Sequence to Sequence Models for Machine Translation (2)

CMSC 723 / LING 723 / INST 725

Marine Carpuat

Slides & figure credits: Graham Neubig

Introduction to Neural Machine Translation

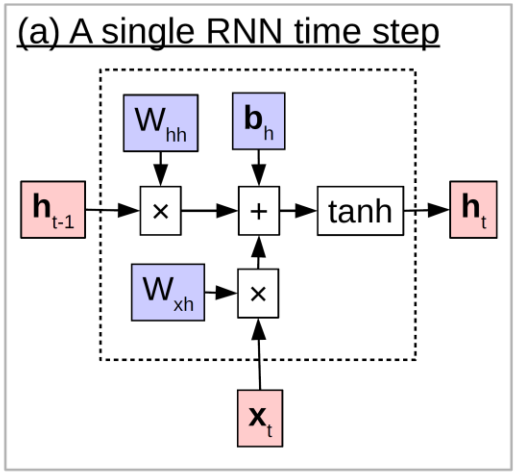
- Neural language models review

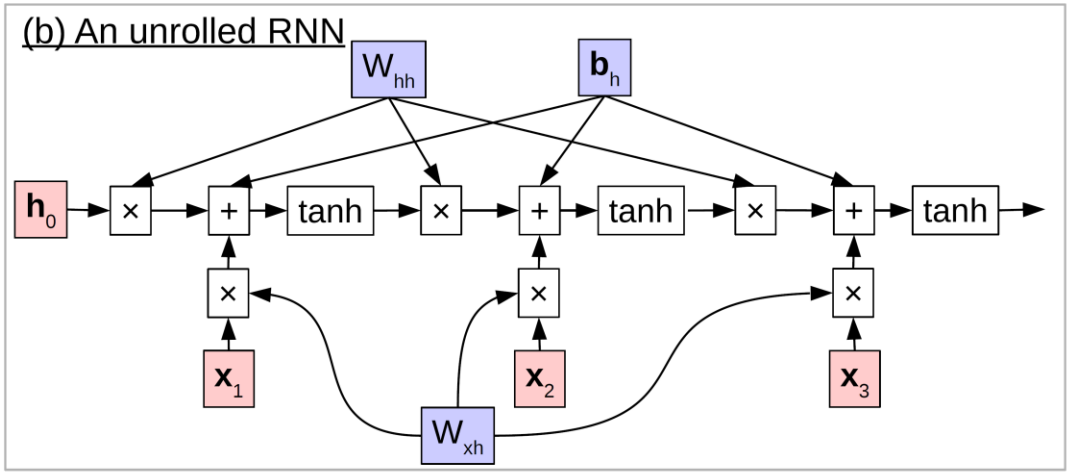
- Sequence to sequence models for MT
  - Encoder-Decoder
  - Sampling and search (greedy vs beam search)
  - **Practical tricks**

