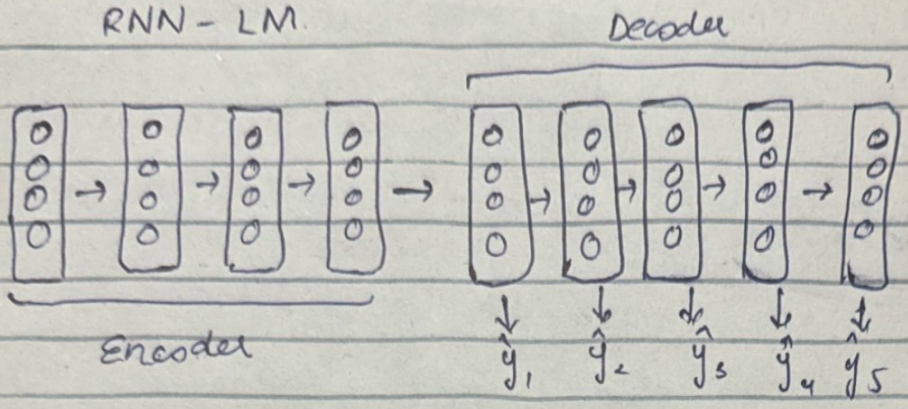
Finals

Natural Language Generation. (NLG)

↳ generation of new text

RNN - LM



For loss : $J_1 = -\log \hat{y}_1$

$J_2 = -\log \hat{y}_2$ & so on

$$\text{Average loss} = \frac{1}{N} \sum J$$

$$= \frac{1}{5}\left[-\log \hat{y}_1 - \log \hat{y}_2 - \log \hat{y}_3 - \log \hat{y}_4 - \log \hat{y}_5\right]$$

Decoder Algorithms for generating text

↳ Greedy Decoding

↳ Beam Search

↳ Sampling Based

Greedy Decoding

↳ most probable word (argmax)

↳ Output of one word is input of next until <END>

↳ Poor output due to lack of backtracking.

(Ungrammatical, Unnatural, Nonsensical, Incorrect)

search

Beam Decoding

↳ Finds highest probability sequences not neccessarily the optimal ones.

↳ Tracks multiple possible sequence at once.

