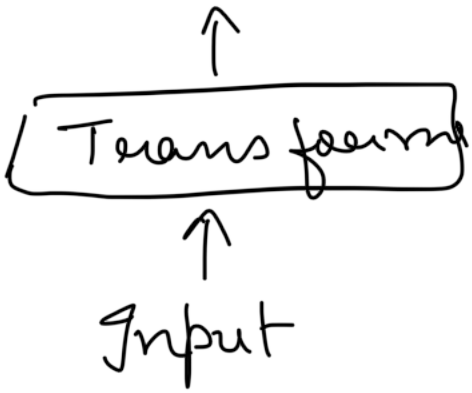
ELM - Week-3

L1 Intro to Language Modelling

- we have learnt about components of transformer architecture



- we can use this transformer to do many NLP tasks. we need to train a dataset for sep. model. And train the models from scratch.
- Parameters are randomly initialized for each task. Thus, taking long time for convergence. It is possible at many times, we do not have enough labelled data for many NLP tasks. Preparing labelled data is costly.
- But, we have many unlabelled

data across internet.

- If you ask some/any lay person based on some statement —

  Task → statement → followed by ques^n

  → but nobody was trained for this ques^n.

  → we develop a strong understanding of lang. w/o any explicit supervision

  → use the same idea.

- Can now my model develop basic understanding of lang. just by exposing to large amount of unlabelled raw text? (pre-training)
- How will it perform on down stream tasks with min. supervision? (supervised fine-tuning)
- this 2-stage thing increases 92% accuracy.

L2 Language Modelling

- pre-training ≡ lang. modeling
- V ⇒ vocab of lang. (unique words)

sentence as a seq. $X_1 \dots X_n$, $X_i \in V$

always there are some sentences which are more probable than others. We expect those sentences to appear more frequently than other sentences.

$f : (X_1 \dots X_n) \to [0, 1]$

↳ this func^n is a lang. model

$$P(x_1, x_2 \dots x_T) = P(x_1)\ P(x_1 \mid x_2) \dots$$

$$= \prod_{i=1}^{T} P(x_i \mid x_1 \dots x_{i-1})$$

if they are independent,

$$P(x_1 \dots x_T) = \prod_{i=1}^{T} P(x_i)$$

- How do we make the model learn the lang.? Make it to learn to predict the next token in a seq. If this idea works, then we have huge amount of such sequences to give model & ask it to predict $(k+1)^{th}$ word.

- This task of predicting the next token in a sequence is c/d lang. modelling.
- But the words aren't indep. They do depend on previous words.

$$\prod_{i=1}^{T} P(x_i \mid x_1 \dots x_{i-1})$$

it is the conditional distribn over the vocabulary.

L3 Transformers for Language Modelling

- How do we estimate these above conditional probabilities?
- Use autoregressive models where conditional probab. are given by parameterized funcns with a fixed no. of params like transformers.
- CLM (Causal language model)

$$P(x_i \mid x_1 \dots x_{i-1}) = f_\theta(x_i \mid x_1 \dots x_{i-1})$$

we are looking for this f_θ.

