CS 533: Natural Language Processing

Conditional Neural Language Models

Karl Stratos

Rutgers University

Language Models Considered So Far

$$p_{Y|X}(y|x_{1:100})$$

- Classical trigram models: $q_{Y|X}(y|x_{99},\ x_{100})$
  - Training: closed-form solution
- Log-linear models: $\text{softmax}_y([w^\top \phi((x_{99},\ x_{100}),\ y')]_{y'})$
  - Training: gradient descent on convex loss
- Neural models
  - Feedforward: $\text{softmax}_y(\text{FF}([E_{x_{99}},\ E_{x_{100}}]))$
  - Recurrent: $\text{softmax}_y(\text{FF}(h(x_{1:99}), E_{x_{100}}))$
  - Training: gradient descent on nonconvex loss

Conditional Language Models

- Machine translation

  `And the programme has been implemented` ⇒ `Le programme a été mis en application`

- Summarization

  `russian defense minister ivanov called sunday for the creation of a joint front for combating global terrorism` ⇒ `russia calls for joint front against terrorism`

- Data-to-text generation

TEAM	WIN	LOSS	PTS	FG_PCT	RB	AS ...
Heat	11	12	103	49	47	27
Hawks	7	15	95	43	33	20

PLAYER	AS	RB	PT	FG	FGA	CITY ...
Tyler Johnson	5	2	27	8	16	Miami
Dwight Howard	4	17	23	9	11	Atlanta
Paul Millsap	2	9	21	8	12	Atlanta
Goran Dragic	4	2	21	8	17	Miami
Wayne Ellington	2	3	19	7	15	Miami
Dennis Schroder	7	4	17	8	15	Atlanta
Rodney McGruder	5	5	11	3	8	Miami
Thabo Sefolosha	5	5	10	5	11	Atlanta
Kyle Korver	5	3	9	3	9	Atlanta
...						

The Atlanta Hawks defeated the Miami Heat , 103 - 95 , at Philips Arena on Wednesday . Atlanta was in desperate need of a win and they were able to take care of a shorthanded Miami team here . Defense was key for the Hawks , as they held the Heat to 42 percent shooting and forced them to commit 16 turnovers . Atlanta also dominated in the paint , winning the rebounding battle , 47 - 34 , and outscoring them in the paint 58 - 26.The Hawks shot 49 percent from the field and assisted on 27 of their 43 made baskets . This was a near wire - to - wire win for the Hawks , as Miami held just one lead in the first five minutes . Miami (7 - 15) are as beat - up as anyone right now and it 's taking a toll on the heavily used starters . Hassan Whiteside really struggled in this game , as he amassed eight points , 12 rebounds and one blocks on 4 - of - 12 shooting ...

(Wiseman et al., 2017)

- Image captioning

⇒ `the dog saw the cat`

Encoder-Decoder Models

Much of machine learning is learning $x \mapsto y$ where x, y are some complicated structures

Encoder-decoder models are **conditional** models that handle this wide class of problems in two steps:

1. **Encode** the given input x using some architecture.
2. **Decode** output y.

Training: again minimize cross entropy

$$\min_{\theta} \mathop{\mathbf{E}}_{(\text{input},\text{output}) \sim p_{Y|X}} \left[-\ln q^{\theta}_{Y|X}(\text{output}|\text{input}) \right]$$

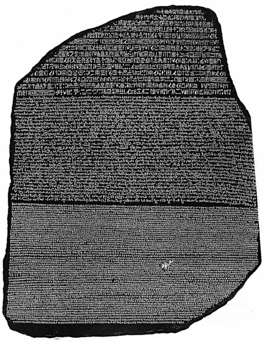
Agenda

1. MT
2. Attention in detail
3. Beam Search

Machine Translation (MT)

- Goal: Translate text from one language to another.

- One of the oldest problems in artificial intelligence.



Some History

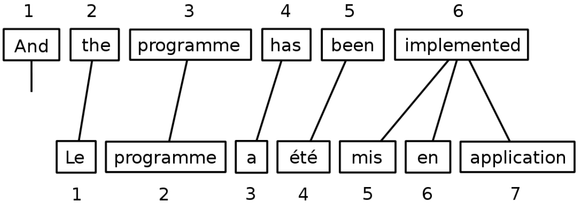
- Early '90s: Rise of statistical MT (SMT)

- Exploit **parallel text**.

  `And the programme has been implemented`

  `Le programme a été mis en application`

- Infer word alignment ("IBM" models, Brown et al., 1993)



SMT: Huge Pipeline

1. Use IBM models to extract word alignment, phrase alignment (Koehn et al., 2003).
2. Use syntactic analyzers (e.g., parser) to extract features and manipulate text (e.g., phrase re-ordering).
3. Use a separate language model to enforce fluency.
4. ...

Multiple independently trained models patched together

- Really complicated, prone to error propogation

Rise of Neural MT

Started taking off around 2014

- Replaced the entire pipeline with a **single** model
- Called "end-to-end" training/prediction

  Input: `Le programme a été mis en application`
  Output: `And the programme has been implemented`

- **Revolution** in MT
  - Better performance, *way* simpler system
  - A hallmark of the recent neural domination in NLP
  - Key: attention mechanism

Recap: Recurrent Neural Network (RNN)

- Always think of an RNN as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$

  Input: an input vector $x \in \mathbb{R}^d$, a state vector $h \in \mathbb{R}^{d'}$
  Output: a new state vector $h' \in \mathbb{R}^{d'}$

- Left-to-right RNN processes input sequence $x_1 \dots x_m \in \mathbb{R}^d$ as

$$h_i = \phi\left(x_i, h_{i-1}\right)$$

  where h_0 is an initial state vector.

- Idea: h_i is a *representation* of x_i that has incorporated *all inputs to the left*.

$$h_i = \phi\left(x_i, \phi\left(x_{i-1}, \phi\left(x_{i-2}, \cdots \phi\left(x_1, h_0\right) \cdots\right)\right)\right)$$

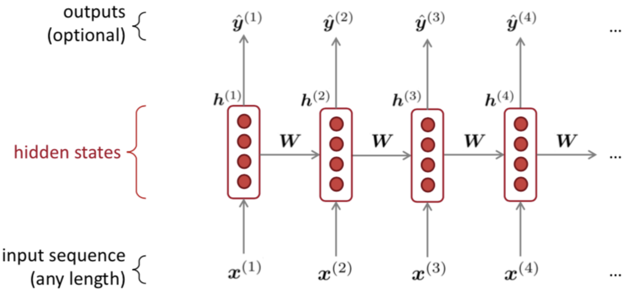
Variety 1: "Simple" RNN

- Parameters $U \in \mathbb{R}^{d' \times d}$ and $V \in \mathbb{R}^{d' \times d'}$

$$h_i = \tanh\left(Ux_i + Vh_{i-1}\right)$$

Picture

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \in \mathbb{R}^d$$



Stacked Simple RNN

- Parameters $U^{(1)} \ldots U^{(L)} \in \mathbb{R}^{d' \times d}$ and $V^{(1)} \ldots V^{(L)} \in \mathbb{R}^{d' \times d'}$

$$
\begin{aligned}
h_i^{(1)} &= \tanh\left(U^{(1)} x_i + V^{(1)} h_{i-1}^{(1)}\right) \\
h_i^{(2)} &= \tanh\left(U^{(2)} h_i^{(1)} + V^{(2)} h_{i-1}^{(2)}\right) \\
&\vdots \\
h_i^{(L)} &= \tanh\left(U^{(L)} h_i^{(L-1)} + V^{(L)} h_{i-1}^{(L)}\right)
\end{aligned}
$$

- Think of it as mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{Ld'} \to \mathbb{R}^{Ld'}$.

$$
x_i \quad \begin{bmatrix} h_{i-1}^{(1)} \\ \vdots \\ h_{i-1}^{(L)} \end{bmatrix} \quad \mapsto \quad \begin{bmatrix} h_i^{(1)} \\ \vdots \\ h_i^{(L)} \end{bmatrix}
$$

Variety 2: Long Short-Term Memory (LSTM)

- Parameters $U^q, U^c, U^o \in \mathbb{R}^{d' \times d}$, $V^q, V^c, V^o, W^q, W^o \in \mathbb{R}^{d' \times d'}$

$$
\begin{aligned}
q_i &= \sigma\left(U^q x_i + V^q h_{i-1} + W^q c_{i-1}\right) \\
c_i &= (1 - q_i) \odot c_{i-1} + q_i \odot \tanh\left(U^c x_i + V^c h_{i-1}\right) \\
o_i &= \sigma\left(U^o x_i + V^o h_{i-1} + W^o c_i\right) \\
h_i &= o_i \odot \tanh(c_i)
\end{aligned}
$$

- Idea: "Memory cells" c_i can carry long-range information.
  - What happens if q_i is close to zero?

- Can be stacked as in simple RNN.

Translation Problem

- Vocabulary of the **source** language V^{src}

$$V^{\text{src}} = \left\{ \text{그, 개가, 보았다, 소식, 2017, 5월} \ldots \right\}$$

- Vocabulary of the **target** language V^{trg}

$$V^{\text{trg}} = \left\{ \text{the, dog, cat, 2021, May, } \ldots \right\}$$

- **Task.** Given any sentence $x_1 \ldots x_m \in V^{\text{src}}$, produce a corresponding translation $y_1 \ldots y_n \in V^{\text{trg}}$.

개가 짖었다 $\Longrightarrow$ the dog barked

Evaluating Machine Translation

- T: human-translated sentences
- $\widehat{T}$: machine-translated sentences
- p_n: precision of n-grams in $\widehat{T}$ against n-grams in T (sentence-wise)
- BLEU: Controversial but popular scheme to automatically evaluate translation quality

$$\text{BLEU} = \min\left(1, \frac{\left|\widehat{T}\right|}{|T|}\right) \times \left(\prod_{n=1}^{4} p_n\right)^{\frac{1}{4}}$$

Translation Model: Conditional Language Model

A **translation model** defines a probability distribution $p(y_1 \ldots y_n | x_1 \ldots x_m)$ over all sentences $y_1 \ldots y_n \in V^{\text{trg}}$ conditioning on any sentence $x_1 \ldots x_m \in V^{\text{src}}$.

Goal: Design a *good* translation model

$$
\begin{aligned}
p(\texttt{the dog barked}|\text{개가 짖었다}) &> p(\texttt{the cat barked}|\text{개가 짖었다}) \\
&> p(\texttt{dog the barked}|\text{개가 짖었다}) \\
&> p(\texttt{oqc shgwqw\#w 1g0}|\text{개가 짖었다})
\end{aligned}
$$

How can we use an RNN to build a translation model?