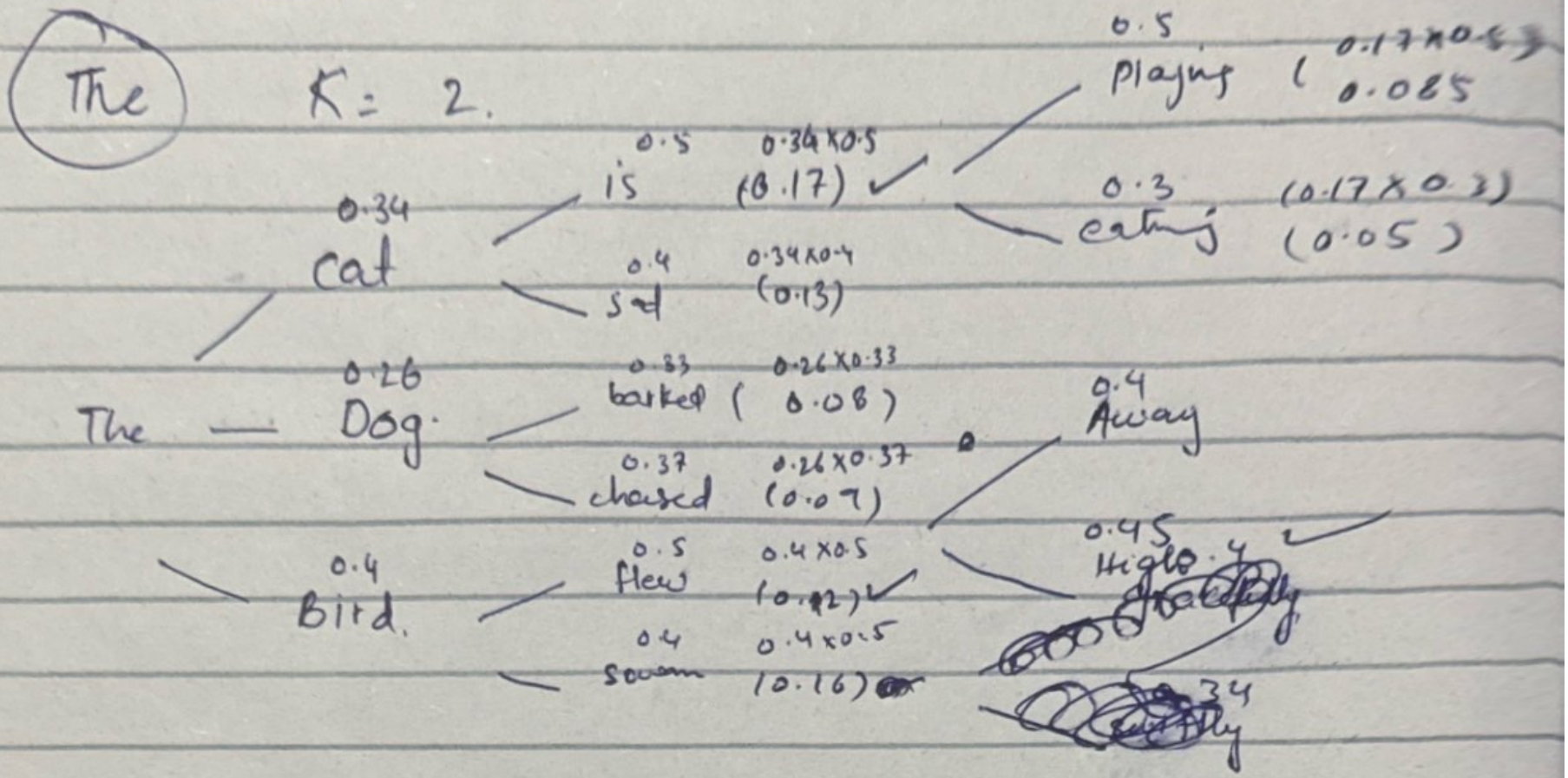
↳ K most probable partial sequences at every step.

↳ K=1 ungrammatical, unnatural insencial, incorrect.

↳ K large reduces some of problems but high computation powee and more generic output and low BLUE score.

NLG

Beam Search



The cat is playing

The bird flew high.

Sampling Based.

↳ Randomness instead of most probable.

↳ Types

① Pure Sampling.

↳ pick randomly.

↳ e.g cat 50%, dog 30%, bird 20%.

↳ you pick dog depsite not being most probable

② Top n sample.

↳ pick any 1 randomly from top n most probable

↳ e.g cat 50%. dog 30%. bird 20%. fish 1

↳ Top 2 = cat dog.

↳ choose dog randomly.

↳ Keep the value of n small for generic & safe outputs

[redacted]

Softmax Temperature (Not decoding algo).

↳ Adjust the probabilities on basis of temp

↳ τ → temp parameter (To control diversity)

$$P_t(w) = \frac{\exp(S_w / \tau)}{\sum_{w'} (\exp S_{w'} / \tau)}$$

↳ Effect of Temp

↳ $\tau > 1$ (high) (flatten)

uniform probabilities, diverse output

↳ $\tau < 1$ (low) (sharpened)

concentrated probabilities, less diverse output

Example:

