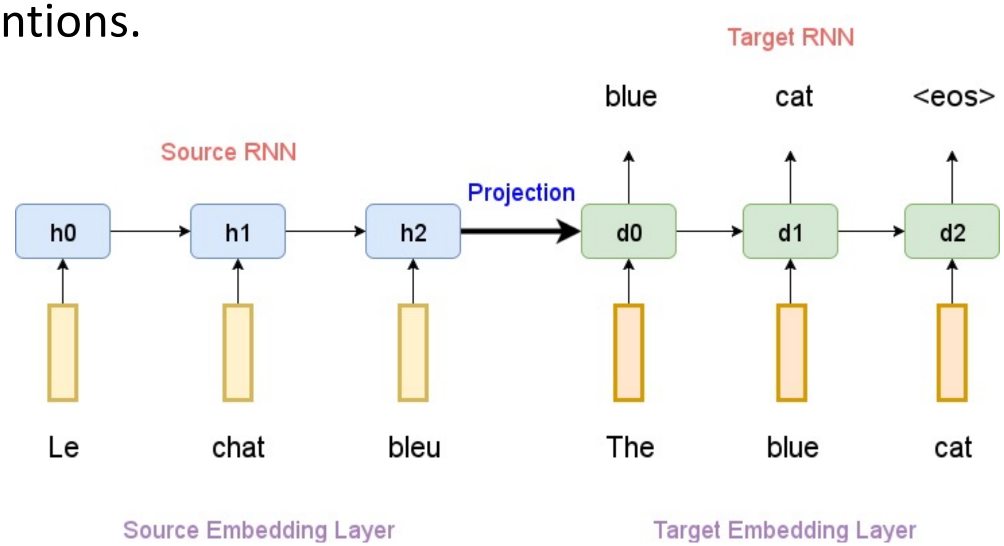
- *The writer of the books are* **(Incorrect)**
- *The writer of the books is* **(Correct)**

LSTM may address this, but there are more direct way

- attention models find which source word is important to predict a word in the target sequence.
- do not depend on the fixed-length vector at the end.
- More contextual information can be provided.

# Attention models

- What source positions are useful when generating “*cat*”?
- Learn to focus rather than human specification.
- Focus <=> Attentions.

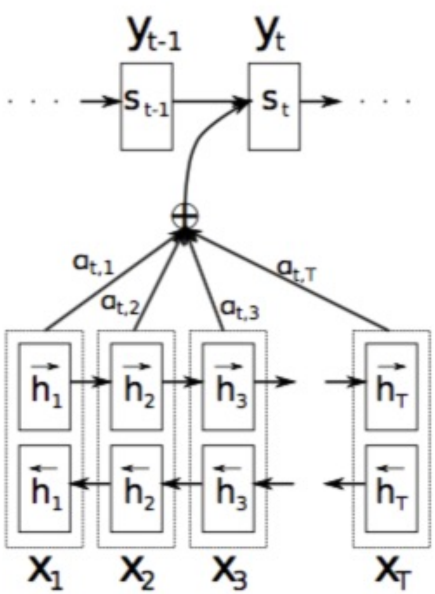



Neural machine translation by jointly learning to align and translate, ICLR, 2015

# Attention models

Decoder

Encoder
(bidirectional RNN)




Neural machine translation by jointly learning to align and translate, ICLR, 2015

Hidden state of the decoder:

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

Context $c_t$ is the weighted sum:

$$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j$$

The weights $\alpha_{t,j}$ is the attention paid to the *j*-th source word

$$\alpha_{t,j} = \frac{\exp(\mathrm{Score}(t,j))}{\sum_{k=1}^{T} \exp(\mathrm{Score}(t,k))}$$

