Recurrent Neural Network

Previous models' prediction

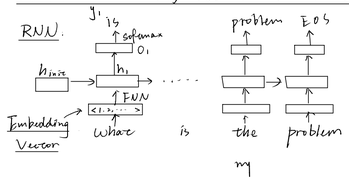
$\{x_i, y_i\}_{i=1}^{N} \rightarrow P(y_i \mid x_i)$ ⇒ 1 input → 1 output.

RNN → Sequential Data

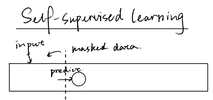
$\{(x_i^{(1)} \dots x_i^{(\tau_i)}), (y_i^{(1)} \dots y_i^{(\tau)})\}_{i=1}^{N} \rightarrow P(y^{(\tau)} \mid (x^{(1)} \dots x^{(\tau-1)}))$

$P(w_k \mid w_1, w_2 \dots w_{k-1}) = P(w_1) P(w_2 \mid w_1) P(w_3 \mid w_1, w_2) \cdots$

RNN.



Self-supervised learning



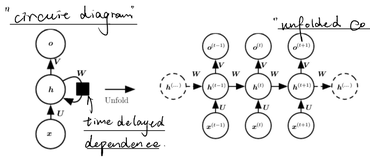
"Training Data is automatically labelled"

in RNN: "Masking"

y_i: the next word after x_{i-1}

$x_i \rightarrow O_i \Leftrightarrow y_i \rightarrow$ loss, BP ...

"circuit diagram" "unfolded (o rotational graph"



$$\begin{cases} h_t = \tanh(W h_{t-1} + U x_t + b) \\ O_t = \text{softmax}(V h_t + c) \end{cases}$$ Bias

x_t: input token (embedding vectors)

word → one hot encoding →

dog	cat	duck
1	0	0
0	1	0
0	0	1

→ FNN → Learnable parameters → Embedding Vector

$L = E[\Sigma_t -\log P(y_t \mid x_1 \dots x_t)]$

Objective: minimize $L(W, U, V, b, c, \theta_{embedding})$ → BPTT

BPTT: Back Propagation Through Time

Learning Long Sequences

Problem:

① Vanishing Gradient —Solution→ Change Activation Function / Residual Connection / Batch Normalization

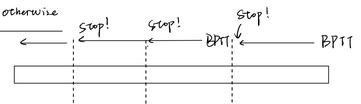
② Exploding Gradient

More serious: weight sharing (W) across different time steps

Gradient Clipping: if exceeding threshold → clips the gradient.

$$g' = \begin{cases} g, & \|g\| \le s \\ \frac{s}{\|g\|} g, & \text{otherwise} \end{cases}$$

Truncated BPTT:



Truncates a long sequence into multiple short subsequence

⇒ Later context cannot have attention to previous context

⇒ Better Approach: replace simple recurrent units by more powerful memory cell

LSTM: Long Short-Term Memory

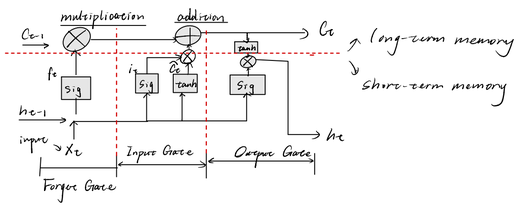
I bought Harry Potter yesterday. Today I want to read it

if truncated BPTT, We can never know what is "it"

LSTM Cell:

$C_t = f_t \otimes C_{t-1} + i_t \otimes \tanh(a_t) \quad h_t = \tanh(b_t)$

LSTM



Forget Gate: $f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$

Input Gate: $i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$

⇒ $C_t = f_t \otimes C_{t-1} + i_t \otimes x_t$

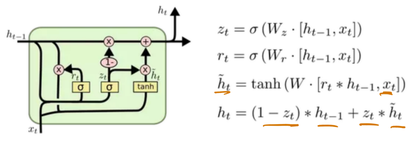
How much to forget — How much to input (add)

Output Gate: $O_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$

$h_t = O_t \otimes \tanh(o_t)$

GRU: Gated Recurrent Unit

similar to LSTM. fewer parameters. more efficient.



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Performance similar to LSTM. except Better on small dataset.

RNN Architecture

① Deep RNNs

- More layers often usually implies better performance.

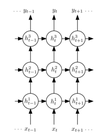
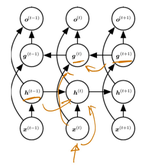


Fig. 3. Deep Recurrent Neural Network

② Bidirectional RNNs.

"I _____ a horse", "I _____ coffee."

We need information from BOTH past and future



③ The Encoder-Decoder Structure

"Seq2Seq" Architecture.

