RNN Encoder-Decoder and Attention models

2016.07.25

Jaemin Cho

Introduction

Today we will cover..

- Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
- Sequence to Sequence Learning with Neural Networks
- Neural Machine Translation by Jointly Learning to Align and Translate
- Memory Networks
- Pointer Networks

Basic RNN model for Text Generation (Prediction)

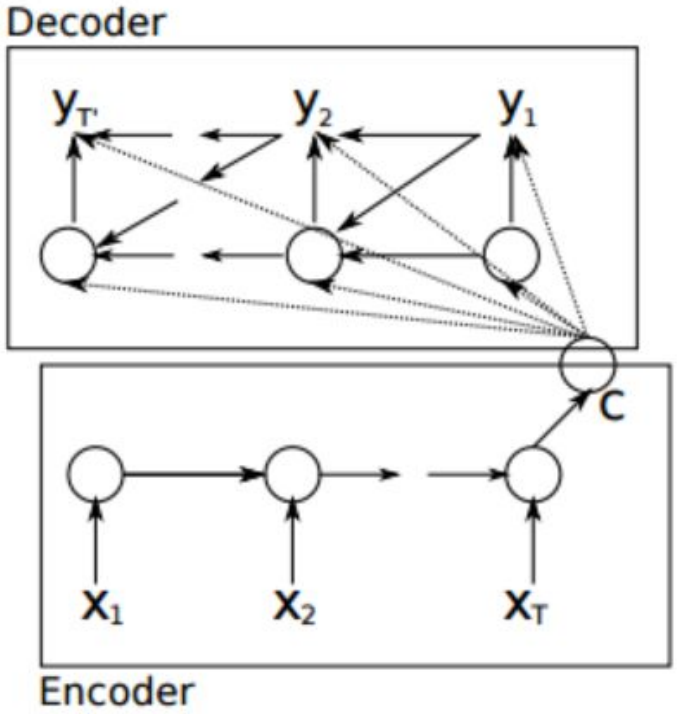
$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right)$$

$$p(x_{t,j} = 1 \mid x_{t-1}, \ldots, x_1) = \frac{\exp\left(\mathbf{w}_j \mathbf{h}_{\langle t \rangle}\right)}{\sum_{j'=1}^{K} \exp\left(\mathbf{w}_{j'} \mathbf{h}_{\langle t \rangle}\right)}$$

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

- First model that used RNN-encoder + RNN-decoder (RNNencdec)
- Introduced GRU

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

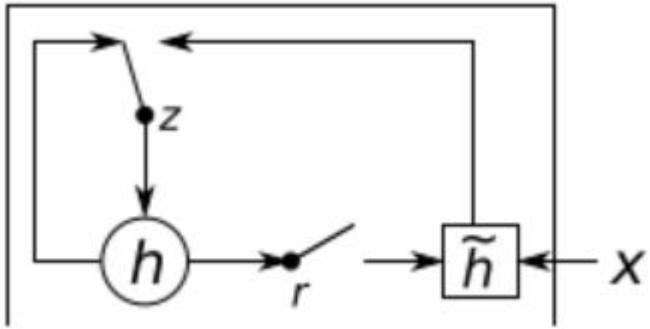


$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right),$$

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g\left(\mathbf{h}_{\langle t \rangle}, y_{t-1}, \mathbf{c}\right)$$

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

- GRU



$$r_j = \sigma\left([\mathbf{W}_r\mathbf{x}]_j + [\mathbf{U}_r\mathbf{h}_{\langle t-1\rangle}]_j\right)$$

$$z_j = \sigma\left([\mathbf{W}_z\mathbf{x}]_j + [\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}]_j\right),$$

$$\tilde{h}_j^{\langle t\rangle} = \phi\left([\mathbf{W}\mathbf{x}]_j + [\mathbf{U}\left(\mathbf{r}\odot\mathbf{h}_{\langle t-1\rangle}\right)]_j\right),$$

$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1-z_j)\tilde{h}_j^{\langle t\rangle}$$

Sequence to Sequence Learning with Neural Networks

- Deep layer LSTM-encoder/decoder
- Reverse order of input sequence
- First RNN model that beated phrased-based SMT baseline

