9. Modern Recurrent Neural Networks

Junggwon Shin, shinjunggwon@gmail.com

Summary for Dive Into Deep Learning, https://d2l.ai/chapter_preface/index.html

9.5 ~ 9.8 Machine Translation and Beam Search

Machine translation

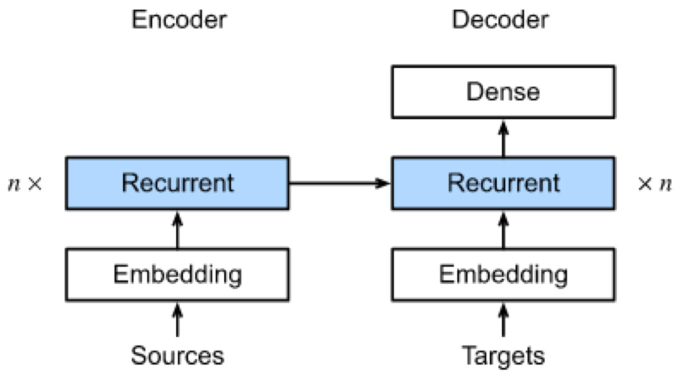
- The automatic translation of a sequence from one language to another
- Machine translation datasets are composed of pairs of text sequences that are in the source language and the target language
- http://www.manythings.org/anki/

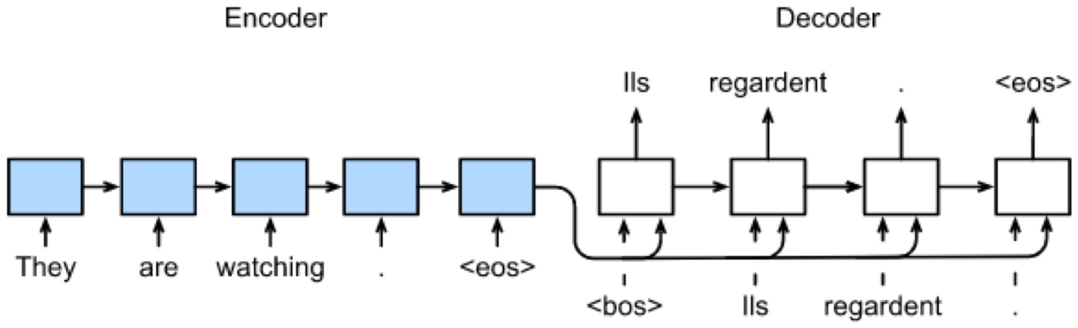
Encoder-Decoder Architecture

- Problem: input and output are both variable-length sequences.
- To handle this type of inputs and outputs, we can design an architecture with two major components.
- Encoder: it takes a variable-length sequence as the input and transforms it into a state with a fixed shape.
- Decoder: it maps the encoded state of a fixed shape to a variable-length sequence.

Sequence to Sequence Learning

- we will use two RNNs to design the encoder and the decoder of this architecture and apply it to *sequence to sequence* learning for machine translation