Basic Encoder-Decoder Framework

Model parameters

- Vector $e_x \in \mathbb{R}^d$ for every $x \in V^{\text{src}}$
- Vector $e_y \in \mathbb{R}^d$ for every $y \in V^{\text{trg}} \cup \{*\}$
- Encoder RNN $\psi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ for V^{src}
- Decoder RNN $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ for V^{trg}
- Feedforward $f : \mathbb{R}^{d'} \rightarrow \mathbb{R}^{|V^{\text{trg}}|+1}$

Basic idea

1. Transform $x_1 \ldots x_m \in V^{\text{src}}$ with ψ into some representation ξ.
2. Build a language model ϕ over V^{trg} conditioning on ξ.

Encoder

For $i = 1 \dots m$,

$$h_i^{\psi} = \psi\left(e_{x_i}, h_{i-1}^{\psi}\right)$$

$$h_m^{\psi} = \psi\left(e_{x_m}, \psi\left(e_{x_{m-1}}, \psi\left(e_{x_{m-2}}, \cdots \psi\left(e_{x_1}, h_0^{\psi}\right) \cdots\right)\right)\right)$$

Decoder

Initialize $h_0^\phi = h_m^\psi$ and $y_0 = *$.

For $i = 1, 2, \ldots$, the decoder defines a probability distribution over $V^{\text{trg}} \cup \{\texttt{STOP}\}$ as

$$h_i^\phi = \phi\left(e_{y_{i-1}},\ h_{i-1}^\phi\right)$$

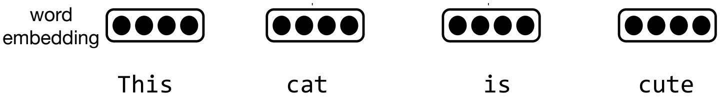
$$p_\Theta(y|x_1 \ldots x_m,\ y_0 \ldots y_{i-1}) = \mathsf{softmax}_y(f(h_i^\phi))$$

Probability of translation $y_1 \ldots y_n$ given $x_1 \ldots x_m$:

$$p_\Theta(y_1 \ldots y_n|x_1 \ldots x_m) = \prod_{i=1}^{n} p_\Theta(y_i|x_1 \ldots x_m,\ y_0 \ldots y_{i-1}) \times$$
$$p_\Theta(\texttt{STOP}|x_1 \ldots x_m,\ y_0 \ldots y_n)$$