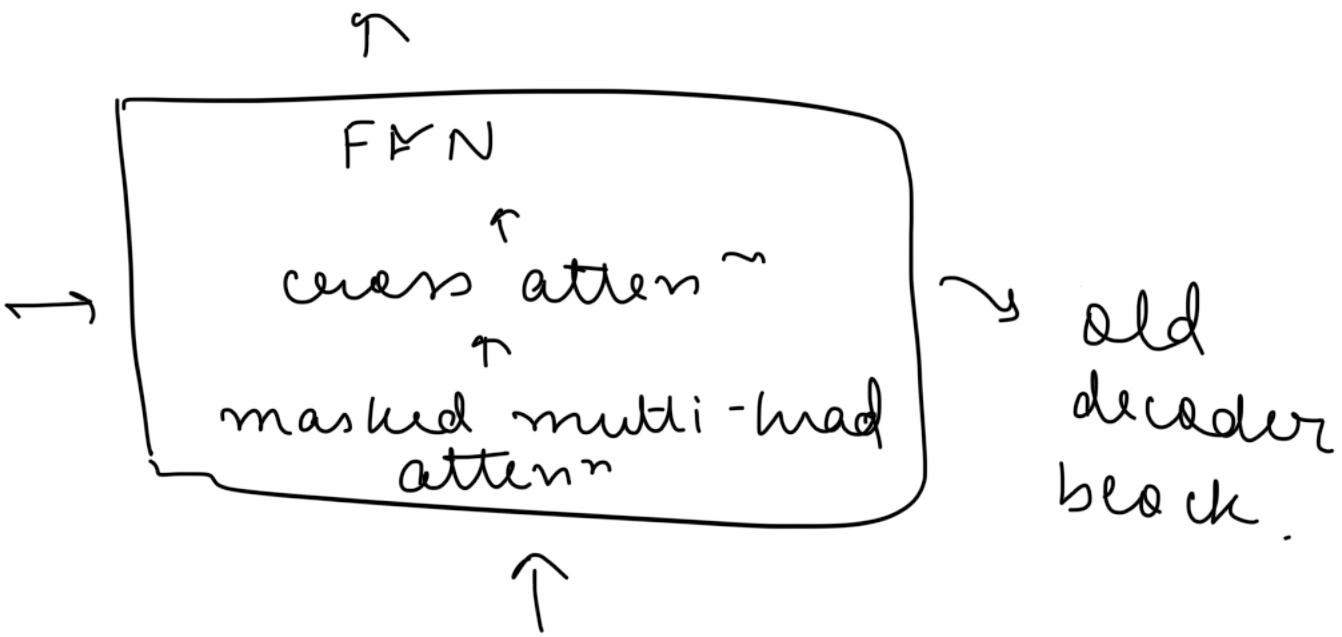
- Can this fo be a transformer ?

Yes.

- Possibilities →
  1. encoder only → Bert
  2. decoder only → GPT
  3. encoder - decoder models. → T5

L4 Causal Language Modelling

- we are looking into decoder only models.



- input is a seq. of words. We want the model to see only the present & past inputs. We can achieve this by applying the task.

$$M = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

↳ this is req.

But cross attenⁿ is not req. as it was used in enc. dec. mod.

$p(x_1)$ ------ $p(x_4 | x_1, x_2, x_3)$

$\langle go \rangle$ x_1 x_2 x_3

output represent each term in the chain rule. Probab. are calculated with the help of parameters of model. The objective should be to maximize the likelihood $L(\theta)$.

LS Generative Pre-Trained Transformer

X is the input seq.

$$h_0 = X \in R^{T \times d_{model}}$$

$$h_l = \text{transformer}(h_{l-1}), \quad \forall l \in [1, n]$$

stack of such transforms (n)