- 4-Layer LSTM (1000 cell each)
- 1000-dimensional word embedding
- WMT 14 dataset (Eng->Fra)

Sequence to Sequence Learning with Neural Networks

- Encoder

$$
\begin{aligned}
h_t &= \text{sigm}\left(W^{\text{hx}} x_t + W^{\text{hh}} h_{t-1}\right) \\
y_t &= W^{\text{yh}} h_t
\end{aligned}
$$

- Decoder

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

Sequence to Sequence Learning with Neural Networks

- Beam search when Decoding

Training objective $1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$ $\hat{T} = \arg\max_T p(T|S)$

- We search for the most likely translation using a simple left-to-right beam search decoder which maintains a s**mall number B of partial hypotheses**, where a partial hypothesis is a prefix of some translation. At each timestep we extend each partial hypothesis in the beam with every possible word in the vocabulary. This greatly increases the number of the hypotheses so we discard all but the B most likely hypotheses according to the model's log probability. **As soon as the "<EOS>" symbol is appended to a hypothesis, it is removed from the beam and is added to the set of complete hypotheses.**

Sequence to Sequence Learning with Neural Networks

- Reversed input

we found it extremely valuable to **reverse the order of the words of the input sentence.** So for example, instead of mapping the sentence a, b, c to the sentence α, β, γ, the **LSTM is asked to map c, b, a to α, β, γ, where α, β, γ is the translation of a, b, c.**

This way, a is in close proximity to α, b is fairly close to β, and so on, a fact that makes it easy for SGD to “establish communication” between the input and the output. We found this simple data transformation to greatly boost the performance of the LSTM.

Neural Machine Translation by Jointly Learning to Align and Translate

- First RNN model applied attention mechanism
- Non fixed-length context vector
- Bidirectional RNN Encoder (RNNSearch)
- Great at long sentences (compared to RNNencdec)

- WMT 14 (Eng -> Fra)

Neural Machine Translation by Jointly Learning to Align and Translate

- Decoder: General Description

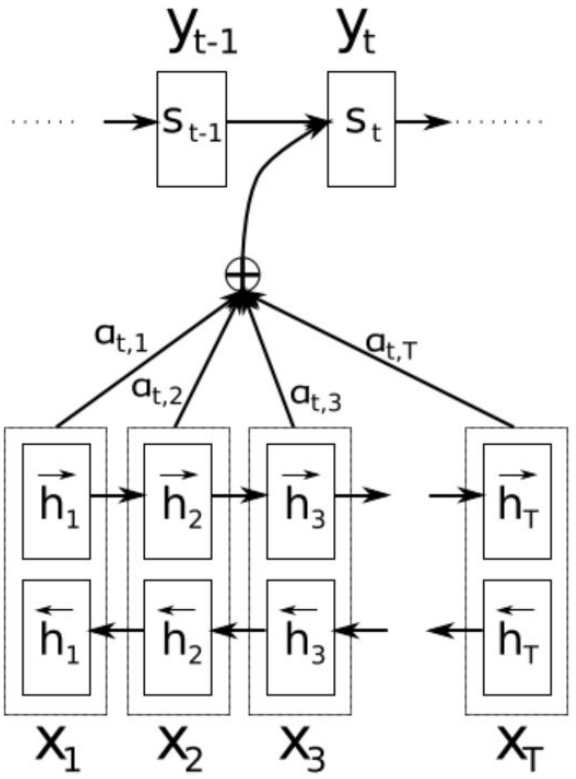
$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1},s_i,c_i)$$

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

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

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

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



Neural Machine Translation by Jointly Learning to Align and Translate

- Decoder: Detail

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

$$s_i = (1-z_i)\circ s_{i-1} + z_i \circ \tilde{s}_i,$$

$$\tilde{s}_i = \tanh\left(Ey_i + U\left[r_i \circ s_{i-1}\right] + Cc_i\right)$$