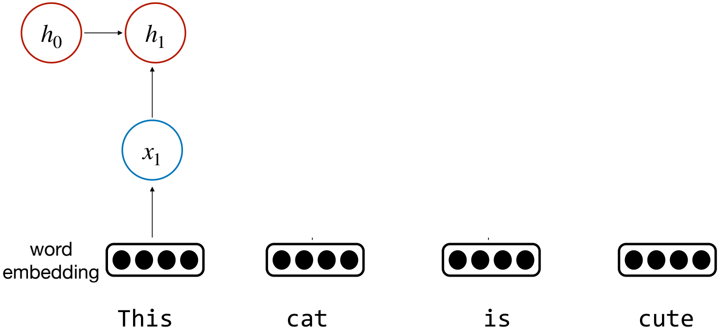
Encoder

Sentence: This cat is cute



Encoder

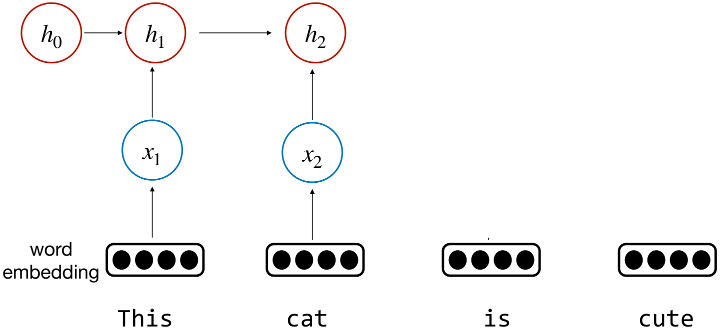
Sentence: This cat is cute



Slide Credit: Danchi Chen & Karthik Narasimhan

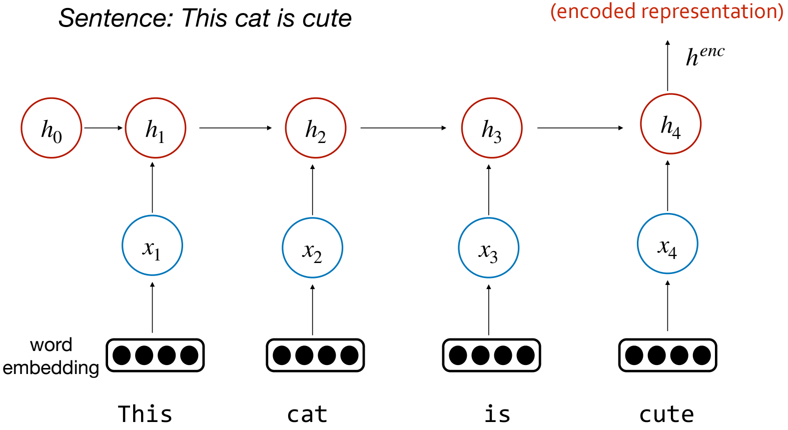
Encoder

Sentence: This cat is cute



Slide Credit: Danchi Chen & Karthik Narasimhan

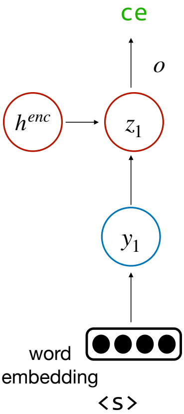
Encoder



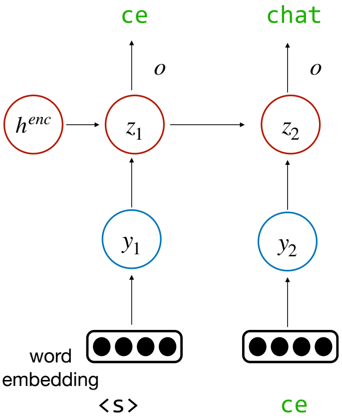
Slide Credit: Danchi Chen & Karthik Narasimhan

Decoder

Decoder

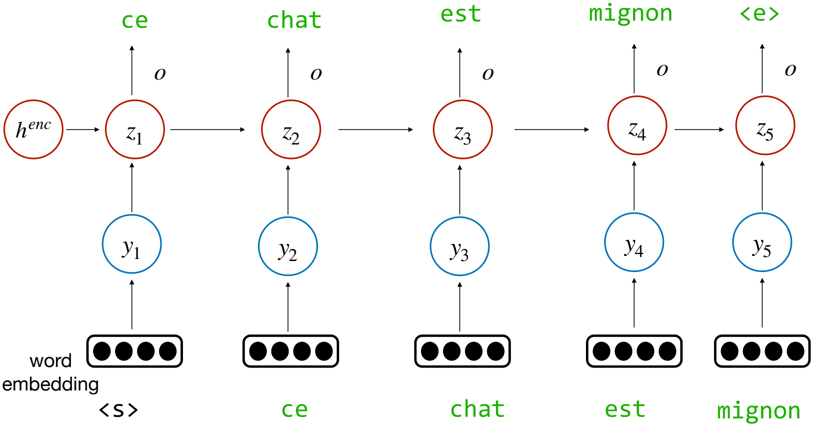


Decoder



Decoder

- A conditioned language model



Training

Given parallel text of N sentence-translation pairs $(x^{(1)}, y^{(1)}) \dots (x^{(N)}, y^{(N)})$, find parameters Θ^* that maximize the log likelihood of the data:

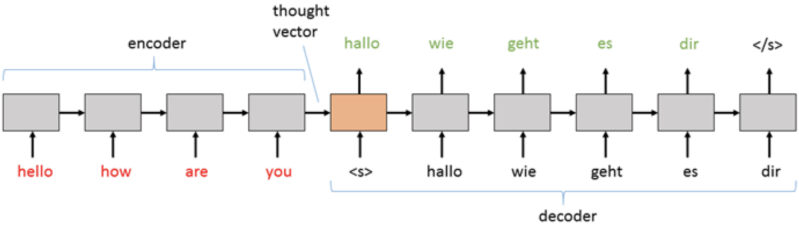
$$\Theta^* \approx \arg\min_{\Theta} \underbrace{-\sum_{i=1}^{N} \log p_\Theta(y^{(i)}|x^{(i)})}_{\text{loss}}$$

In PyTorch

```
loss.back()
optim.step()
```

Training not trivial due to exploting/vanishing gradients

Sequence-to-Sequence (Seq2Seq) Learning (Sutskever et al., 2014)



Problems?

Decoder with Attention

- Instead of using 1 fixed vector to encode all $x_1 \dots x_m$, **decoder decides which words to pay attention to**, at every step.

- For $i = 0, 1, \dots$,

$$p_\Theta(y | x_1 \dots x_m,\ y_0 \dots y_i) = \mathrm{softmax}_y \left(\mathrm{FF} \left(\sum_{j=1}^{m} \alpha_{i,j} h_j^\psi \right) \right)$$

Attention Weights

$$\sum_{j=1}^{m} \alpha_{i,j} h_j^{\psi}$$

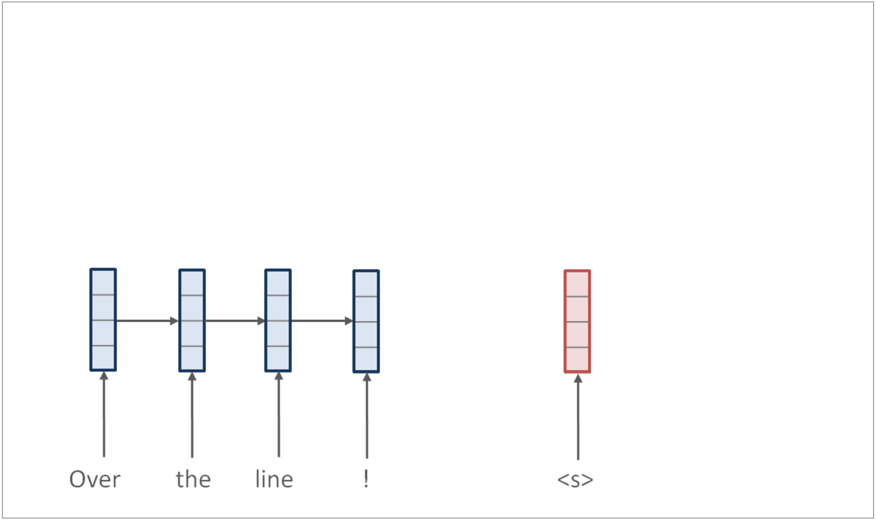
- $\alpha_{i,j}$: Importance of x_j for predicting i-th translation
- Various options

$$\beta_{i,j} = u^\top \tanh\left(W h_i^{\phi} + V h_j^{\psi}\right)$$
$$\beta_{i,j} = \left(h_i^{\phi}\right)^\top B h_j^{\psi}$$

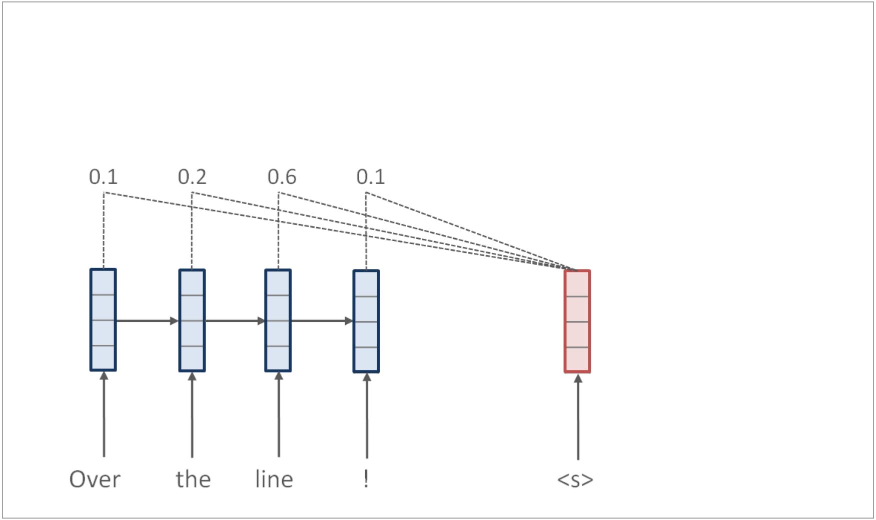
Typically take softmax to make them probabilities:

$$(\alpha_{i,1} \ldots \alpha_{i,m}) = \mathsf{softmax}\left(\beta_{i,1} \ldots \beta_{i,m}\right)$$

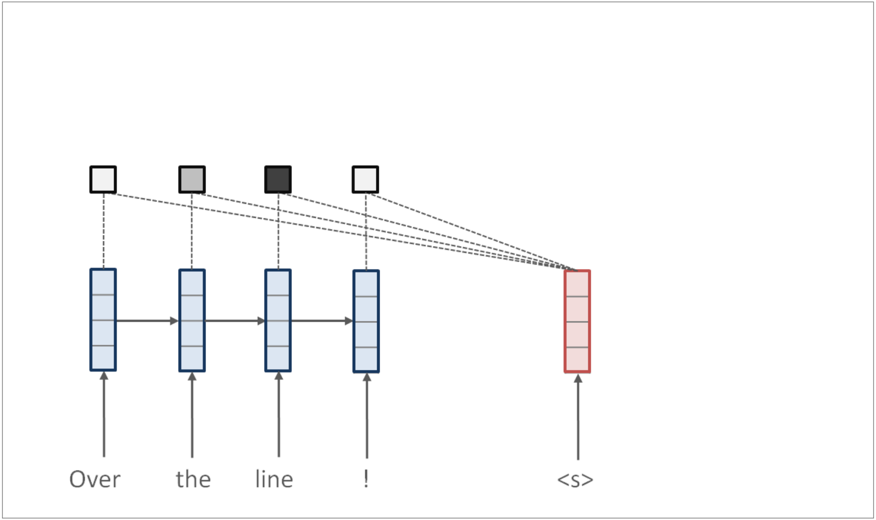
Encoder-Decoder with Attention: Slides by Sasha Rush



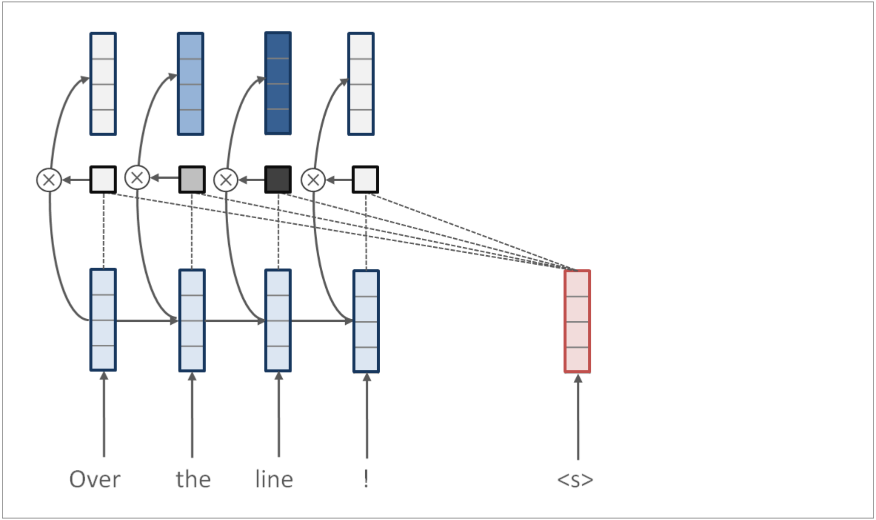
Encoder-Decoder with Attention: Slides by Sasha Rush