$S_{cat} = 2$, $S_{dog} = 1$, $S_{bird} = 0.5$

Temperature (T) = 0.5

$$P_t(i) = \frac{e^{i/T}}{\sum e^{i/T}}$$

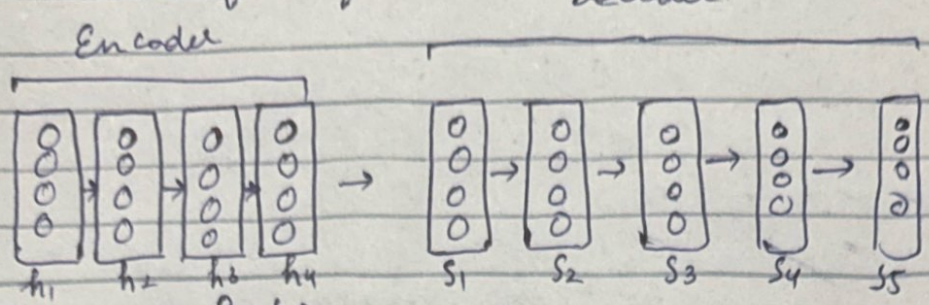
$$P_t(cat) = \frac{e^{2/0.5}}{e^{2/0.5} + e^{1/0.5} + e^{0.5/0.5}} = 0.84$$

$$P_t(dog) = \frac{e^{1/0.5}}{e^{2/0.5} + e^{1/0.5} + e^{0.5/0.5}} = 0.11$$

$$P_t(bird) = \frac{e^{0.5/0.5}}{e^{2/0.5} + e^{1/0.5} + e^{0.5/0.5}} = 0.04$$

Attention in Seq to Seq.

Old Seq 2 Seq



↳ Problem:

Encoder sends all of the information of each hidden state to decoder which causes bottleneck problem.

↳ Solution? Attention (focuses on particular parts of encoder)

Seq to Seq with Attention

① Take dot product of all hidden layers with the first decoder state to get attention score.

$$e_t = [\, s_t h_{t_i}, \ldots \,]$$

② Take softmax of attention scores to get attention distribution.

$$\alpha_t = \text{softmax}(e_t)$$

(3) Take the weighted sum of this attention distribution to get the attention output.

$$a_t = \sum \alpha_t \cdot h_i$$

(4) concatenate the decoder state being used with attention output to get the new encoder state.

$$\begin{bmatrix} a_t \\ s_t \end{bmatrix}$$

(5) Do it for all the decoder hidden layers.

Ways of finding attention score.

- Basic Dot Product
- Multiplicative
- Additive.

Benefits

- improves NMT performance
- Solves bottleneck problem
- helps with vanishing gradient
- provides interpretability
- soft allignment.

* query attends to values.

A way to obtain a fixed size representation of arbitrary set of representaions dependent on some other representations

Example Question:

$h_1 = [0.1, 0.2, 0.3, 0.4]$

$h_2 = [0.2, 0.3, 0.4, 0.5]$

$h_3 = [0.3, 0.4, 0.5, 0.6]$

$S_t = [0.4, 0.5, 0.6, 0.7]$

① Attention Scores

$e_{h_i}^t = S_t \cdot h_{t_i}$

$e_{h_1} = [0.4, 0.5, 0.6, 0.7][0.1, 0.2, 0.3, 0.4]$

$= 0.04 + 0.1 + 0.18 + 0.28$

$= 0.6$

$e_{h_2} = [0.4, 0.5, 0.6, 0.7][0.2, 0.3, 0.4, 0.5]$

$= 0.82.$

$e_{h_3} = [0.4, 0.5, 0.6, 0.7][0.3, 0.4, 0.5, 0.6]$

$= 1.04$

$e^t = [0.6, 0.82, 1.04]$

② Attention Distribution

$$\alpha^t = \text{Softmax}(e^t)$$

$$\alpha_{h1} = \frac{e^{0.6}}{e^{0.6} + e^{0.82} + e^{1.04}} = 0.26$$

$$\alpha_{h2} = \frac{e^{0.82}}{e^{0.6} + e^{0.82} + e^{1.04}} = 0.32$$

$$\alpha_{h3} = \frac{e^{1.04}}{e^{0.6} + e^{0.82} + e^{1.04}} = 0.41$$

$$\alpha^t = [0.26, \ 0.32, \ 0.41]$$

③ Attention Output

$$a^t = \sum \alpha_i^t h_i$$

$$a^t = 0.26 \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix} + 0.32 \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.5 \end{bmatrix} + 0.41 \begin{bmatrix} 0.3 \\ 0.4 \\ 0.5 \\ 0.6 \end{bmatrix}$$