Example score functions

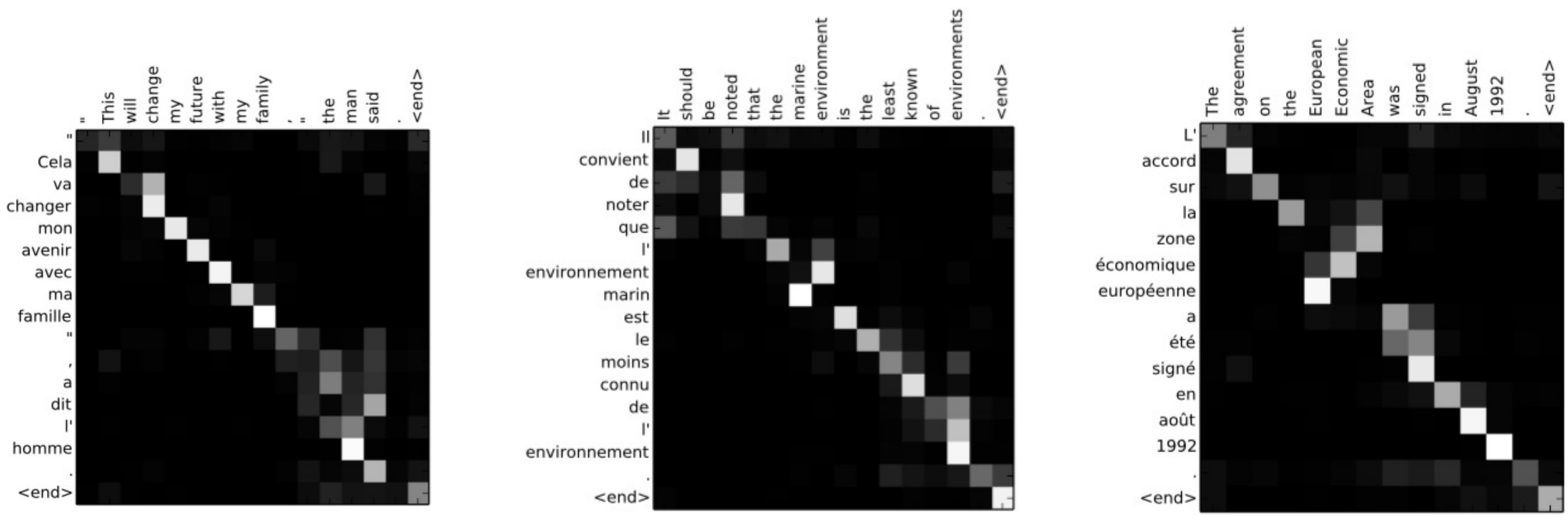
$$\mathrm{Score}(t,k) = \langle s_{t-1}, h_t \rangle$$

$$\mathrm{Score}(t,k) = s_{t-1}^{\top} W_a h_t$$

# Attention models

How attention helps machine translation?

- One-to-many and reverse alignments can easily be modeled.

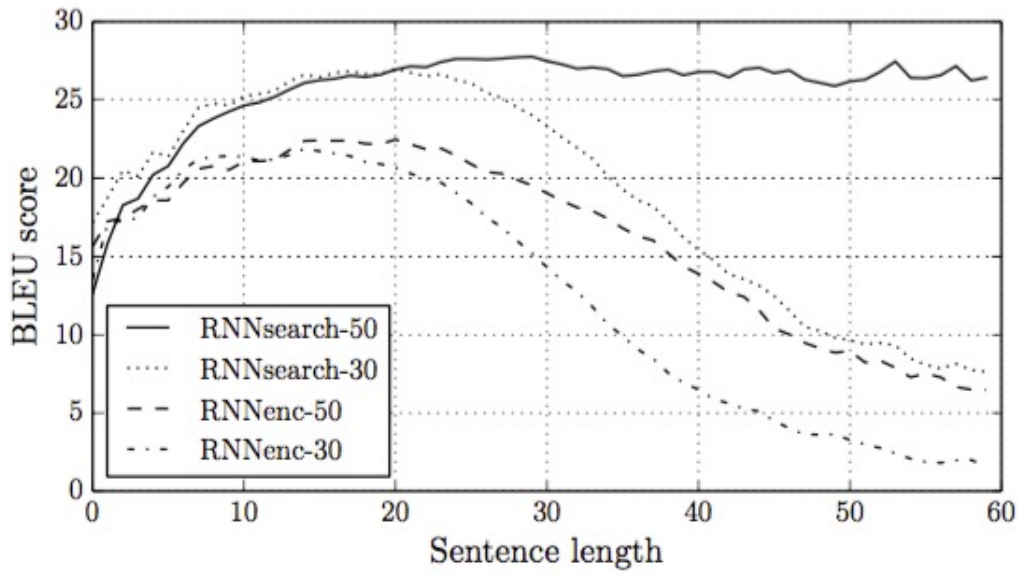



Neural machine translation by jointly learning to align and translate, ICLR, 2015

# Attention models

How attention helps machine translation?

- Longer sentences are harder without attention




Neural machine translation by jointly learning to align and translate, ICLR, 2015