$$z_i = \sigma\left(W_z E y_i + U_z s_{i-1} + C_z c_i\right)$$

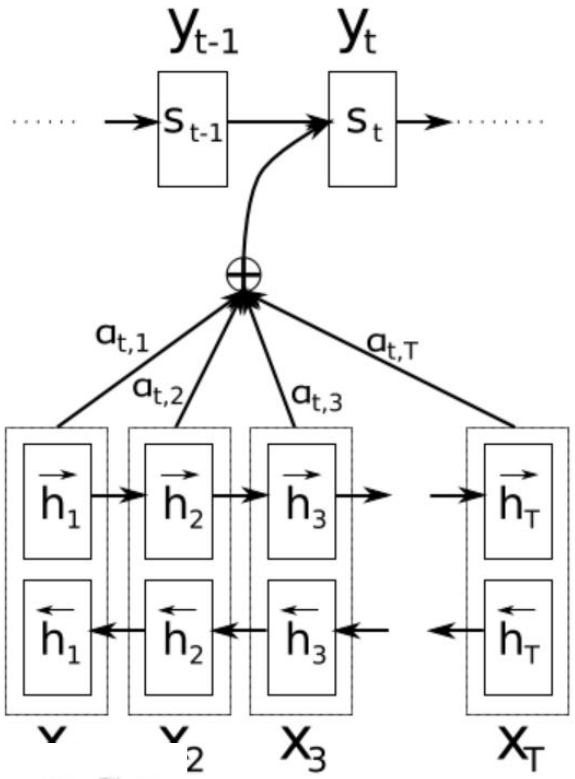
$$r_i = \sigma\left(W_r E y_i + U_r s_{i-1} + C_r c_i\right)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j, \qquad \alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x}\exp\left(e_{ik}\right)}$$

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

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

$$t_i = \left[\max\left\{\tilde{t}_{i,2j-1},\tilde{t}_{i,2j}\right\}\right]_{j=1,\ldots,l}^\top \quad \tilde{t}_i = U_o s_{i-1} + V_o E y_{i-1} + C_o c_i.$$



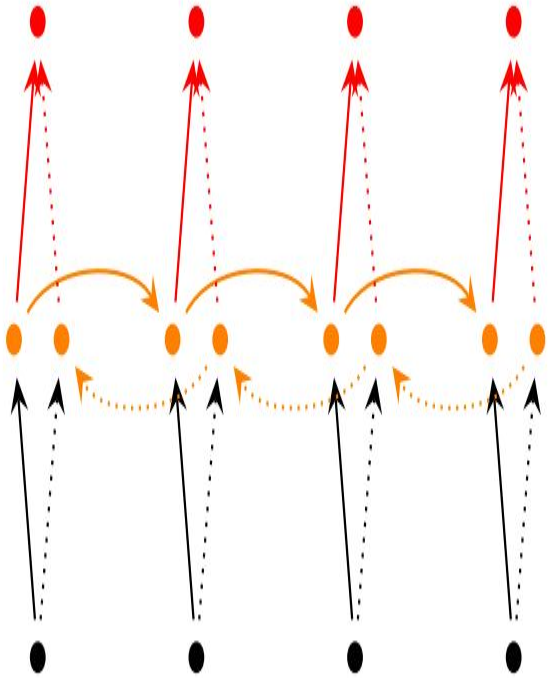
Neural Machine Translation by Jointly Learning to Align and Translate