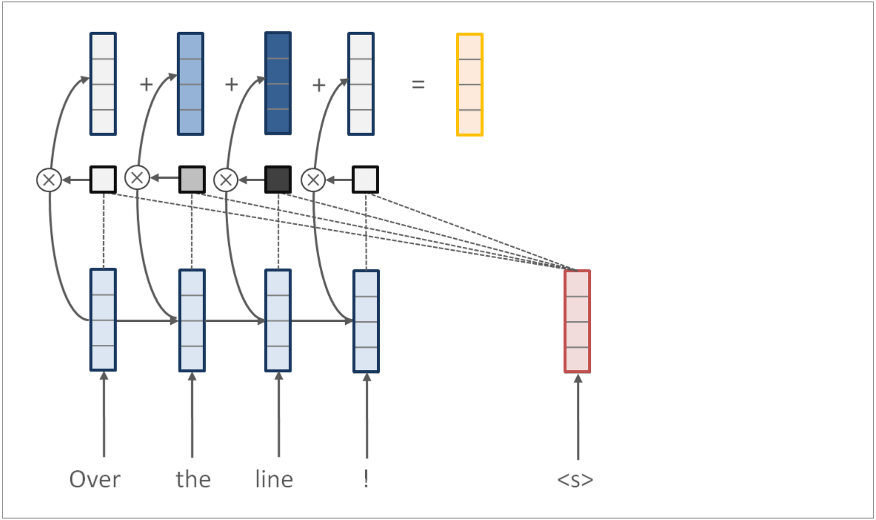
Encoder-Decoder with Attention: Slides by Sasha Rush



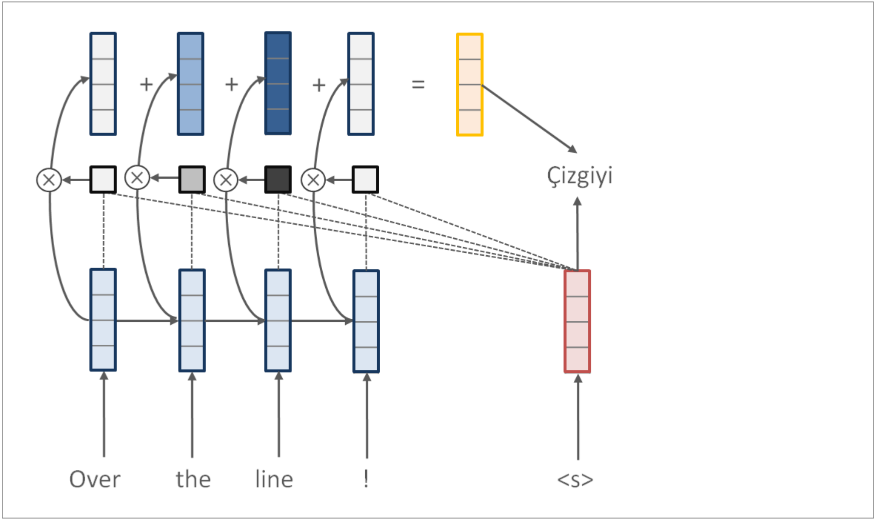
Encoder-Decoder with Attention: Slides by Sasha Rush



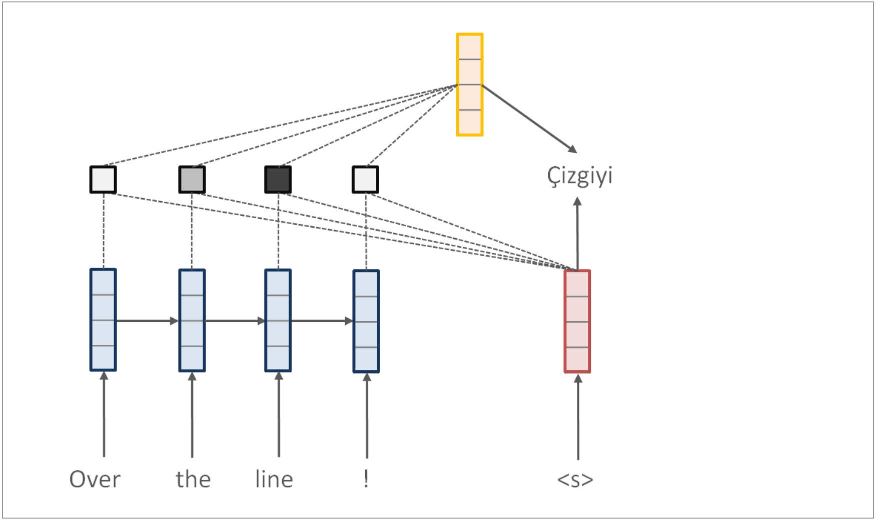
Encoder-Decoder with Attention: Slides by Sasha Rush