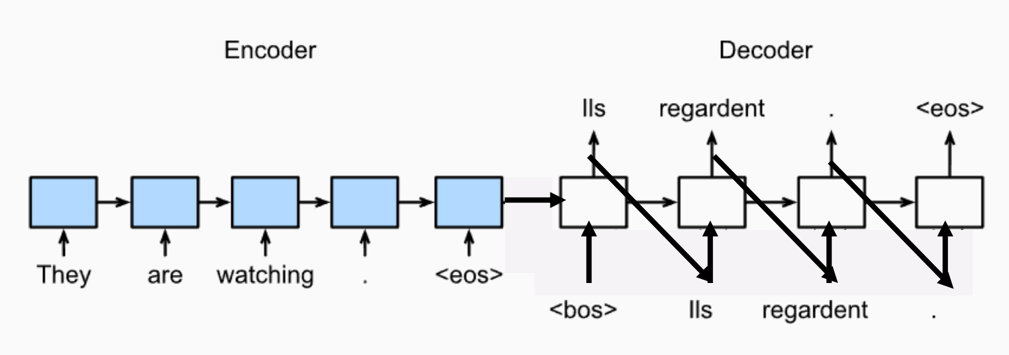




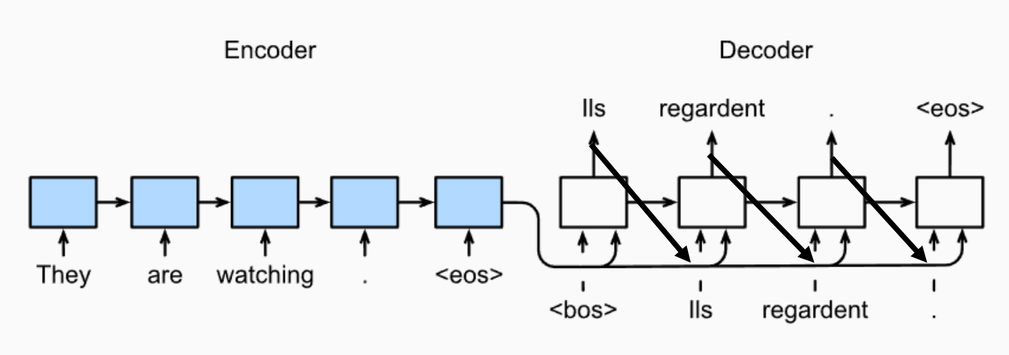
Sequence to Sequence Learning (cont.)

- RNN *encoder* can take a variable-length sequence as the input and transforms it into a fixed-shape hidden state
- RNN *decoder* can predict the next token based on what tokens have been generated (such as in language modeling), together with the encoded information of the input sequence
- Different approaches on RNN decoder

[Sutskever et al., 2014]



[Cho et al., 2014b]



RNN Encoder

- RNN transforms the input vector $\mathbf{x}_t$ and the hidden state $\mathbf{h}_{t-1}$ from the previous time step into the current hidden state $\mathbf{h}_t$
- The encoder transforms the hidden states at all the time steps into the context variable through a customized function q

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}). \Rightarrow \mathbf{c} = q(\mathbf{h}_1, \ldots, \mathbf{h}_T).$$

$$q(\mathbf{h}_1, \ldots, \mathbf{h}_T) = \mathbf{h}_T$$ for vanilla Seq-to-seq

Machine Translation

RNN Decoder

- RNN decoder take the output $\boldsymbol{y}_{t-1}$, the previous hidden state $\boldsymbol{s}_{t-1}$, and context variable $\mathbf{c}$ for current hidden state $\boldsymbol{s}_t$.

$$\mathbf{s}_{t'} = g(y_{t'-1}, \mathbf{c}, \mathbf{s}_{t'-1}).$$

- After obtaining the hidden state of the decoder, we can use an output layer and the softmax operation to compute the conditional probability distribution, $P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c})$.

Loss function

- Softmax cross-entropy loss with exclusion of padding for loss calculations.

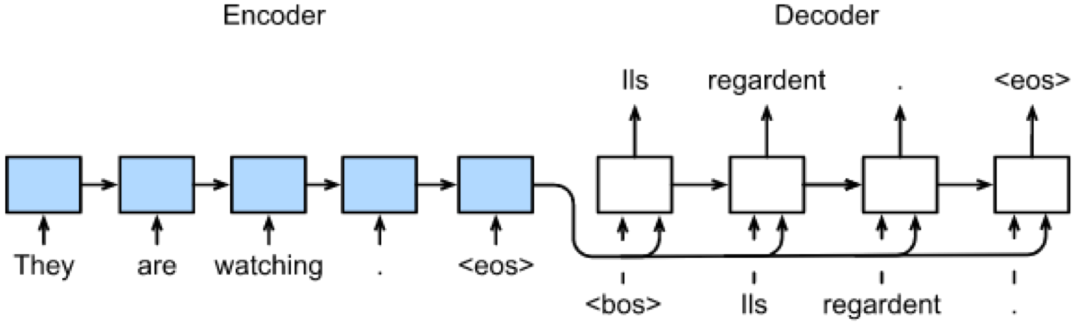
Training: Teacher forcing

- The original output sequence (token labels) is fed into the decoder excluding the final token of decoder