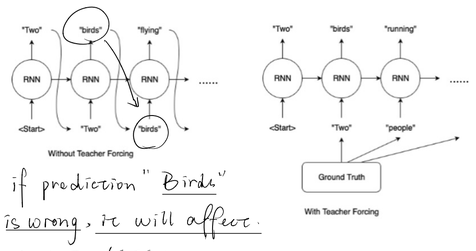
$(x^{(1)} \dots x^{(n_x)}) \rightarrow C \rightarrow (y^{(1)} \dots y^{(n_y)})$

$n_x \neq n_y$, vanilla RNN can only output sequence which has same length with input

C: Context variable ⇒ "Summary".

Used in dialogue system / machine translation, ("Transformer")

Teacher forcing in Decoder



if prediction "Birds" is wrong, it will affect next prediction

Pros: makes training faster

if we don't use it. errors will accumulate

Cons: Discrepancy Between training and inference

"Since when inference. this is no ground truth!"

"Exposure Bias"

Recent work has shown exposure bias is not a serious issue

Decoding:

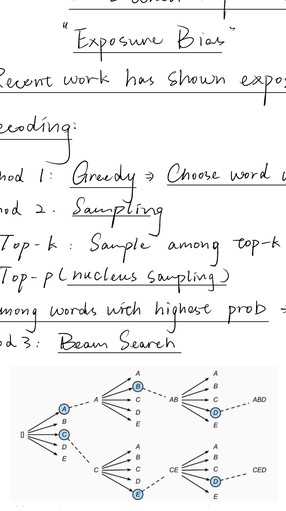
Method 1: Greedy ⇒ Choose word with max prob

Method 2. Sampling

1. Top-k: Sample among top-k words.
2. Top-p (nucleus sampling)

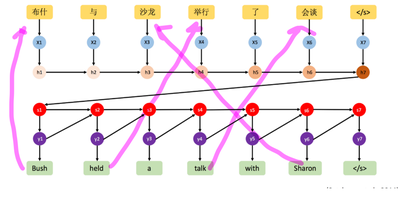
among words with highest prob ⇒ till Sum(prob) ≥ p

Method 3: Beam Search



5. Attention

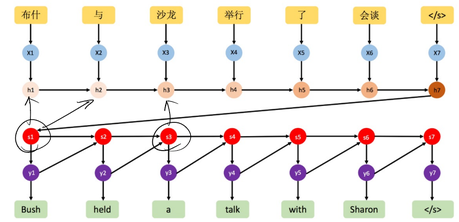
"A mechanism to allow model focus on different parts of the input sequence when generating each output token"



In vanilla Seq2Seq: Context variable C → token 1, token 2, ⋮

"We're using same Summary when generating at All time Step"

⇒ Not desirable



(Sutskever et al., 2014)

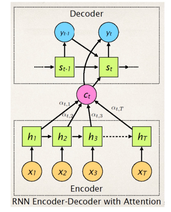
In fact, We want to focus on different parts of input at different time

- Focus at $\mathbf{h}_1$ at time step 1,
- Focus at $\mathbf{h}_4$ at time step 2,
- Focus at $\mathbf{h}_6$ at time step 4,
- ...

∴ C_t: Context variable at time step t

"Weight"

$C_t = \Sigma_j a_{t,j} h_j$

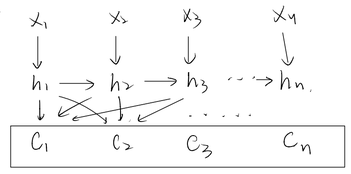


At different step. We use different value of α

"What we want to see / What we don't want to see"

$C_1 = \underbrace{0.8 h_1}_{\text{What we want to see}} + 0.1 h_2 + \cdots + \underbrace{0.001 h_7}_{\text{What we don't want to see}}$

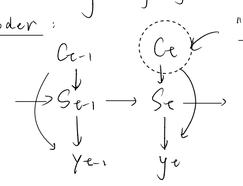
∴ Encoder



Context variables

"$C_t = \Sigma_j a_{t,j} h_j$"

Decoder:



"What do I need at time t" ("Query")

"What is available at h_j?" ("key")

$C_t = \Sigma_j \alpha_{t,j} h_j$

attention paid to h_j at time t

$\alpha_{t,j} = \frac{e^{w_{t,j}}}{\Sigma_j e^{w_{t,j}}}$ (softmax)

where $w_{t,j} = C_{t-1}^T \cdot h_j$, ("Query" * "key")

Looking for (C_{t-1}) → "How much we want it"

what we have: h_j

"dynamic Weight" ⇒ Depends on input

Attention Example: