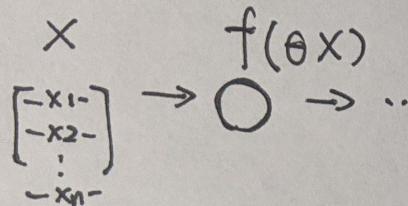


Deep Learning

• Neuron:



X : data matrix

$$\theta = \begin{bmatrix} 1 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix} \text{ Params}$$

θX : linear comb of param * data

f : activation function:

ReLU:

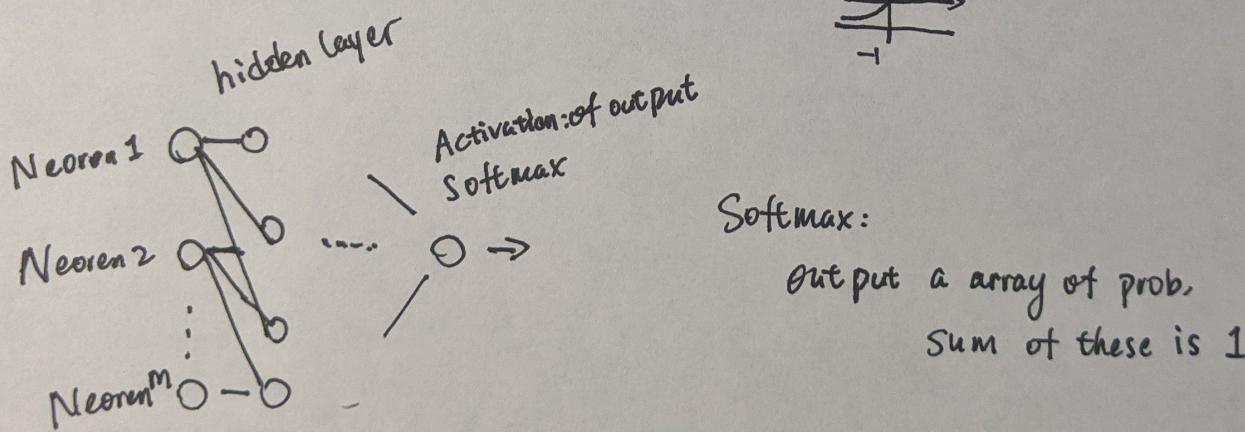
$$y = \max(0, x)$$

Sigmoid:

$$y = \frac{1}{1 + e^{-x}}$$

tanh:

$$y = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

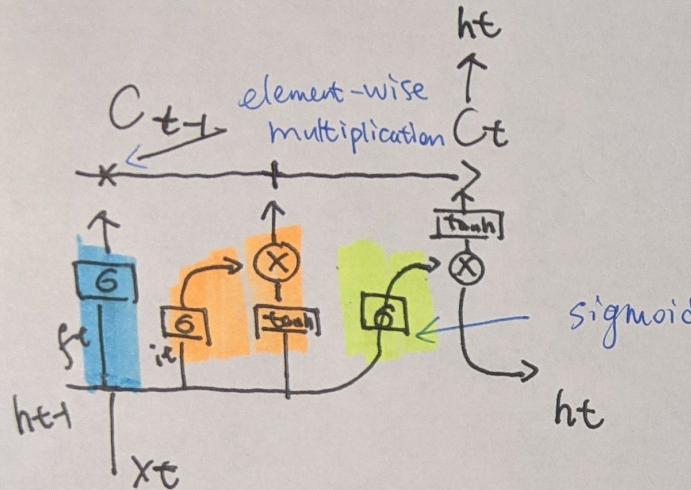


Softmax:

Output a array of prob.
Sum of these is 1

• Backpropagation: update params by gradient descent & chain rule.