- **Sequence to sequence models for other NLP tasks**

- **Attention mechanism**

A recurrent language model



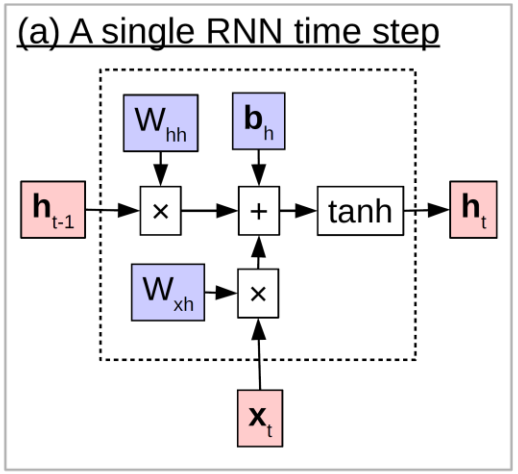


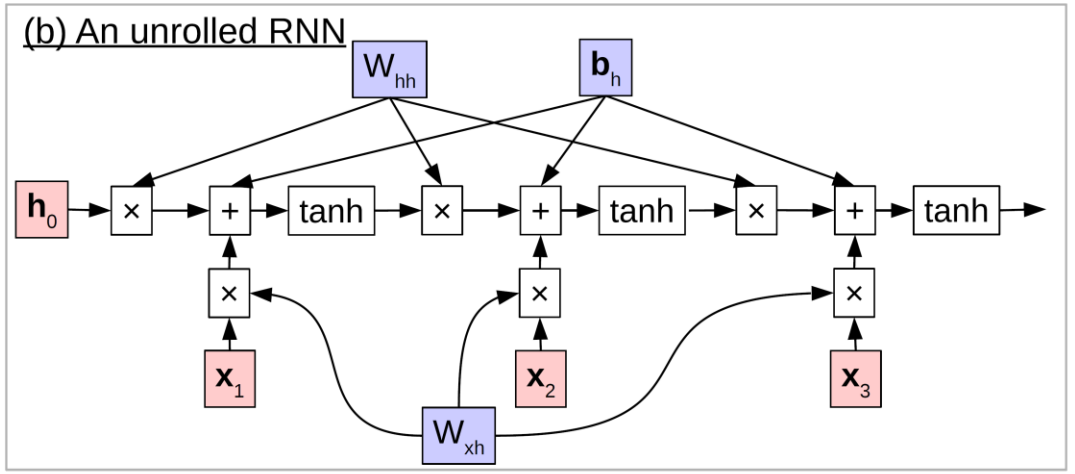
$$\boldsymbol{m}_t = M_{\cdot,e_{t-1}}$$

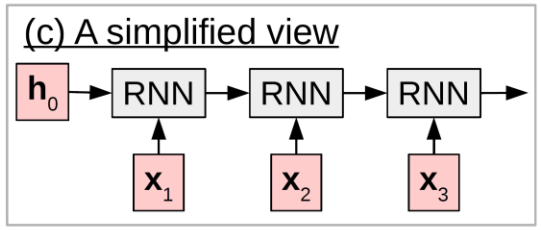
$$\boldsymbol{h}_t = \begin{cases} \tanh(W_{mh}\boldsymbol{m}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) & t \geq 1, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{p}_t = \text{softmax}(W_{hs}\boldsymbol{h}_t + b_s).$$