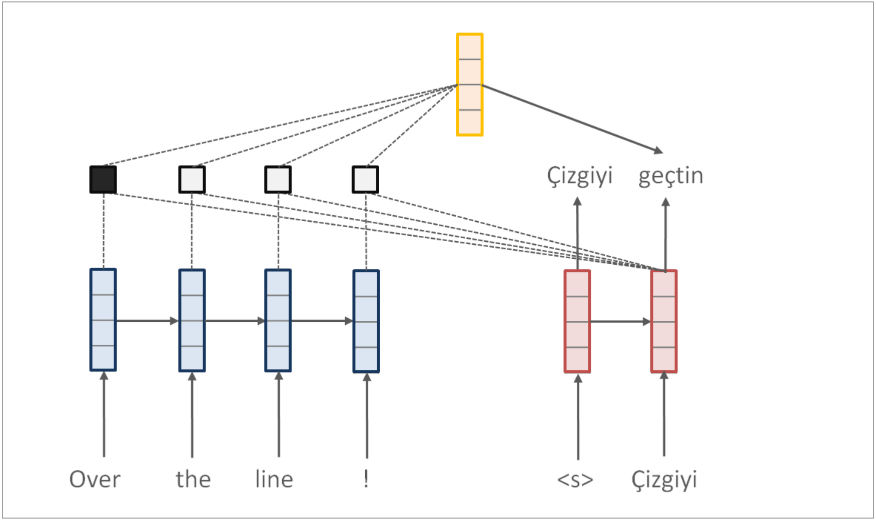
Encoder-Decoder with Attention: Slides by Sasha Rush



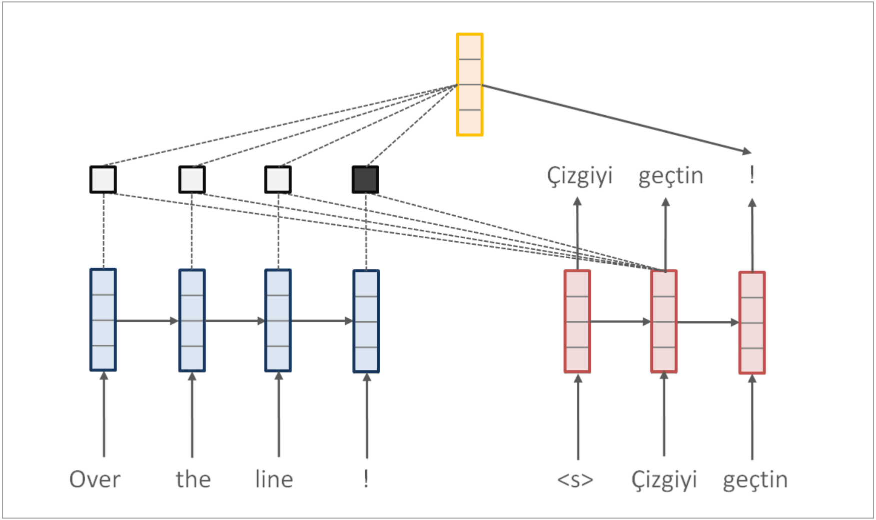
Encoder-Decoder with Attention: Slides by Sasha Rush



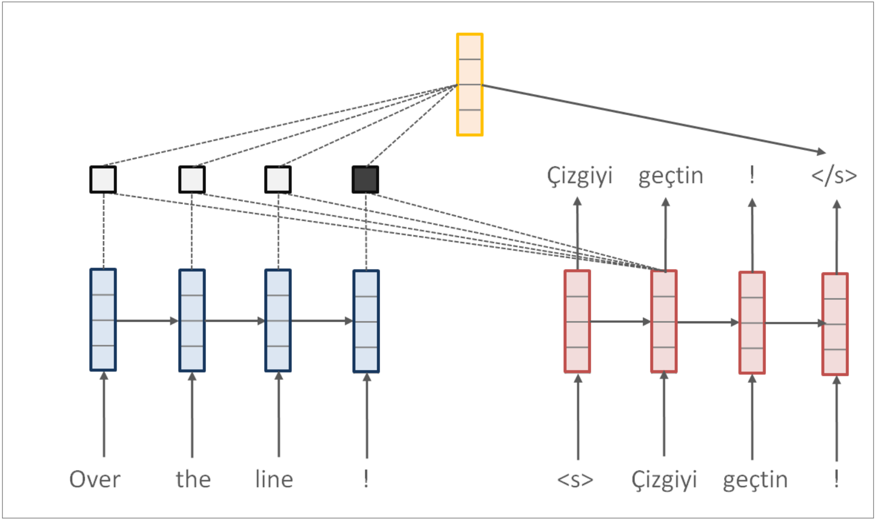
Encoder-Decoder with Attention: Slides by Sasha Rush



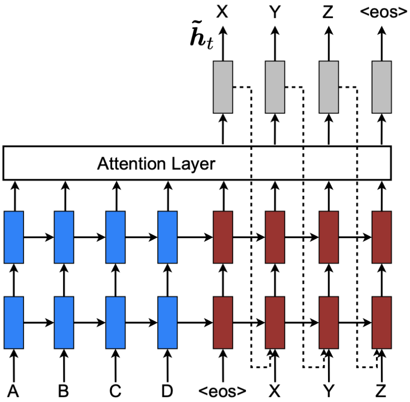
Encoder-Decoder with Attention: Slides by Sasha Rush



Encoder-Decoder with Attention: Slides by Sasha Rush



Input-Feeding Approach (Luong et al., 2015)



Explicit alignment information. Very large computation graph. Parallel computation during training no longer possible.