LSTM



Computation:

Forget Gate: $\sigma(W_f [h_{t-1}] + b_f)$

Input Gate: $\sigma(W_i [h_{t-1}] + b_i)$

Cell State: $tanh(W_g [h_{t-1}] + b_g)$

Output Gate: $\sigma(W_o [h_{t-1}] + b_o)$

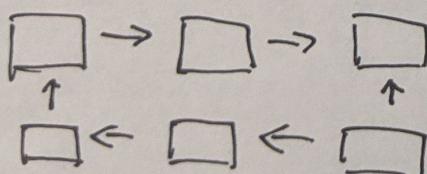
Final Output: $tanh(W_o [h_{t-1}] + b_o) * \sigma(W_f [h_{t-1}] + b_f) * tanh(W_g [h_{t-1}] + b_g) + \sigma(W_i [h_{t-1}] + b_i) * \text{new Value}$

params:

$$4 \times \underbrace{\text{Shape}(h)}_{w_f, w_i, w_c, w_o} \times (\underbrace{\text{Shape}(h) + \text{shape}(x)}_{w[h+x]})$$

Bidirectional LSTM:

forward pass



Backward pass

Word Embedding

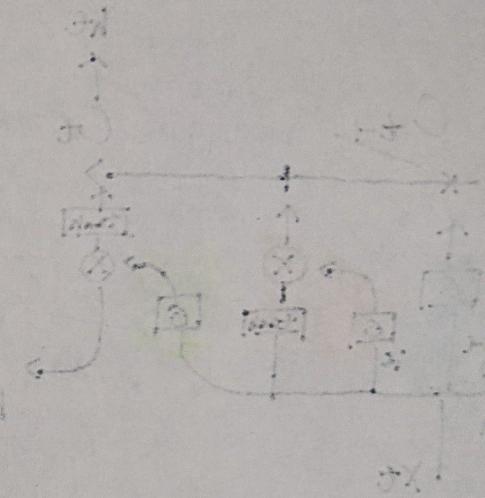
MTBD

$$\begin{array}{c} \text{Word in dict} \\ \boxed{\text{A} \quad \text{B} \quad \text{C} \quad \text{D}} \end{array} \times \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \rightarrow \boxed{\text{A}}$$

one hot encoding

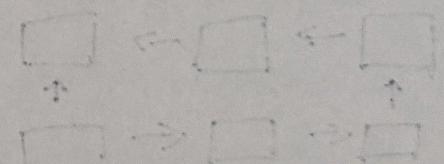
Encoding for one word,

Length is customized.



word embedding

word vector

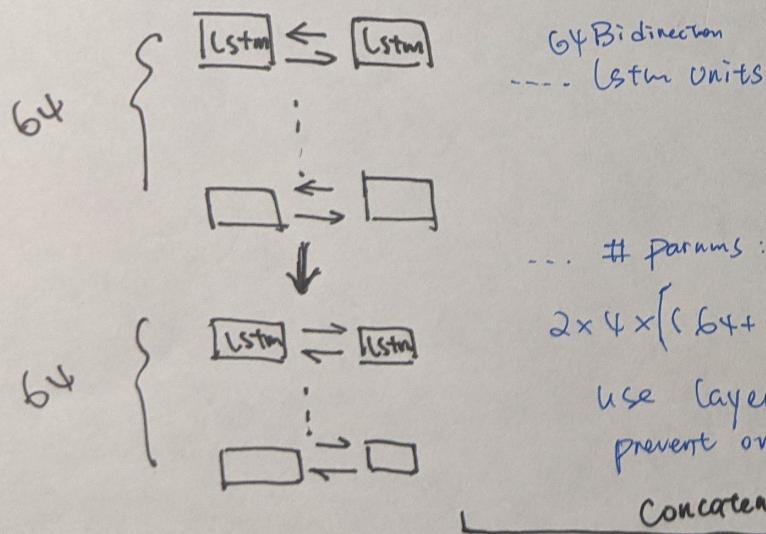
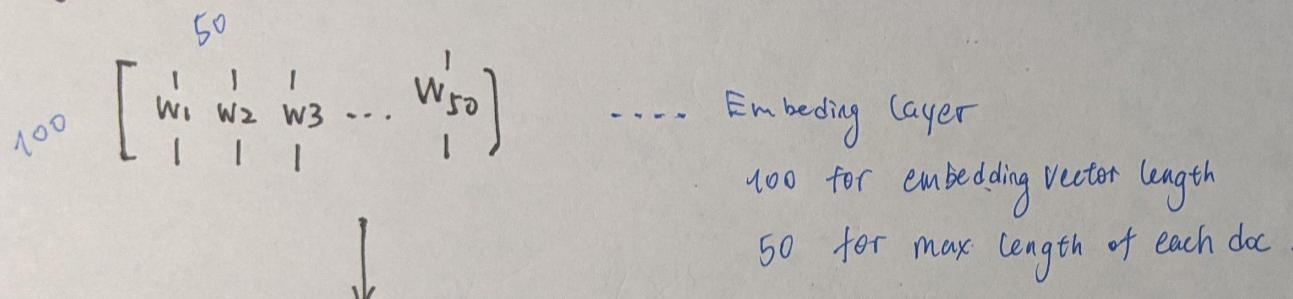