A recurrent language model





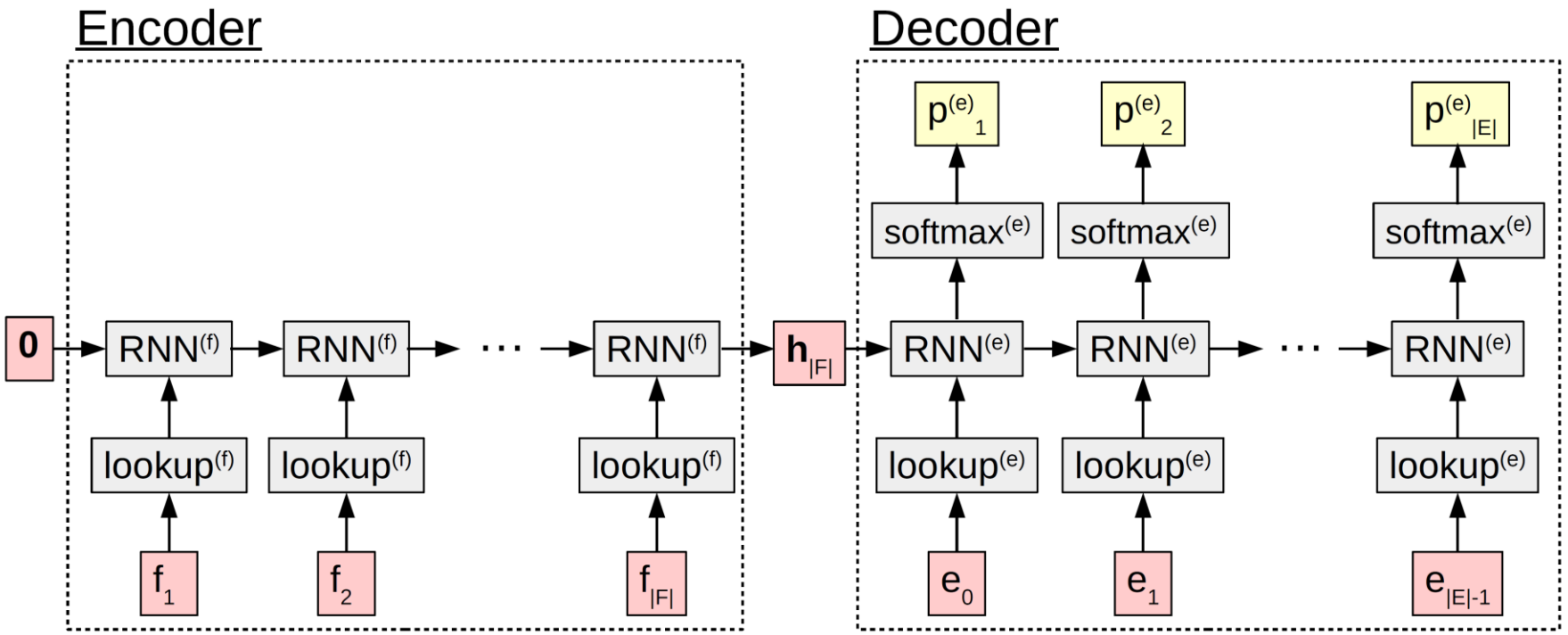


$$\boldsymbol{m}_t = M_{\cdot,e_{t-1}}$$

$$\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{m}_t, \boldsymbol{h}_{t-1})$$

$$\boldsymbol{p}_t = \mathrm{softmax}(W_{hs}\boldsymbol{h}_t + b_s).$$

Encoder-decoder model



Encoder-decoder model

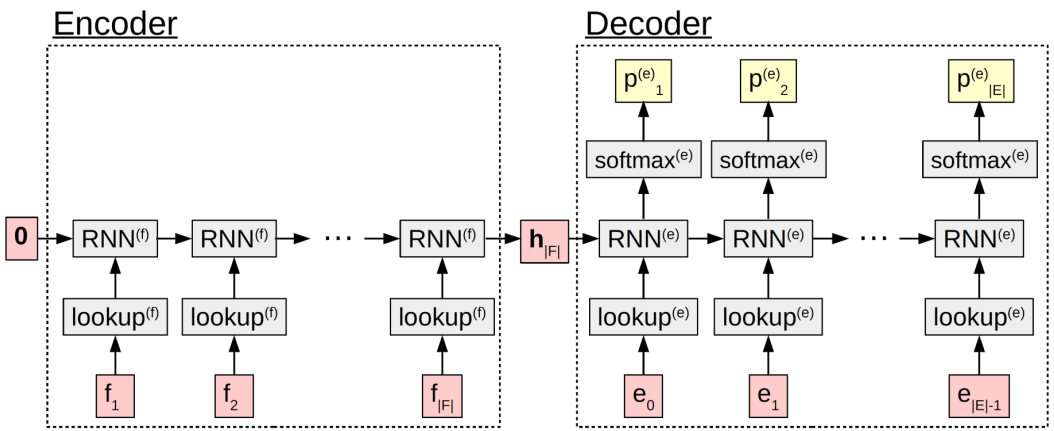
$$\boldsymbol{m}_t^{(f)} = M_{\cdot,f_t}^{(f)}$$

$$\boldsymbol{h}_t^{(f)} = \begin{cases} \mathrm{RNN}^{(f)}(\boldsymbol{m}_t^{(f)}, \boldsymbol{h}_{t-1}^{(f)}) & t \geq 1, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{m}_t^{(e)} = M_{\cdot,e_{t-1}}^{(e)}$$

$$\boldsymbol{h}_t^{(e)} = \begin{cases} \mathrm{RNN}^{(e)}(\boldsymbol{m}_t^{(e)}, \boldsymbol{h}_{t-1}^{(e)}) & t \geq 1, \\ \boldsymbol{h}_{|F|}^{(f)} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{p}_t^{(e)} = \mathrm{softmax}(W_{hs}\boldsymbol{h}_t^{(e)} + b_s)$$



Generating Output

- We have a model P(E|F), how can we generate translations?
- 2 methods
  - **Sampling**: generate a random sentence according to probability distribution
  - **Argmax**: generate sentence with highest probability

Training

- Same as for RNN language modeling

- Loss function
  - Negative log-likelihood of training data
  - Total loss for one example (sentence) = sum of loss at each time step (word)

- BackPropagation Through Time (BPTT)
  - Gradient of loss at time step t is propagated through the network all the way back to 1st time step

Note that training loss differs from evaluation metric (BLEU)