- Encoder: Bi-RNN for annotating Sequence

forward hidden states $(\overrightarrow{h}_1, \cdots, \overrightarrow{h}_{T_x})$

backward hidden states $(\overleftarrow{h}_1, \cdots, \overleftarrow{h}_{T_x})$

$$h_j = \left[\overrightarrow{h}_j^\top ; \overleftarrow{h}_j^\top\right]^\top$$



Neural Machine Translation by Jointly Learning to Align and Translate

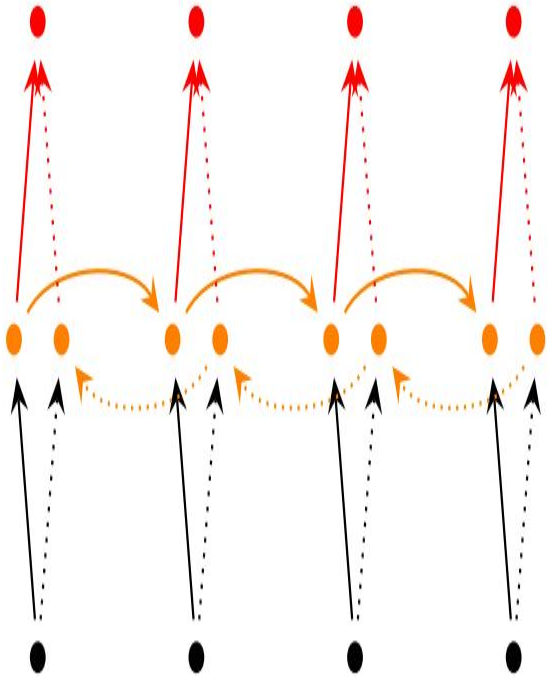
- Encoder: Bi-RNN for annotating Sequence

$$\overrightarrow{h}_i = \begin{cases} (1 - \overrightarrow{z}_i) \circ \overrightarrow{h}_{i-1} + \overrightarrow{z}_i \circ \underline{\overrightarrow{h}}_i & , \text{if } i > 0 \\ 0 & , \text{if } i = 0 \end{cases}$$

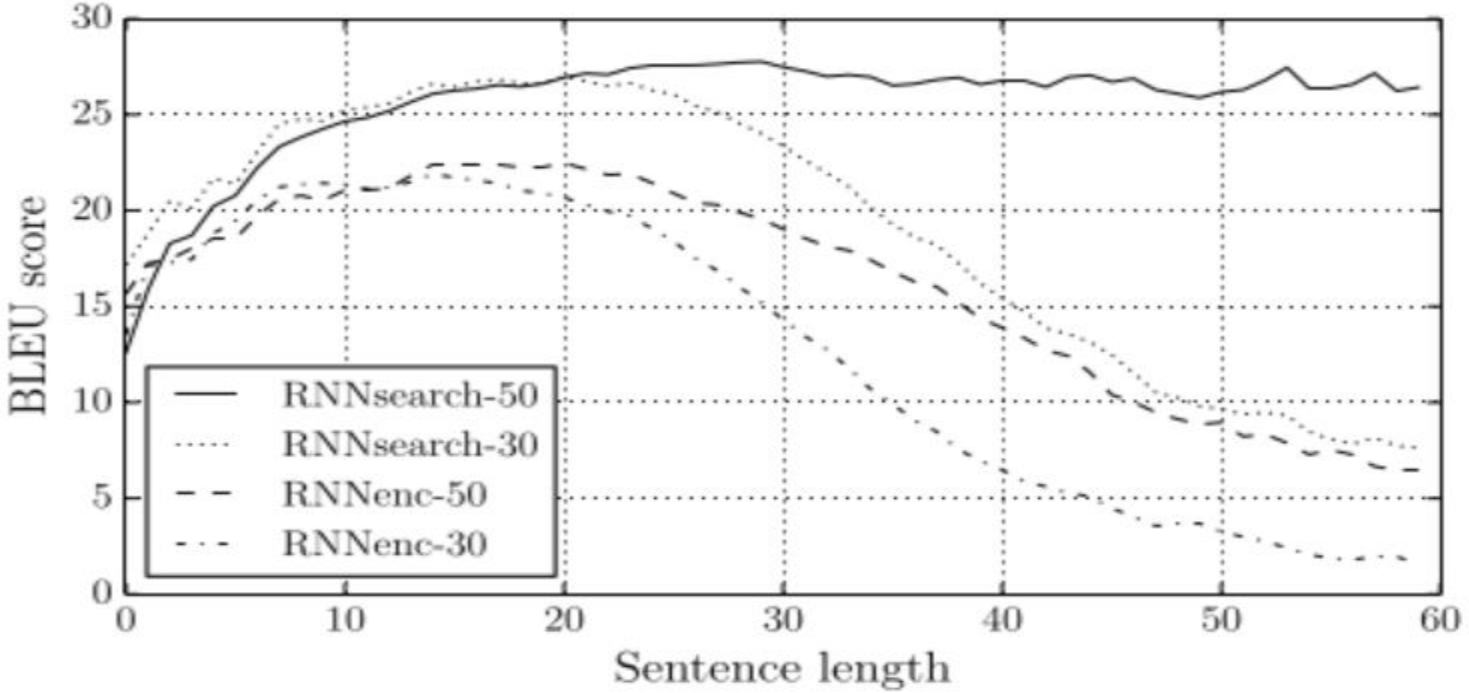
$$\underline{\overrightarrow{h}}_i = \tanh\left(\overrightarrow{W}\overrightarrow{E}x_i + \overrightarrow{U}\left[\overrightarrow{r}_i \circ \overrightarrow{h}_{i-1}\right]\right)$$

