Lecture NO. 18

Machine Translation

↓ many-to-many ↓ sequence-to-sequence

Main Problem → ① mapping

↓ ideal (1-to-1)

② Alignment → should be identical

1 RNN used for understanding source language

~~1 RNN~~ ~~without empty~~. 1 RNN is used to generate healthy and safe products with high accuracy.

x = source language

y = target/destination language.

Total RNN's : 2

Encoder RNN → understand your task

Decoder RNN → Translate

Beam search → gives sub-optimal result.

↓ not always optimal.

Lecture No. 19

Seq 2 Seq (Machine Translation)

→ LSTM better than RNN in machine translation as well as it solves vanishing & exploding gradient problem.

•) When using LSTM in encoder, LSTM should also be used in decoder.

•) Weights for LSTM for encoder & decoder are different.

EOS → End of String
SOS → Start of sentence
] marker in the corpus for easy processing etc.

→ Attention Mechanism
(solves the problem in LSTM when used for machine translation).

not used with LSTM.
used with Transformers.

LSTM

Lecture NO. 20

Transformers.

- Problems of Natural Language Model
  - → Positional Encoding (Not knowing position of word)
  - → Fixed Length
  - → Padding

- Problems of RNN
  - → Vanishing Gradient
  - → Exploding Gradient
    - ↳ values are too large

Solved the problem of "Fixed Length" and handle sequences.

- LSTM with Attention → complex problem model
  - → high computational cost

example 5 units, 3 layers, hidden layer = 1 neuron
50 times unroll (example)

10+10+10 ← 30×50 = 1500

bytes to ← 8×1500 = 12000 MB
store

→ storage
→ sequential computation ~~(O^n)~~
O(n)

gives all the input at once
not sequential

Transformers — Encoder
— Decoder

O(1)

Encoder → gets the hidden info from input

→ input
→ embeddings from dense vectors
→ positional encoding (attached) vector

main thing behind the ~~redu~~ decrease in time complexity.

if removed, permutation variance occurs

Positional Encoding (pos, 2i) = sin $\frac{\text{position}}{10000^{\frac{2i}{d_{model}}}}$

for even positions (sin)
" odd " (cos)

→ pos encodings computed only once
→ length of dense vectors → dmodel

final encoder input = dense vector + positional encoding

- Attention can be calculated using dot product
  ↳ for similarity

→ Self-Attention: (One-Head)

Input $\xrightarrow{\text{copy}}$ values (V)

Input → (K) Key

copy → Query (Q)

$$= \text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_{model}}}\right) V$$

↳ size of input

→ Multi-Head Attention:
Add some learning with the help of weights

→ getting Q' (obtained after multiplication with weights)

→ output of encoder will be multi-head attention.

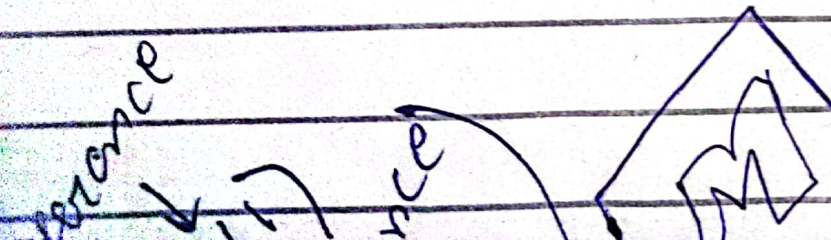
e.g searching a movie

Query → content searching (love)

key → genre (romantic)

value → name of movie (Titanic)

Normalization → layer normalization
↓
-multi-headed



• inference

$\downarrow$

$O(1)$

• in inference

Decoder not $O(1)$

TKP

Lecture NO. 21

Masked Multi-Head Attention

↳ we have prior info but no future info

Lecture NO. 22

→ Why trainning is done in O(1) when softmax is applied (seq × vocab size vector) is formed from which the first ~~imp~~ words can be identified.

As input and output are both matrices so only multiplication is done which can be done in O(1).

- GPT is decoder from transformer
- 12 encoders = BERT
- 12 decoders = GPT

BERT → Bi-directional encoder representation from transformers (only gives the word embeddings)

→ Uses :-

- Feature Extraction / Speech to speech translation (anything related to text).

→ Static embeddings:

Word to Vec, GLOVE, Sparse despite of where & how the word is used.

→ Dynamic embedding:

Embedding that are induced by

contexts are important ~~and is~~
~~is Bi-LSTM used~~.
Bi-LSTM used (go in 2 directions)
- left to right (input)
- right to left (short)

BERT trains in both directions using self-attention (encoder - 2 bi directional form)

BERT (base) → encoder (12 copies)
" (Large) → encoder (24 copies)

→ Segment Embedding:
telling sentence wise according to the document

sentence localizing, word's localization in the text.

→ Pre-trainning and Fine tuning done ~~in parallel~~ simultaneously

→ Mask Language Model: (MLM)
Predict where "mask" is written

→ embeddings changed : pre-trained weights are not used, random weights are used.

→ Tasks to be performed simultaneously by BERT
→ Mask Language Model (increase predictive powers)
→ Next Sentence Predictions

→ Insert a special token 'CLS' (for NSP) to make prediction

→ (CLS returns (Yes | No) "class". At first randomly but after that prediction on the basis of context of the sentence. | Store the crux of image / text

→ We need cross attention in BERT.
→ write question in CLS, get answer after seperation

→ If back propogation is done on all the words, then the "trainning" process will be slow.

Process of Pre-Training:
input → input token → Bert. → 15% Mask → 10% Random → Remaining 10% original words → new sentence → classification task (NSP)

Lecture No. 24

accuracies are very good.

Part of Speech Tagging

↓

classify every word into categories
e.g noun, adverb etc.
Example: apple can be a fruit
or company.

- Categories:

→ Open class
→ closed class

10-15% words are ambigious.

- look at the neighbouring words and their classes and the predict the probability of next class/words (One of the ways for POS Tagging)

Lecture NO. 7

Confusion Matrix

Y → Actual

Y → Predicted
↳ classifier

(Binary classification) Predicted →

		1	0
Actual	1	5 (11, TP)	4 (10, FN)
	0	2 (01, FP)	3 (00, TN)

• will be valid when both the classes are balanced

~~y/+ve~~

- - → to be filled by classifier

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$= \frac{5+3}{5+3+4+2} = \frac{8}{14} = 0.57$$

• Simple accuracy is not good for skewed data that is why we used precision & recall

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$= \frac{5}{7} = 0.71$$

→ actual = true, predicted = true

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{5}{9} = 0.55$$

macro avg => equal weight (simple avg)

weighted avg $= \frac{\sum(\text{support} \times \text{Pre Ans})}{\sum \text{support}}$

F_β

$$F1\ \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$= \frac{2 \times 0.71 \times 0.55}{0.71 + 0.55}$$

$$= 0.63$$

"support" comes from actual data

Multi-class

actual \ predicted	ant	bird	cat
ant	2	0	0
bird	0	0	1
cat	1	0	2

row — FN

column → FP

convert multi-class to binary class for calculation

cat

	1	0
1	2 (11)	1 (10)
0	1 (01)	2 (00)

P = 2/3 = 0.67

R = 2/3 = 0.67

F1 = 0.67

ant

	1	0
1	2 (11)	0 (10)
0	1 (01)	3 (00)

P = 2/3 = 0.67

R = 2/2 = 1

$F1 = 2 \times \frac{0.67 \times 1}{1 + 0.67}$

$= 0.8$

bird

	1	0
1	0 (11)	1 (10)
0	0 (01)	5 (00)

P → 0

R → 0

F1 = 0.