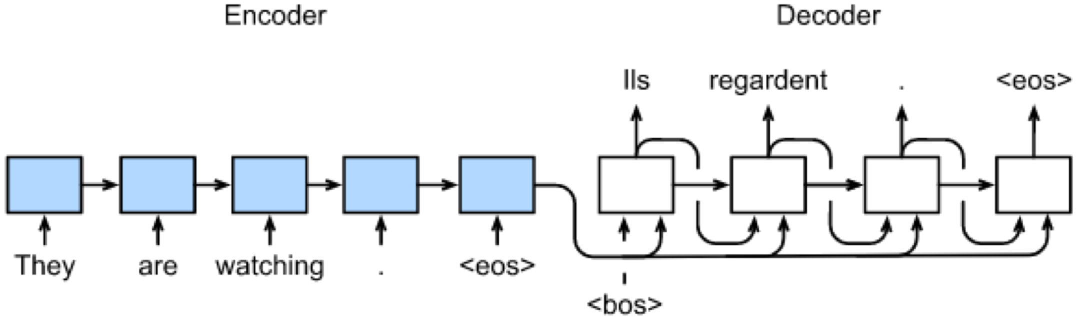
Prediction

- The predicted token from the previous time step is fed into the decoder as an input

Training



Prediction



Machine Translation

RNN Decoder

- RNN decoder take the output $\boldsymbol{y}_{t-1}$, the previous hidden state $\boldsymbol{s}_{t-1}$, and context variable $\mathbf{c}$ for current hidden state $\boldsymbol{s}_t$.

$$\mathbf{s}_{t'} = g(y_{t'-1}, \mathbf{c}, \mathbf{s}_{t'-1}).$$

- After obtaining the hidden state of the decoder, we can use an output layer and the softmax operation to compute the conditional probability distribution, $P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c})$.

Loss function

- Softmax cross-entropy loss with exclusion of padding for loss calculations.

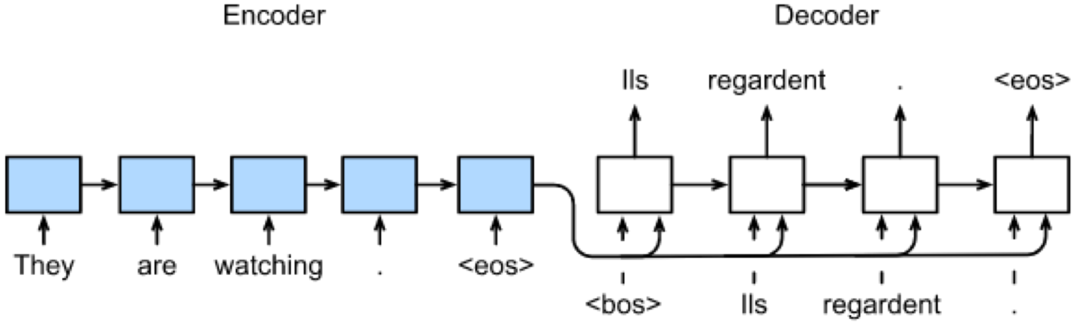
Training: Teacher forcing

- The original output sequence (token labels) is fed into the decoder excluding the final token of decoder

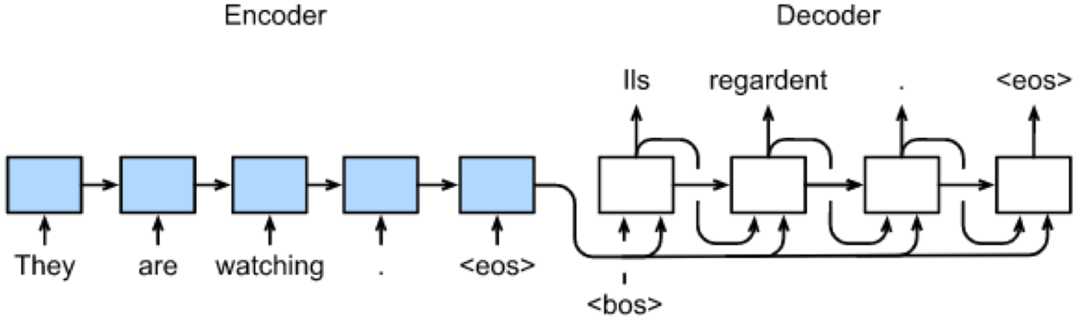
Prediction

- The predicted token from the previous time step is fed into the decoder as an input

Training



Prediction



Evaluation metric, BLEU (Bilingual Evaluation Understudy)

- BLEU evaluates whether this ***n*-grams** appears in the label sequence.