word vector

Model From Internship,

text classification for
cleaning text.

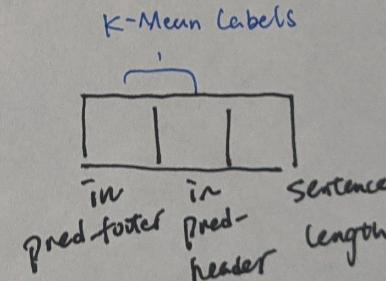
- each piece of doc are sep by " " (triple space),
- 0 as need to keep
- 1 as need to remove
-



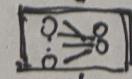
... # Params:

$$2 \times 4 \times ((64+100) \times 64 + 64) \approx 80K$$

use layer regularizer to prevent overfitting.



Concatenate



DNN, ReLU

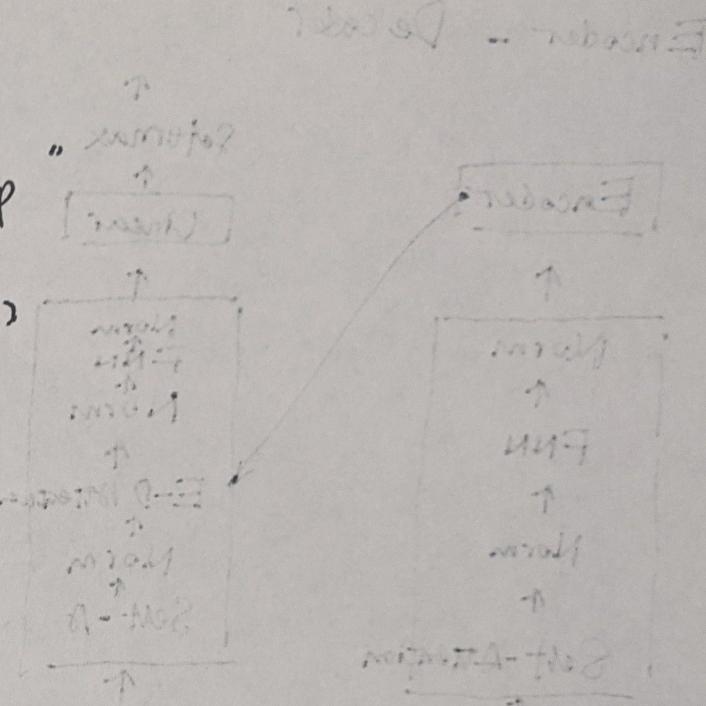
Sigmoid

Cross Entropy Loss

Transformer Networks

Vector-Seg Model :

Dog img \rightarrow "Dog in a cup"

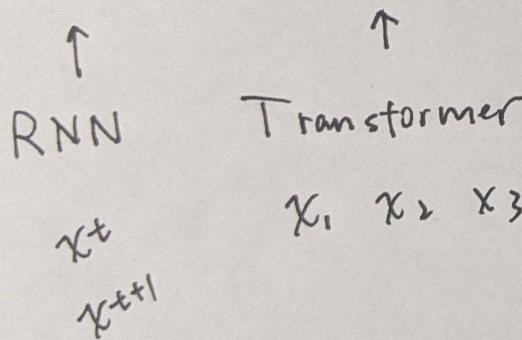


Seq-vec Model :

"Text..." \rightarrow [0~1]

Seq-seq Model :

Translation



-- Positional Encoder:

vector that gives context based on position of word in sentence

Seg. Model

Attention Layer : What part of the input should we focus?