N-gram overlap between machine translation output and reference translation

Compute precision for n-grams of size 1 to 4

Add brevity penalty (for too short translations)

$$\text{BLEU} = \min\left(1, \frac{\textit{output-length}}{\textit{reference-length}}\right) \left(\prod_{i=1}^{4} \textit{precision}_i\right)^{\frac{1}{4}}$$

Typically computed over the entire corpus, not single sentences

Other encoder structures: Bidirectional encoder

$$\overrightarrow{\boldsymbol{h}}_t^{(f)} = \begin{cases} \overrightarrow{\mathrm{RNN}}^{(f)}(\boldsymbol{m}_t^{(f)}, \overrightarrow{\boldsymbol{h}}_{t+-}^{(f)}) & t \geq 1, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

$$\overleftarrow{\boldsymbol{h}}_t^{(f)} = \begin{cases} \overleftarrow{\mathrm{RNN}}^{(f)}(\boldsymbol{m}_t^{(f)}, \overleftarrow{\boldsymbol{h}}_{t+1}^{(f)}) & t \leq |F|, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Motivation:
- Help bootstrap learning
- By shortening length of dependencies

$$\boldsymbol{h}_0^{(e)} = \tanh(W_{\overrightarrow{f}\,e}\overrightarrow{\boldsymbol{h}}_{|F|} + W_{\overleftarrow{f}\,e}\overleftarrow{\boldsymbol{h}}_1 + \boldsymbol{b}_e)$$

Motivation:
- Take 2 hidden vectors from source encoder
- Combine them into a vector of size required by decoder

A few more tricks: addressing length bias

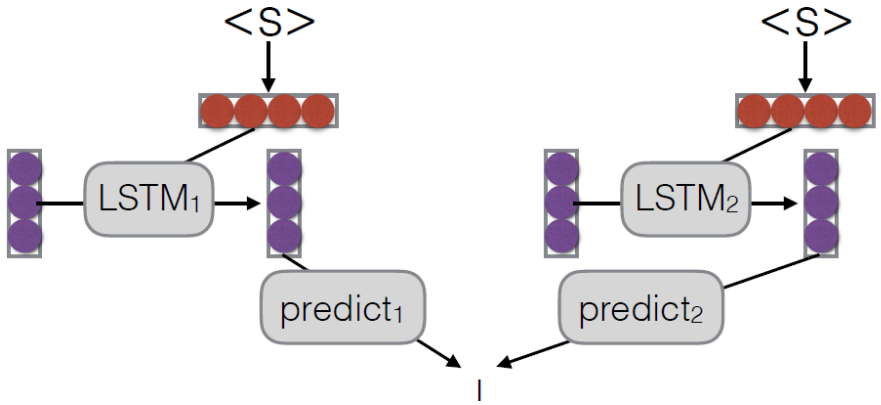
- Default models tend to generate short sentences
- Solutions:
  - Prior probability on sentence length

$$\hat{E} = \underset{E}{\operatorname{argmax}} \ \log P(|E| \mid |F|) + \log P(E \mid F).$$

  - Normalize by sentence length

$$\hat{E} = \underset{E}{\operatorname{argmax}} \ \log P(E \mid F)/|E|.$$

A few more tricks: ensembling



- Combine predictions from multiple models
- Methods
  - Linear or log-linear interpolation
  - Parameter averaging

Introduction to Neural Machine Translation

- Neural language models review

- Sequence to sequence models for MT
  - Encoder-Decoder
  - Sampling and search (greedy vs beam search)
  - **Practical tricks**

- **Sequence to sequence models for other NLP tasks**

- **Attention mechanism**

Beyond MT: Encoder-Decoder can be used as Conditioned Language Models to generate text Y according to some specification X

Input *X*	Output *Y* (**Text**)	Task
Structured Data	NL Description	NL Generation
English	Japanese	Translation
Document	Short Description	Summarization
Utterance	Response	Response Generation
Image	Text	Image Captioning
Speech	Transcript	Speech Recognition

Introduction to Neural Machine Translation

- Neural language models review

- Sequence to sequence models for MT
  - Encoder-Decoder
  - Sampling and search (greedy vs beam search)
  - **Practical tricks**

- **Sequence to sequence models for other NLP tasks**

- **Attention mechanism**

Problem with previous encoder-decoder model

- Long-distance dependencies remain a problem
- A single vector represents the entire source sentence
  - No matter its length
- Solution: attention mechanism
  - An example of incorporating inductive bias in model architecture