Predicted: [A, B, B, C, D]

Target: [A, B, C, D, E, F]

1-gram		2-gram		3-gram		4-gram	
A	**A**	**AB**	**AB**	ABB	ABC	ABBC	ABCD
B	**B**	**BB**	**BC**	BBC	**BCD**	BBCD	BCDE
B	**C**	BC	**CD**	**BCD**	CDE		CDEF
C	**D**	**CD**	DE		DEF		
D	E		EF				
	F						
4/5		3/4		1/3		0/4	

$$\exp\left(\min\left(0, 1-\frac{\mathrm{len}_{\mathrm{label}}}{\mathrm{len}_{\mathrm{pred}}}\right)\right)\prod_{n=1}^{k} p_n^{1/2^n},$$

```
go . => va , maintenant ., bleu 0.000
i lost . => je le <unk> en nous ., bleu 0.000
i'm home . => je suis tom ., bleu 0.512
he's calm . => il est paresseux ., bleu 0.658
```

- BLEU assigns a greater weight to a longer n-gram precision
- Since predicting shorter sequences tends to obtain a higher p_n value, penalizes shorter predicted sequences
- Example
  - Predicted: [A, B], Target: [A, B, C, D, E, F] => $p_1 = 1$, $p_2 = 1$, $\exp(1 - {}^{6}/_{2}) = 0.14$

Beam Search

Searching strategy

- Searching candidate predicted output tokens for feed to next predictions until the occurrence of <eos> token

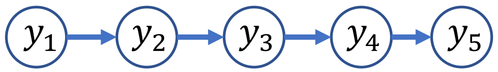
Greedy Search

- taking only **one** output token for each prediction with the maximum likelihood

0.5 × 0.4 × 0.4 × 0.6 = 0.048

Time step	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

$$y_{t'} = \operatorname*{argmax}_{y \in \mathcal{Y}} P(y \mid y_1, \ldots, y_{t'-1}, \mathbf{c}),$$

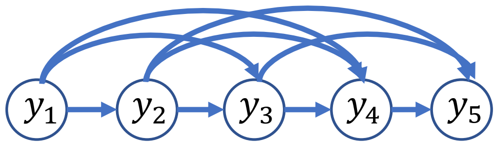


- This cannot guarantee the optimal sequence, since optimal sequence requires all possible combinations with previous steps

0.5 × 0.3 × 0.6 × 0.6 = **0.054**

Time step	1	2	3	4
A	0.5	0.1	0.1	0.1
B	0.2	0.4	0.6	0.2
C	0.2	0.3	0.2	0.1
<eos>	0.1	0.2	0.1	0.6