E-C Attention

Enc

x_1	x_2	x_3	\cdots	x_m
-------	-------	-------	----------	-------

$$\rightarrow K_i = W_k x_i \quad \cdots$$

$$\rightarrow V_i = W_v x_i \quad \cdots$$

Structure:

Dec

x'_1	x'_2	\cdots	x'_m
--------	--------	----------	--------

$$\rightarrow g_j = W_a x'_j \quad \cdots$$

Formulas

$$C = \text{Attention}(x, x') \quad \alpha_i = \text{Softmax}(K^T g_i) \in \mathbb{R}^m$$

$$c_i = \alpha_1 v_1 + \cdots + \alpha_m v_m = V \alpha_i$$

$$\alpha_2 = \dots$$

$$\alpha_m = \dots$$

Notes:

C can be used to generate next word (Translation Task)

Self-Attention Layer

$$\text{Query } q_i = W_Q x_i$$

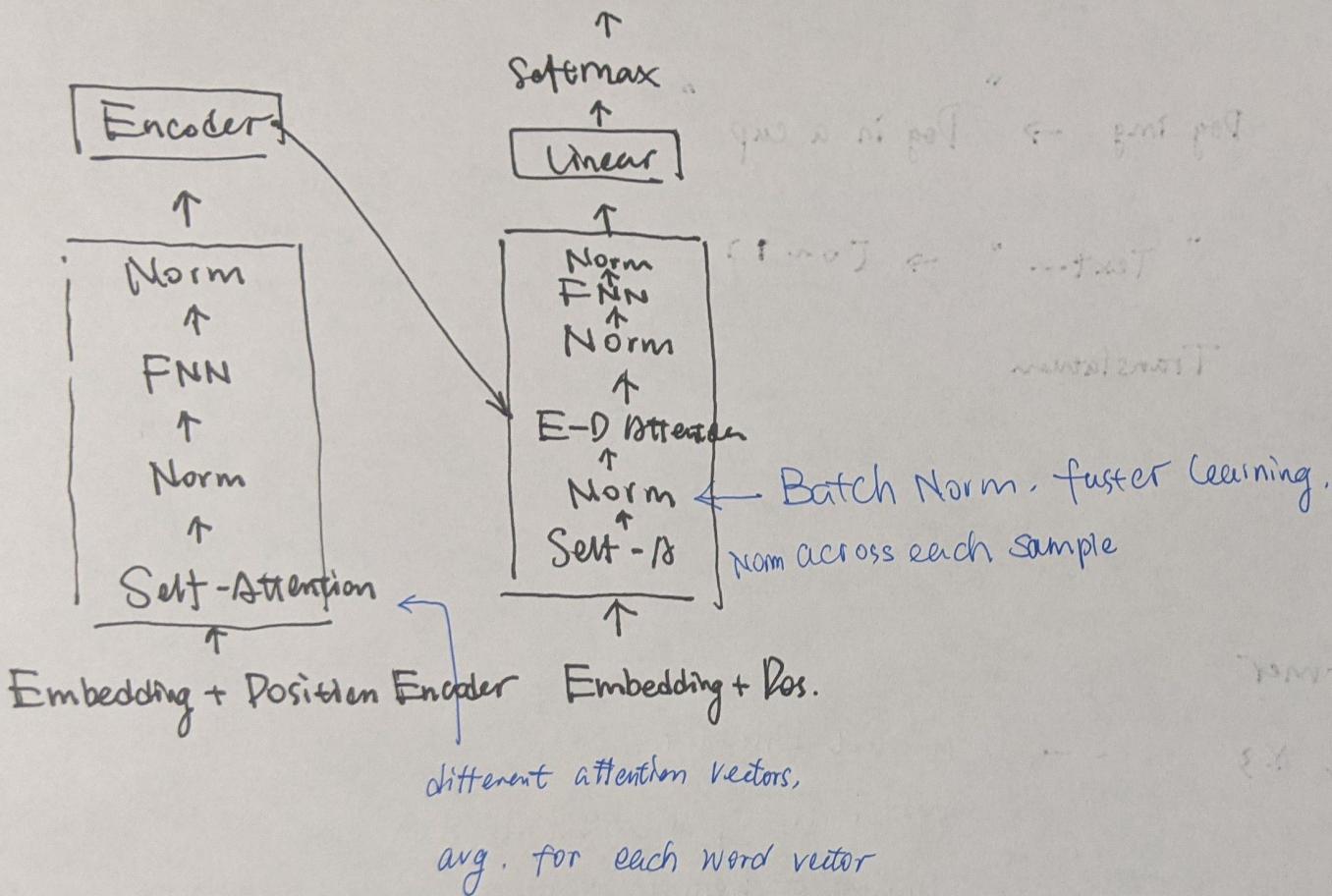
$$\text{Key } k_i = W_K x_i$$

$$\text{Value } v_i = W_V x_i$$

$$\alpha_j = \text{Softmax}(K^T q_i : j) \\ c_1 = \alpha_{11} v_1 + \dots + \alpha_{m1} v_m = V d_2 \quad \left. \right\} \Rightarrow c_j = V \cdot \text{Softmax}(K^T q_i : j)$$

$$\text{Self-attention layer: } C = \text{Attn}(X, X)$$

• Encoder - Decoder



BERT

- is for pre-training Transformer's encoders