Matrix Form of Input-Feeding Attention

Bank $X \in \mathbb{R}^{d \times T}$, hidden state $h_{t-1} \in \mathbb{R}^d$, current word $y_t \in V$

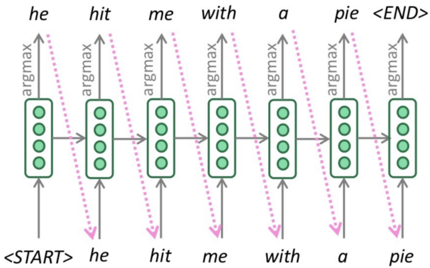
(Board)

Greedy Decoding

Given sentence x: for $t = 1 \dots T_{\max}$

$$y_t \leftarrow \operatorname*{arg\,max}_{y \in V \cup \{\texttt{STOP}\}} \log q(y|y_{<t}, x)$$

stop if $y = \texttt{STOP}$.



Is this what we want?

$$y^* = \operatorname*{arg\,max}_{y \in V^+ :\, |y| \leq T_{\max}} \log q(y|x)$$

Beam Search: Idea

- Instead of enumerating $|V|^{T_{\max}}$ candidates, keep K (called beam size) highest scoring partial structures at every step.

Only an approximation

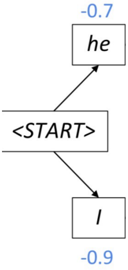
- Applicable to any decomposable score function
- Score function in seq2seq:

$$\begin{aligned}\mathrm{score}(y_1 \dots y_t) &= \log q(y_1 \dots y_t | x) = \sum_{i=1}^{t} \log q(y_i | y_{<i}, x) \\ &= \mathrm{score}(y_1 \dots y_{t-1}) + \log q(y_t | y_{<t}, x)\end{aligned}$$

- Runtime: $O(|V|\, T_{\max} K^2 \log K)$

Beam Search

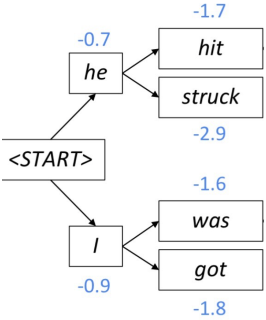
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam Search

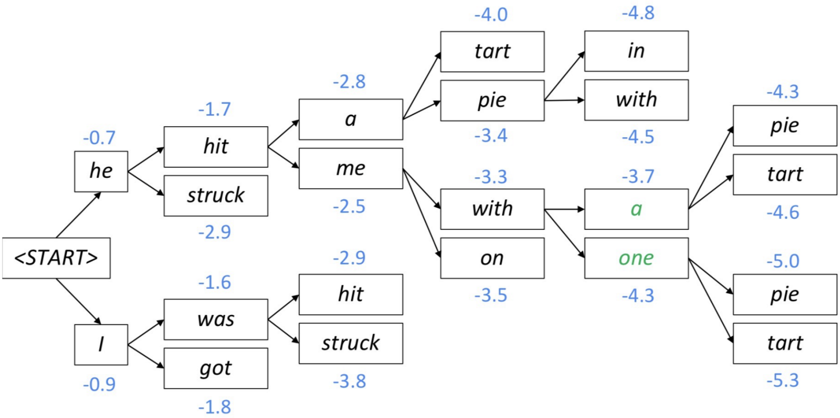
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam Search

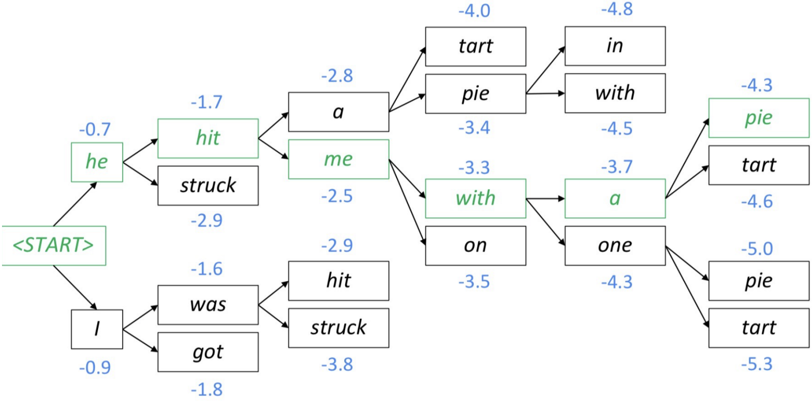
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam Search: Backtrack

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam Search: More Details

- Different hypotheses may stop at different time steps (place them aside)
- Continue beam search until
  - All K hypotheses stop
  - We hit max length limit T
- Select top hypotheses using the normalized likelihood score

$$\frac{1}{M}\sum_{i=1}^{M}\log q(y_i|y_{<i},x)$$

Otherwise hypotheses get higher scores for just being shorter

Copy Mechanism

$$q(y_t|y_{<t}, x) = \sum_{z \in \{0,1\}} q(y_t, z|y_{<t}, x)$$

Picture (Credit: See et al., 2017)