$$P(x_i) = \text{softmax}(h_n[i] W_e)$$

$$L = \sum_{i=1}^{T} \log\left(P(x_i \mid x_1, \ldots x_n)\right)$$

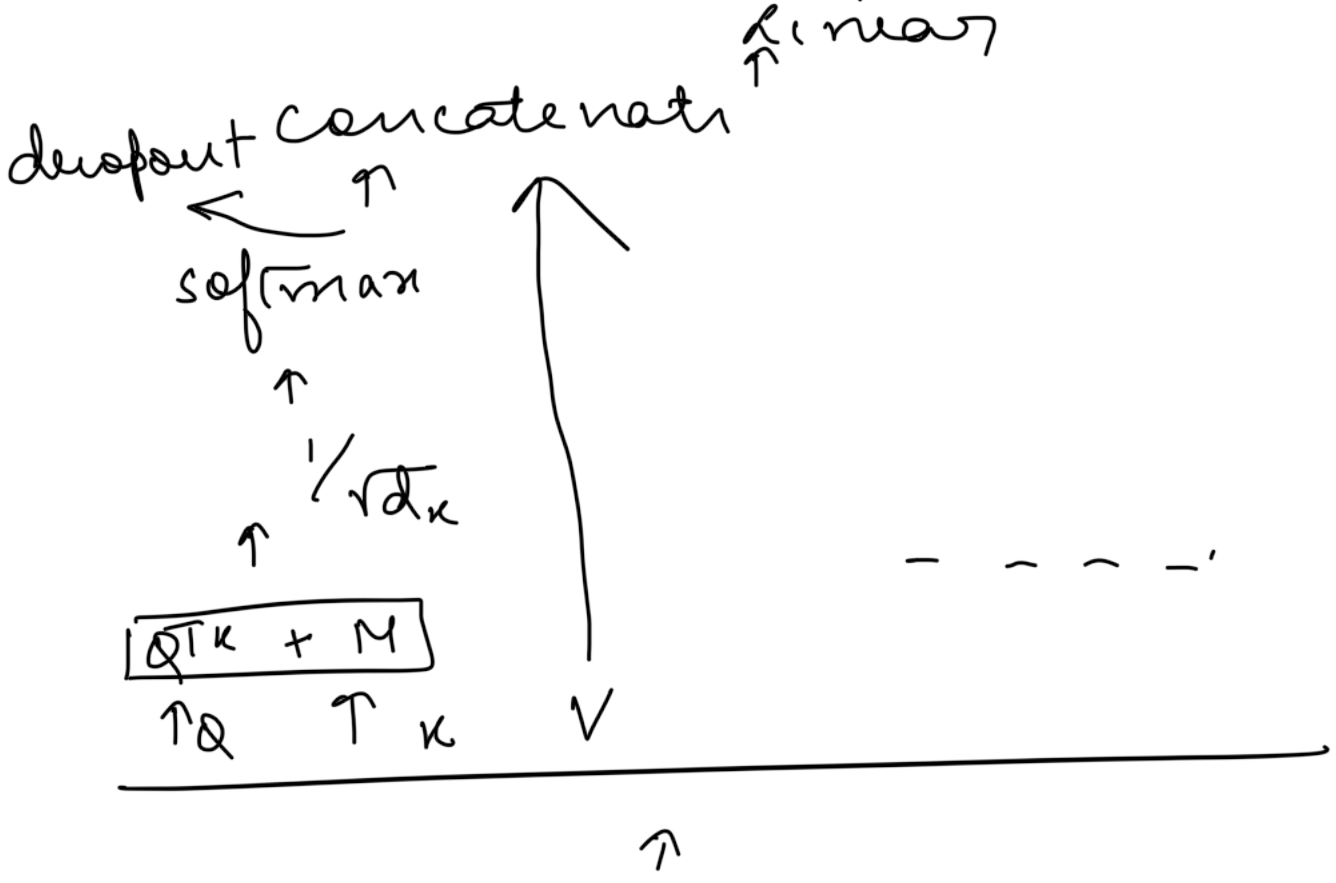
- input → book corpus, long-range contagious text (no shuffling of sentences)
- Byte-Pair encoding → tokenizer
- $V = 40478$ (vocab size)
- embedding dim $= 768$
- 12 decoder layers
- T = content size $= 512$
- atten^n heads $= 12$
- FFN layer size $= d_{model} \times 4$

$= 768 \times 4 = 3072$
(64×12)

- GeLu activa^n func^n → in FF N
- dropout, residual connec^{ns} were used to increase the performance to reach to the convergence.