- Predict masked words next sentence.

- ① mask 15% words randomly.
output a prob @ masked position.

calculate $\text{Loss} = \text{CrossEntropy}(e, p)$

↑
ground truth ↑
 Prob

Perform GD to update parameters

- ② ~~mask~~ use 50% true linked 2 sentences,
50% faked

[CLS] Sent 1 [SEP] Sent 2



token for classification for separating sentences

- Combing Two Tasks

input

example:

"[CLS] Calculus is a [MASK]
of math,

[SEP] it [MASK] developed
by newton and leibniz."

Target: true, "branch", "was"

LOSS:

next-sent:

binary clf: cross

Entropy

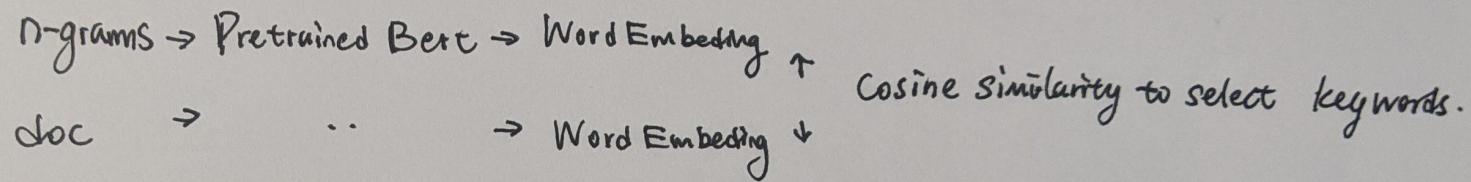
masked word:

multi-class ~~clf~~ ~~prob~~
Classification

Sum of three loss

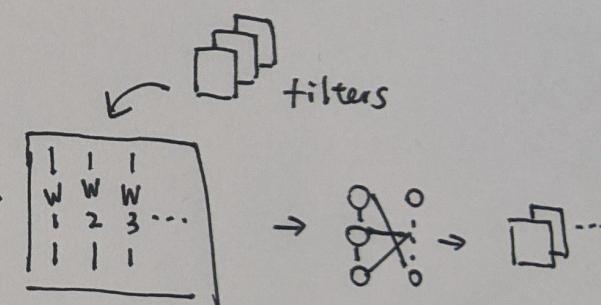
Model From Internship, Keyphrase extraction task

- Unsupervised Keyphrase Extraction:



- Supervised Keyphrase Extraction. (BERT-KPE)

n-grams → Pretrained Bert
(Base/SpanBert/RoBERT)



CNN on word
Embedding

FNN

MaxPool

Best Local Score
for each phrase

Loss :

Hinge Loss : pairwise ranking loss

CE Loss : is or not true keyphrase