Attention model intuition

- Encode each word in source sentence into a vector
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”
- Use this combination when predicting next word

[Bahdanau et al. 2015]

Attention model
Source word representations

- We can use representations from bidirectional RNN encoder

$$\overrightarrow{\boldsymbol{h}}_j^{(f)} = \mathrm{RNN}(\mathrm{embed}(f_j), \overrightarrow{\boldsymbol{h}}_{j-1}^{(f)})$$
$$\overleftarrow{\boldsymbol{h}}_j^{(f)} = \mathrm{RNN}(\mathrm{embed}(f_j), \overleftarrow{\boldsymbol{h}}_{j+1}^{(f)}).$$

$$\boldsymbol{h}_j^{(f)} = [\overleftarrow{\boldsymbol{h}}_j^{(f)}; \overrightarrow{\boldsymbol{h}}_j^{(f)}].$$

- And concatenate them in a matrix

$$H^{(f)} = \mathrm{concat\_col}(\boldsymbol{h}_1^{(f)}, \ldots, \boldsymbol{h}_{|F|}^{(f)}).$$

Attention model
Create a source context vector

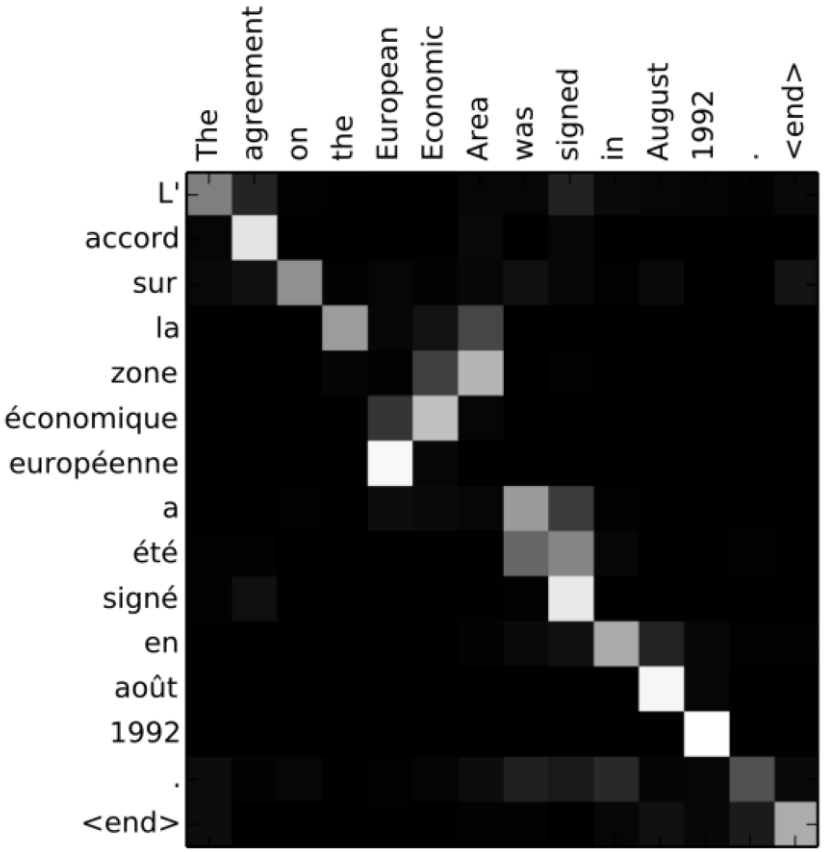
$$\boldsymbol{c}_t = H^{(f)}\boldsymbol{\alpha}_t.$$

Context vector

Attention vector

- Attention vector:
  - Entries between 0 and 1
  - Interpreted as weight given to each source word when generating output at time step t

Attention model
Illustrating attention weights



Attention model
How to calculate attention scores

$$\boldsymbol{h}_t^{(e)} = \text{enc}([\text{embed}(e_{t-1}); \boldsymbol{c}_{t-1}], \boldsymbol{h}_{t-1}^{(e)}).$$

$$a_{t,j} = \text{attn\_score}(\boldsymbol{h}_j^{(f)}, \boldsymbol{h}_t^{(e)}).$$