$$= \begin{bmatrix} 0.026 \\ 0.052 \\ 0.078 \\ 0.104 \end{bmatrix} + \begin{bmatrix} 0.064 \\ 0.096 \\ 0.128 \\ 0.16 \end{bmatrix} + \begin{bmatrix} 0.123 \\ 0.164 \\ 0.205 \\ 0.246 \end{bmatrix}$$

$$= \begin{bmatrix} 0.213 \\ 0.312 \\ 0.411 \\ 0.51 \end{bmatrix}$$

$$a^t = [0.213, \ 0.312, \ 0.411, \ 0.51]$$

④ New Encoder State

$[a_t][S_t]$

$= [[0.213, 0.312, 0.411, 0.51], [0.4, 0.5, 0.6, 0.7]]$

⇒ Attention Score Calculation using Multiplicative Attention

Let $W = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.5 & 0.6 \\ 0.4 & 0.5 & 0.6 & 0.7 \end{bmatrix}$

$e_i = S^T W h_i$

$Wh_1 = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.5 & 0.6 \\ 0.4 & 0.5 & 0.6 & 0.7 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.5 \\ 0.6 \end{bmatrix}$

$Wh_2 = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.5 & 0.6 \\ 0.4 & 0.5 & 0.6 & 0.7 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.5 \\ 0.6 \\ 0.7 \end{bmatrix}$

$wh_3 = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.5 & 0.6 \\ 0.4 & 0.5 & 0.6 & 0.7 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.6 \\ 0.7 \\ 0.8 \end{bmatrix}$

$e_1 = S_t \, Wh_1$

$= [0.4, 0.5, 0.6, 0.7] \cdot [0.3, 0.4, 0.5, 0.6]$

$= 1.04$

$e_2 = 1.26$

$e_3 = 1.48$

$e^t = [1.04, 1.26, 1.48]$

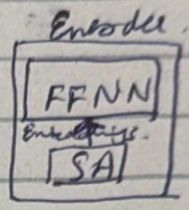
Transformers

(Encoder → Decoder)

↑ Input ↓ Output

↳ Encoder (last hidden state's)'s output is the input of Decoder's (first hidden states)

↳ Sublayers of Encoder
- Feed Forward Neural Network
- Self Attention



↳ Sublayers of Decoder.
- Feed Forward NN
- Encoder Decoder Attention
- Self Attention.

↳ Path of one encoder layer.

Input → SA → Embeddings → FFNN → Output

Question

$X_1 = [0.1 \quad 0.2 \quad 0.3 \quad 0.4]$

$X_2 = [0.5 \quad 0.6 \quad 0.7 \quad 0.8]$

$X_3 = [0.9 \quad 1.0 \quad 1.1 \quad 1.2]$

$$W_Q = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.5 & 0.6 \\ 0.4 & 0.5 & 0.6 & 0.7 \end{bmatrix}$$

$$W_K = \begin{bmatrix} 0.4 & 0.3 & 0.2 & 0.1 \\ 0.5 & 0.4 & 0.3 & 0.2 \\ 0.6 & 0.5 & 0.4 & 0.3 \\ 0.7 & 0.6 & 0.5 & 0.4 \end{bmatrix}$$

$$W_V = \begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.7 \\ 0.2 & 0.4 & 0.6 & 0.8 \\ 0.3 & 0.5 & 0.7 & 0.9 \\ 0.4 & 0.6 & 0.8 & 1.0 \end{bmatrix}$$