$$\overrightarrow{z}_i = \sigma\left(\overrightarrow{W}_z\overrightarrow{E}x_i + \overrightarrow{U}_z\overrightarrow{h}_{i-1}\right)$$

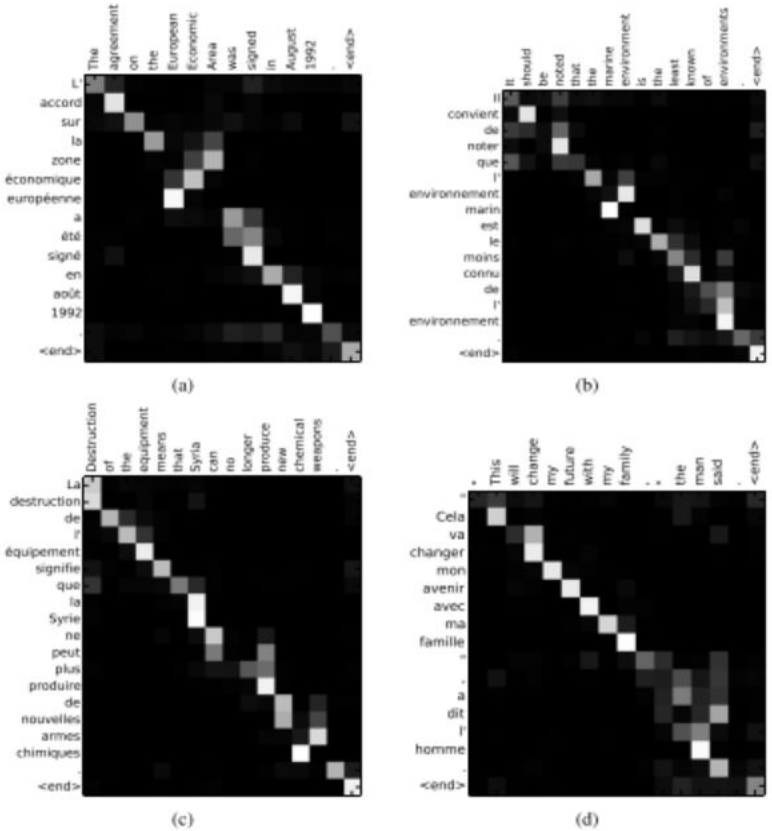
$$\overrightarrow{r}_i = \sigma\left(\overrightarrow{W}_r\overrightarrow{E}x_i + \overrightarrow{U}_r\overrightarrow{h}_{i-1}\right).$$



Neural Machine Translation by Jointly Learning to Align and Translate



Neural Machine Translation by Jointly Learning to Align and Translate



Neural Machine Translation by Jointly Learning to Align and Translate

As an example, consider this source sentence from the test set:

> *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.*

The RNNencdec-50 translated this sentence into:

> *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.*

On the other hand, the RNNsearch-50 generated the following correct translation, preserving the whole meaning of the input sentence without omitting any details:

> *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.*

Memory Networks

- Facebook AI Research
- Question Answering(QA)
- Generalization of many attention mechanism

- Efficient Memory usage Hashing + clustering world embeddings
- Unseen words

Memory Networks

A memory network consists of a memory $\mathbf{m}$ (an array of objects[1] indexed by $\mathbf{m}_i$) and four (potentially learned) components I, G, O and R as follows:

I: (input feature map) – converts the incoming input to the internal feature representation.

G: (generalization) – updates old memories given the new input. We call this generalization as there is an opportunity for the network to compress and generalize its memories at this stage for some intended future use.

O: (output feature map) – produces a new output (in the feature representation space), given the new input and the current memory state.

R: (response) – converts the output into the response format desired. For example, a textual response or an action.

Memory Networks

1. Convert x to an internal feature representation $I(x)$.
2. Update memories $\mathbf{m}_i$ given the new input: $\mathbf{m}_i = G(\mathbf{m}_i, I(x), \mathbf{m}), \ \forall i.$
3. Compute output features o given the new input and the memory: $o = O(I(x), \mathbf{m})$.
4. Finally, decode output features o to give the final response: $r = R(o)$.

Memory Networks

- $$o_1 = O_1(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O(x, \mathbf{m}_i)$$

$$o_2 = O_2(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_i)$$

$$r = \text{argmax}_{w \in W} s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], w)$$

$$s(x, y) = \Phi_x(x)^\top U^\top U \Phi_y(y).$$

Memory Networks

- Modeling write time

$$s_{O_t}(x, y, y') = \Phi_x(x)^\top U_{O_t}{}^\top U_{O_t}\Big(\Phi_y(y) - \Phi_y(y') + \Phi_t(x, y, y')\Big).$$

Φt(x, y, y') uses three new features which take on the value 0 or 1: **whether x is older than y, x is older than y', and y older than y' .** (That is, we extended the dimensionality of all the Φ embeddings by 3, and set these three dimensions to zero when not used.) Now, if sOt (x, y, y') > 0 the model prefers y over y' , and if sOt (x, y, y') < 0 it prefers y'. Then take argmax

Memory Networks

- X = Where is the milk now?
- Mo1 = Joe left the milk
- Mo2 = Joe traveled to the office

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

Memory Networks

- Large-Scale QA
- Simulated World QA

Table 3: Test accuracy on the simulation QA task.

	Difficulty 1			Difficulty 5	
Method	actor w/o before	actor	actor+object	actor	actor+object
RNN	100%	60.9%	27.9%	23.8%	17.8%
LSTM	100%	64.8%	49.1%	35.2%	29.0%
MemNN $k = 1$	97.8%	31.0%	24.0%	21.9%	18.5%
MemNN $k = 1$ (+time)	99.9%	60.2%	42.5%	60.8%	44.4%
MemNN $k = 2$ (+time)	100%	100%	100%	100%	99.9%

Memory Networks

- Large-Scale QA

Bilbo travelled to the cave. Gollum dropped the ring there. Bilbo took the ring.
Bilbo went back to the Shire. Bilbo left the ring there. Frodo got the ring.
Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died.
Frodo went back to the Shire. Bilbo travelled to the Grey-havens. The End.
Where is the ring? A: Mount-Doom
Where is Bilbo now? A: Grey-havens
Where is Frodo now? A: Shire

Memory Networks

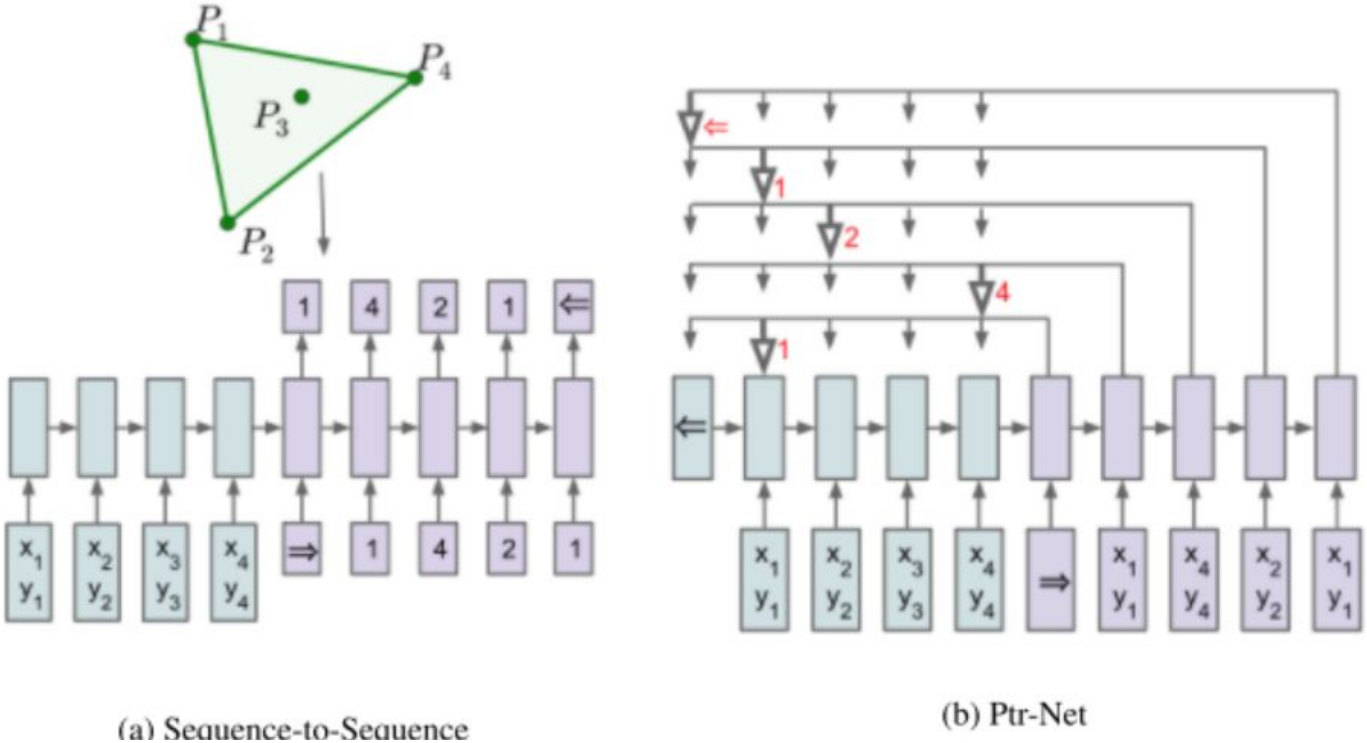
- Large-Scale QA + simulated world

Fred went to the kitchen. Fred picked up the milk. Fred travelled to the office.
Where is the milk ? A: office
Where does milk come from ? A: milk come from cow
What is a cow a type of ? A: cow be female of cattle
Where are cattle found ? A: cattle farm become widespread in brazil
What does milk taste like ? A: milk taste like milk
What does milk go well with ? A: milk go with coffee
Where was Fred before the office ? A: kitchen

Pointer Networks

- Solved problem of variable size output dictionaries
- Combinatorial optimization problems
- Convex hull, Delaunay triangulations, TSP

Pointer Networks



(a) Sequence-to-Sequence

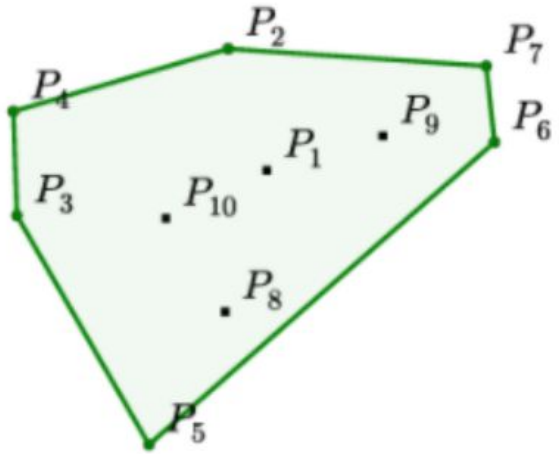
(b) Ptr-Net

Pointer Networks

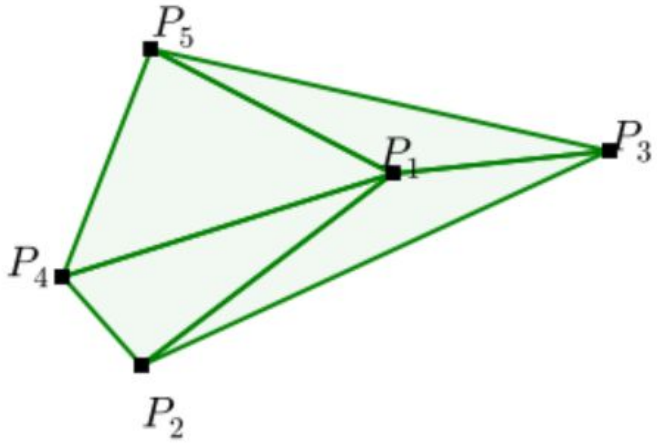
$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
a_j^i &= \text{softmax}(u_j^i) \quad j \in (1, \ldots, n) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) &= \text{softmax}(u^i)
\end{aligned}
$$