Input size $\Rightarrow T \times d_{model}$

- masked multi-head attn^n



- after linear layer

Residual connec^{n} → layer Norm

↑

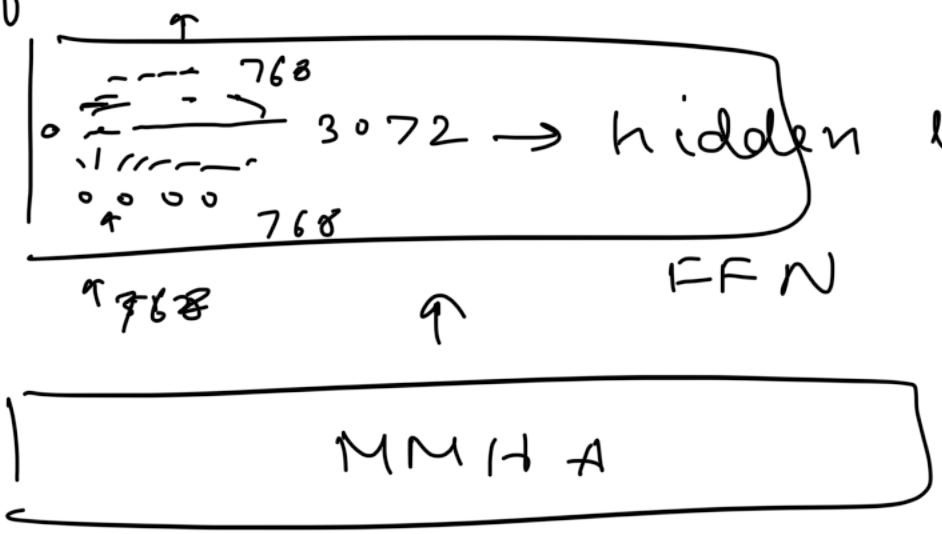
Linear

⟨⟨ allows the gradients to flow back.

$$H' = X + H$$

↳ skip / residual connec^{n}

- after MMHA,



- token emb. = V x emb-dim

= 40478 x 768 = 31 m.

- Posi^{n} emb. = context len x edemb.

= 512 x 768 = 0.3 m

31.3 m in input layer.

- the positional emb. are also learned, unlike original transformer which

uses fixed sinusoidal embeddings to encode the posi^{n}.

- Atten:^{n} params per block.

$W_Q = W_K = W_V = (768 \times 64)$

Per atten^{n} $= 3 \times (768 \times 64)$

12 heads $= 12 \times 3 \times 768 \times 64$

$= 1.7$ m

Linear $= 768 \times 768 = 0.6$ m

for all 12 blocks $= 12 \times 2.3 \simeq 27.6$ m

- FFN params per block.

$2 \times (768 \times 3072) + 3072 + 768$

$= 4.7$ m $\times 12$

$= 56.4$ m

Total = 117 m

L6 Pre-Training & Fine-Tuning

- Pre-Training

Batch size $= 64$

input $= (B, T, C) = (64, 512, 768)$

↳ emb. dim.

Min.

$$L = - \sum_{x \in V : X \in x_i = 1} \sum^{T} y_i (\log(\hat{y}_i))$$

Optimizer → adam with cosine learning rate.

Strategy → Teacher forcing for quicker & stable convergence

- Fine-tuning → adapt the model for various downstream tasks (with a minimal Δ in the architecture).
- each sample of labelled data, $x_1 \dots x_m$ → label y.
- initialize the params with the params learned by solving the pre-trained objective.
- at the output head, replace the pre-training LM head with a classifier head (a linear layer W_y)
- Obj. is to predict the label of input seq.

$$\hat{y} = P(y | x_1 \dots x_m) = \text{softmax}(W_y h_l^m)$$

Min.

$$L = -\sum_{(x,y)} \log(\hat{y}_i)$$

- Eg. Sentiment Analysis

  Text: ----

  sentiment : Positive $\hat{y} = \{+1, -1\}$

- Eg. Text Entailment / Contradiction

  Text: ---

  Hypothesis : ---

  Entailment : True / False / Not Related

we use a delimiter token <$> to differentiate two sentences.

- the network gets adapted as per the task.
- yes, it produces the same output seq. for a given prompt. => Deterministic output.
- but, we don't want same text, or irrelevant text. Should be creative. Also, accelerate the gen^ of tokens.