$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{a}_t).$$

$$\boldsymbol{p}_t^{(e)} = \text{softmax}(W_{hs}[\boldsymbol{h}_t^{(e)}; \boldsymbol{c}_t] + b_s).$$

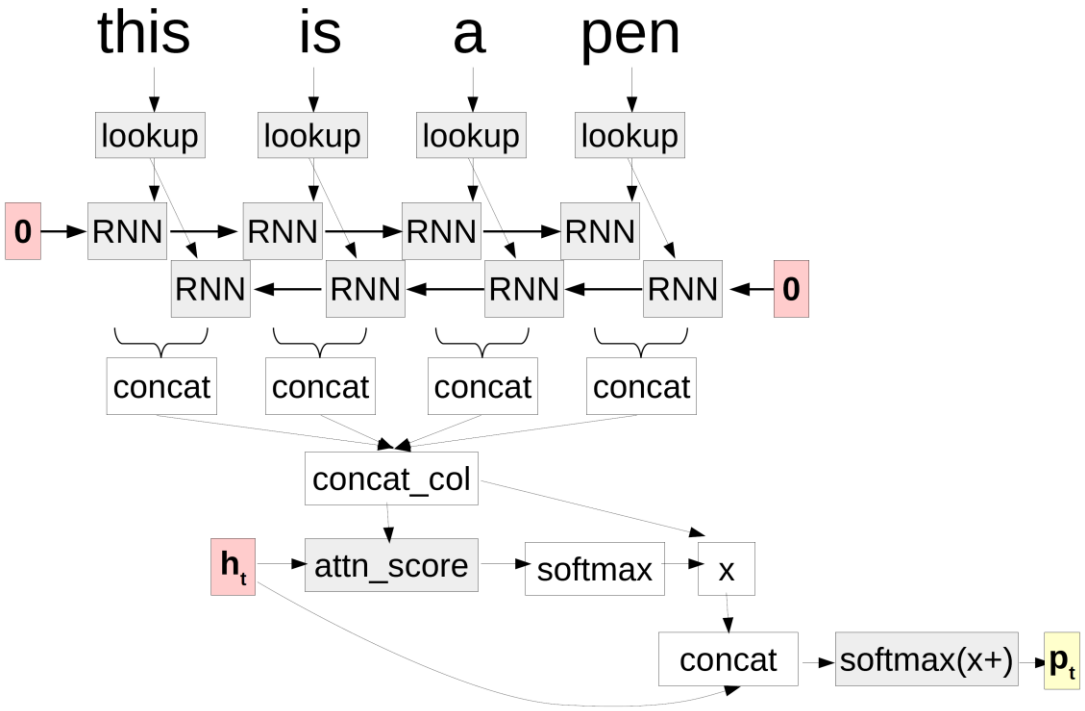


Figure 28: A computation graph for attention.

Attention model
Various ways of calculating attention score

- Dot product

$$\text{attn\_score}(\boldsymbol{h}_j^{(f)}, \boldsymbol{h}_t^{(e)}) := \boldsymbol{h}_j^{(f)\intercal} \boldsymbol{h}_t^{(e)}$$

- Bilinear function

$$\text{attn\_score}(\boldsymbol{h}_j^{(f)}, \boldsymbol{h}_t^{(e)}) := \boldsymbol{h}_j^{(f)\intercal} W_a \boldsymbol{h}_t^{(e)}.$$

- Multi-layer perceptron (original formulation in Bahdanau et al.)

$$\text{attn\_score}(\boldsymbol{h}_t^{(e)}, \boldsymbol{h}_j^{(f)}) := \boldsymbol{w}_{a2}^{\intercal} \tanh(W_{a1}[\boldsymbol{h}_t^{(e)}; \boldsymbol{h}_j^{(f)}])$$

Advantages of attention

- Helps illustrate/interpret translation decisions
- Can help insert translations for OOV
  - By copying or look up in external dictionary
- Can incorporate linguistically motivated priors in model

Attention extensions
An active area of research

- Attend to multiple sentences (Zoph et al. 2015)
- Attend to a sentence and an image (Huang et al. 2016)
- Incoprorate bias from alignment models

Introduction to Neural Machine Translation

- Neural language models review

- Sequence to sequence models for MT
  - Encoder-Decoder
  - Sampling and search (greedy vs beam search)
  - **Practical tricks**

- **Sequence to sequence models for other NLP tasks**

- **Attention mechanism**