Moving Towards GPT

Generatively Pre-trained $\boxed{\text{Transformer.}}$

Self Attention?

what does self-attention do?

We already have a NN model which can predict words or tokens ahead.

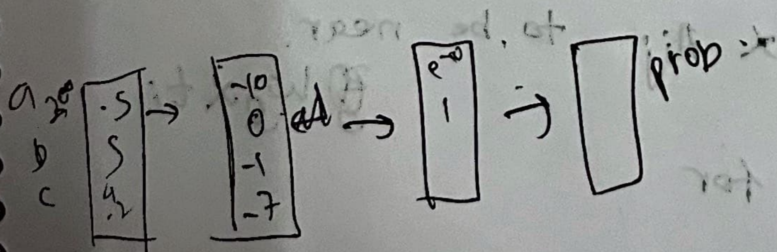
But the current embedding of words into the embedding space isn't satisfactory.

We want ~~to develop~~ the mapping from words to their embeddings to have richer context hidden in it.

So now, for eg. we want

$$E(\text{man}) - E(\text{woman}) = E(\text{nephew}) - E(\text{neice})$$

Basically adding a vector X on current words to their ~~feminine~~ female versions etc.



We want the model to understand the context

So we want the previous words to send information to the last current word in the sequence, and then the prediction will be based on the last in the sequence.

Eg: have adjectives adjust meanings of corresponding nouns.

initial embedding ⟶ converts words to a high dimensional space [here 12,000] and their position.

Now, first we need to identify which ~~words~~ preceding words (tokens) affect the current word embedding, for this we use a K, Q Key Query system.

W_K W_Q. ⟶ dim = smaller emb space × Og emb spa
128 × 12 000.

let A_i be the set of tokens preceding a token t_i that affect t_i more

∴ We want $W_K \times A_{ij}$ to be near $W_Q \times t_i$

in the smaller space for

Basically, for all words in the current context window:

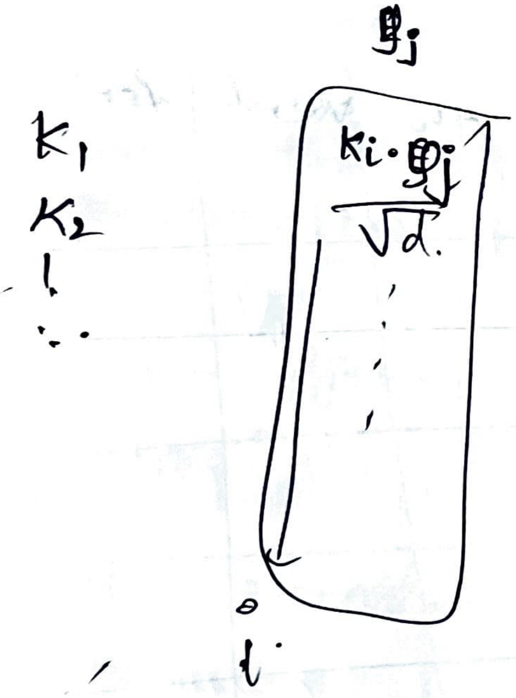
	fluffy	blue	crea	roamed	the	verd	for
	fluffy $\downarrow$ E_1 $W_Q \times E_1 \rightarrow \vec{Q_1}$	.	-	-	-		
fl $W_K E \rightarrow \vec{K_1}$	$\vec{K_1} \cdot \vec{Q_1}$	$\vec{K_1} \cdot \vec{Q_2}$					
bl :							
crea :							
roa							
the							
ve							
fore							

$\therefore \vec{K_i} \cdot \vec{Q_j} \Longrightarrow$ How much should word/token i affect word/token j.

also we divide $\dfrac{\vec{K_i} \cdot \vec{Q_1}}{\sqrt{d}}$

$\sqrt{d} \longrightarrow$ dim of smaller emb space

Then we apply softmax on each col



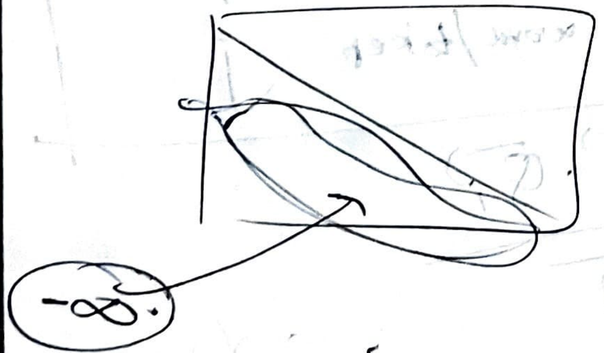
Softmax to get prob of how much that word (i) affects word. j

---

At prediction time our model only has access to words prior to the one being predicted so we can only want words before current one to affect them :)

$\therefore$ we set $K_i \cdot q_j = -\infty \;\; \forall \; i > j$

So their prob after softmax becomes 0.



called MASKING.

in bidirectional models, where the goal is to understand the meaning of a sequence rather than predict. then we can use that info too.