① $Q_1 = [0.1 \quad 0.2 \quad 0.3 \quad 0.4] \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.5 & 0.6 \\ 0.4 & 0.5 & 0.6 & 0.7 \end{bmatrix}$

$= [0.3 \quad 0.4 \quad 0.5 \quad 0.61]$

$K_1 = [0.1 \quad 0.2 \quad 0.3 \quad 0.4] \begin{bmatrix} 0.4 & 0.3 & 0.2 & 0.1 \\ 0.5 & 0.4 & 0.3 & 0.2 \\ 0.6 & 0.5 & 0.4 & 0.3 \\ 0.7 & 0.6 & 0.5 & 0.4 \end{bmatrix}$

$= [0.6 \quad 0.5 \quad 0.4 \quad 0.3]$

$V_1 = [0.1 \quad 0.2 \quad 0.3 \quad 0.4] \begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.7 \\ 0.2 & 0.4 & 0.6 & 0.8 \\ 0.3 & 0.5 & 0.7 & 0.9 \\ 0.4 & 0.6 & 0.8 & 1.0 \end{bmatrix}$

$= [0.3 \quad 0.5 \quad 0.7 \quad 0.9]$

$K_2 = [0.5 \quad 0.6 \quad 0.7 \quad 0.8] \begin{bmatrix} 0.4 & 0.3 & 0.2 & 0.1 \\ 0.5 & 0.4 & 0.3 & 0.2 \\ 0.6 & 0.5 & 0.4 & 0.3 \\ 0.7 & 0.6 & 0.5 & 0.4 \end{bmatrix}$

$= [1.48 \quad 1.22 \quad 0.96 \quad 0.7]$

$K_3 = [0.9 \quad 1.0 \quad 1.1 \quad 1.2] \begin{bmatrix} 0.4 & 0.3 & 0.2 & 0.1 \\ 0.5 & 0.4 & 0.3 & 0.2 \\ 0.6 & 0.5 & 0.4 & 0.3 \\ 0.7 & 0.6 & 0.5 & 0.4 \end{bmatrix}$

$= [2.36 \quad 1.94 \quad 1.52 \quad 1.1]$

② Scores

$S_1 = Q_1 \cdot K_1$

$= [0.3\ 0.4\ 0.5\ 0.6] \cdot [0.6\ 0.5\ 0.4\ 0.3]$

$= 0.18 + 0.2 + 0.2 + 0.18$

$= 0.76$

$S_1 = Q_1 \cdot K_2$

$= [0.3\ 0.4\ 0.5\ 0.6][1.48\ 1.22\ 0.96\ 0.7]$

$= 1.85$

$S_3 = Q_1 \cdot K_3$

$= [0.3\ 0.4\ 0.5\ 0.6][2.36\ 1.94\ 1.52\ 1.1]$

$= 2.9$

③ Div by $\sqrt{d_K}$, Let $\sqrt{d_K} = 8$

$S_{1new} = 0.76 / 8 = 0.095$

$S_{2new} = 1.85 / 8 = 0.231$

$S_{3new} = 2.9 / 8 = 0.362$

④ Softmax for S_{1new}

$$S_{1\text{-}softmax} = \left(\frac{e^{0.095}}{e^{0.095} + e^{0.231} + e^{0.362}} \right)$$

$= 0.29$

(5) Now calculating Z_1

$$Z_1 = S_{1\,softmax} \times V_1$$

$$= 0.29 \; [0.3 \quad 0.5 \quad 0.7 \quad 0.9]$$

$$= [0.087 \quad 0.145 \quad 0.203 \quad 0.261]$$

↳ Multi headed.

Contatenate all Z's and multiply with W_o. to get Z.

$$Z = Z_c \cdot W_o.$$

↳ Positional Encoding.

Used for getting diffs probs of results for diff sentences based on positions.