Pointer Networks

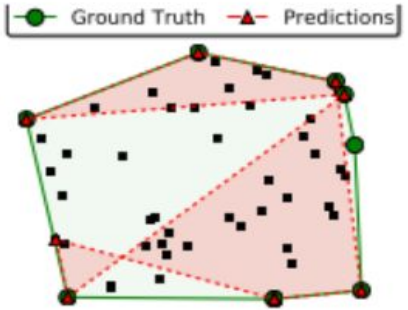


(a) Input $\mathcal{P} = \{P_1, \ldots, P_{10}\}$, and the output sequence $\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, 2, 4, 3, 5, 6, 7, 2, \Leftarrow\}$ representing its convex hull.

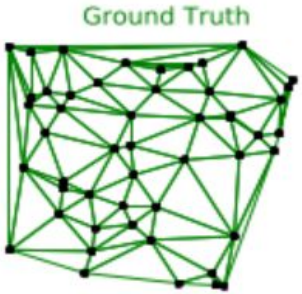


(b) Input $\mathcal{P} = \{P_1, \ldots, P_5\}$, and the output $\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, (1, 2, 4), (1, 4, 5), (1, 3, 5), (1, 2, 3), \Leftarrow\}$ representing its Delaunay Triangulation.

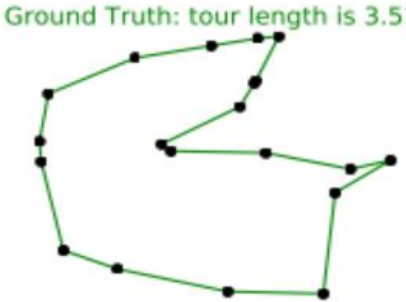
Pointer Networks



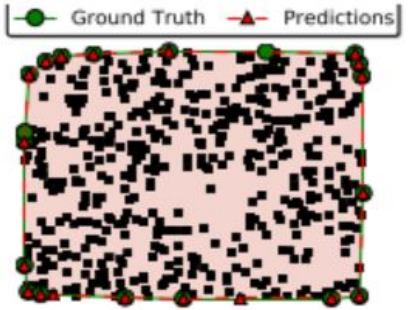
(a) LSTM, m=50, n=50



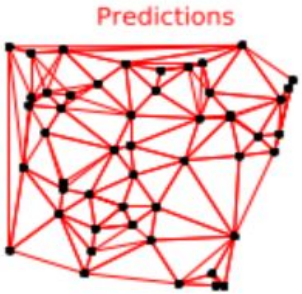
(b) Truth, n=50



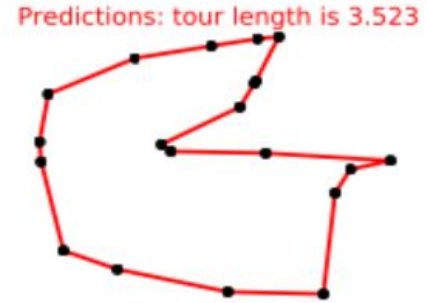
(c) Truth, n=20



(d) Ptr-Net, m=5-50, n=500



(e) Ptr-Net , m=50, n=50



(f) Ptr-Net , m=5-20, n=20

Pointer Networks

- TSP problem

METHOD	TRAINED n	n	ACCURACY	AREA
LSTM [1]	50	50	1.9%	FAIL
+ATTENTION [5]	50	50	38.9%	99.7%
PTR-NET	50	50	72.6%	99.9%
LSTM [1]	5	5	87.7%	99.6%
PTR-NET	5-50	5	92.0%	99.6%
LSTM [1]	10	10	29.9%	FAIL
PTR-NET	5-50	10	87.0%	99.8%
PTR-NET	5-50	50	69.6%	99.9%
PTR-NET	5-50	100	50.3%	99.9%
PTR-NET	5-50	200	22.1%	99.9%
PTR-NET	5-50	500	1.3%	99.2%

Thank you