Seeing this square matrix we can see how context size is a huge bottleneck for models.

Now that we know how much a word affects another, how to update that value.

Value vector that lives in the emb space

$$\therefore \quad \vec{E_j} = \underset{\text{og}}{\vec{E_j}} + \Delta \vec{E_j}$$

$$= \vec{E_j} + \sum \left[\frac{\vec{k_i} \cdot \vec{q_j}}{\sqrt{d}} \right]_{\text{softmax}} \cdot \vec{W_V} \times \vec{E_i}$$

$$= \boxed{\vec{E_j} + \sum \text{softmax}\left(\frac{(W_K \times E_i) \cdot (W_Q \times E_j)}{\sqrt{d}} \right) \cdot \vec{W_V} \times \vec{E_i}}$$

No. of paramet

$Q \rightarrow 12 \times 10^3 \times 120 \sim 1.5 \times 10^6$.

$K \rightarrow 1.5 \times 10^6$

$V \rightarrow (1.2 \times 10^4)^2 \rightarrow 150 \times 10^6$.

Value → too large

∴ try to keep |Value| ~ |Query| order.

∴ we try to break down Value

$V_\downarrow$ and $V_\uparrow$.

eg

large emb spa $\xrightarrow{V_\downarrow}$ small space $\xrightarrow{V_\uparrow}$ large emb space

$\therefore V_\downarrow \vec{E_i}$ $V_\uparrow \times (V_\downarrow \times (\vec{E_i}))$

$\vec{V}W_x V$

$1.2 \times 10^4 \times 1.2 \times 10^2 \sim 1.5 \times 10^6$

$\therefore W_{V_\downarrow} + W_{V_\uparrow} + W_k + W_q = 6 \times 10^6$ params.

Basically.

$V_\uparrow \left(\sum (\text{prob}) \cdot (V_\downarrow \times E_i) \right)$

↑

calc in smaller space.

Cross attention: K and Q act on 2 diff datasets eg: translating 1 lang to another.
here we again Do not mask as we are trying to understand the rel^n b/w 2 languages.

here $K_j Q_i$ would tell prolly which word "i" in lang 1 corres to word "j" in lang 2.

---

* Multiple Attention heads:

We run this calculation to find ΔE_i ~~for~~ MANY times independently n in parallel.

$$\Delta E_i = \Delta E_{i1} + \Delta E_{i2} + \dots$$

each head.

Diff heads can specialise on diff aspects of self attention:

eg 1 → Syntatic roles (noun-verb-align)
2 → local word collaboration ("New York")
3 → - - -

How to encourage specialisation in diff heads)

① Random initian.

② loss func is comm so if 2/more heads do the same thing ⟹ kind redundant ⟹ loss f^n pushes them to diff things.

(from all heads)
in papers/stuff, the $V_{\uparrow}$ matrices are often
all concatenated into one matrix

$$\underbrace{12000}_{} \overbrace{\left[\quad\right]}^{128 \times \text{no. of heads}} \; V_{\uparrow}$$

GPT-3

Weights devoted to Attention !!

$$\text{n\_layers} \times \text{n\_heads} \times \begin{bmatrix} d\_emb * d\_query & \leftarrow K \\ + \; demb * d\_query & \leftarrow Q \\ + \; d\_value * d\_emb & \leftarrow V_{\uparrow} \\ + \; demb * d\_value & \leftarrow V_{\downarrow} \end{bmatrix}$$

10^2 $\qquad$ 10^2 $\qquad$ $4 \times 1.5 \times 10^{6}$

$\approx 6 \times 10^{10}$

≈ 60 billion

$\sim \frac{1}{3} \times \underbrace{180 \text{ billion}}_{\text{total params}}$

How do ~~m~~ LLM's store facts?)

Each MLP layer adds information to the words.

Johnson - Lindenstrauss lemma:

No. of vectors that you can cram into an n-dim
nearly perp.
space incr. exponentially with 'n'.

not intuitive but it allows LLMs to have independent ideas - ALMOST ⊥ to each other and have exponential increase.
thus allowing so many more ide

∴ models scale so well with size
this is one reason

∴ diff ~~vecto~~ rows n columns in MLP weight matrices correspond to ideas.