$$\prod_{t'=1}^{T'} P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c})$$



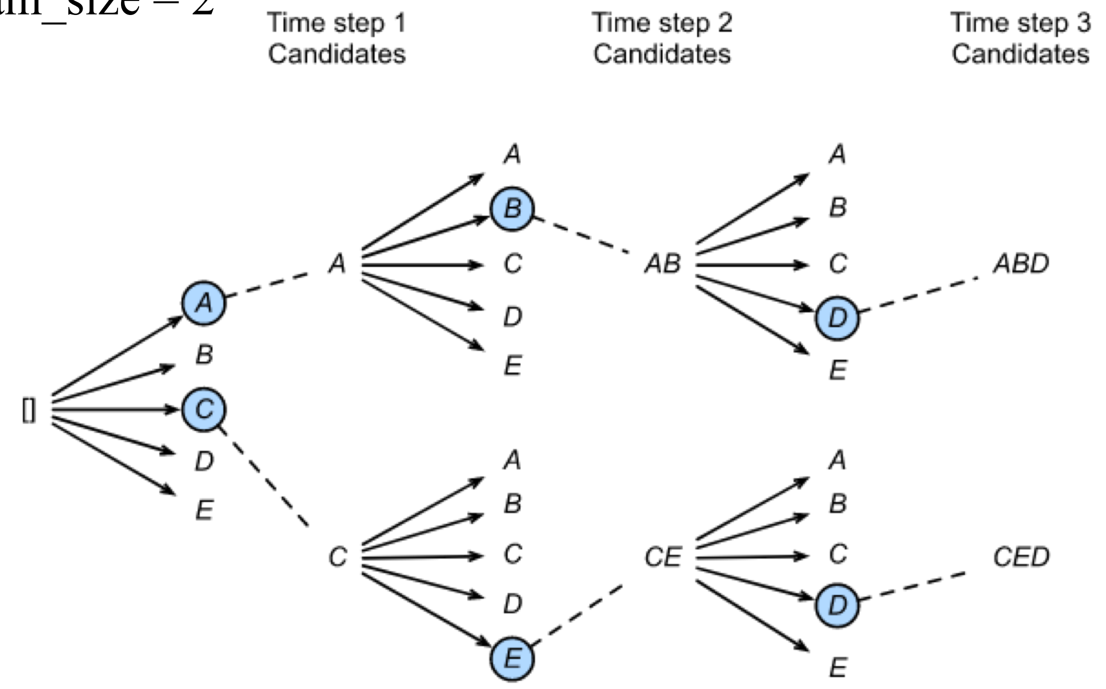
- Computation cost: O(|V||S|), (10 sequence length |S|, 10000 word corpus |V| => 10000 × 10 = 100000 = 10^5)

Beam Search

Exhaustive Search
- Exhaustively enumerate all the possible output sequences, then select the one output with the highest conditional probability.
- Too high computational cost, $O(|V|^{|S|})$ (10 sequence length |S|, 10000 word corpus |V| => $10000^{10} = 10^{40}$)

Bean Search
- Select *k* tokens with the highest probabilities at each step and continue to select total *k* candidate output, O(k|S||V|)
- Greedy search can be treated as a special type of beam search with a beam size of 1

beam_size = 2



Candidate:
(i) A (ii) C (iii) A, B (iv) C, E (v) A, B, D (vi) C, E, D.

$$P(y_1 \mid \mathbf{c}) = \boldsymbol{A, C}$$

$$P(A, y_2 \mid \mathbf{c}) = P(A \mid \mathbf{c})P(y_2 \mid A, \mathbf{c}),$$
$$P(C, y_2 \mid \mathbf{c}) = P(C \mid \mathbf{c})P(y_2 \mid C, \mathbf{c}),$$

$$P(A, B, y_3 \mid \mathbf{c}) = P(A, B \mid \mathbf{c})P(y_3 \mid A, B, \mathbf{c}),$$
$$P(C, E, y_3 \mid \mathbf{c}) = P(C, E \mid \mathbf{c})P(y_3 \mid C, E, \mathbf{c}),$$

Bean Search (Cont.)

- Choose the sequence with the highest of the following score,
  - L is the length of a candidate sequence, a is hyperparameter which usually set to 0.75

$$\frac{1}{L^{\alpha}}\log P(y_1, \ldots, y_L) = \frac{1}{L^{\alpha}}\sum_{t'=1}^{L}\log P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c}),$$

Candidate:

(i) A (ii) C (iii) A, B (iv) C, E (v) A, B, D (vi) C, E, D.

Time step	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

A	**1/(1^0.75) × log(0.5) = 1 × -0.301 = -0.301**
C	1/(1^0.75) × log(0.2) = 1 × -0.698 = -0.698
AB	1/(2^0.75) × (log(0.5) + log(0.4)) = 0.594 × -0.698 = -0.414
CC	1/(2^0.75) × (log(0.2) + log(0.3)) = 0.594 × -1.221 = -0.725
ABB	1/(3^0.75) × (log(0.5) + log(0.4) + log(0.2)) = 0.438 × -1.397 = -0.611
CCB	1/(3^0.75) × (log(0.2) + log(0.3) + log(0.4)) = 0.438 × -1.619 = -0.709