StatsQuest - Transformer NNs

transformer is a type of nn

nns need numbers as inputs so start w/ word embedding
= number for word

embed networks are the same for each word

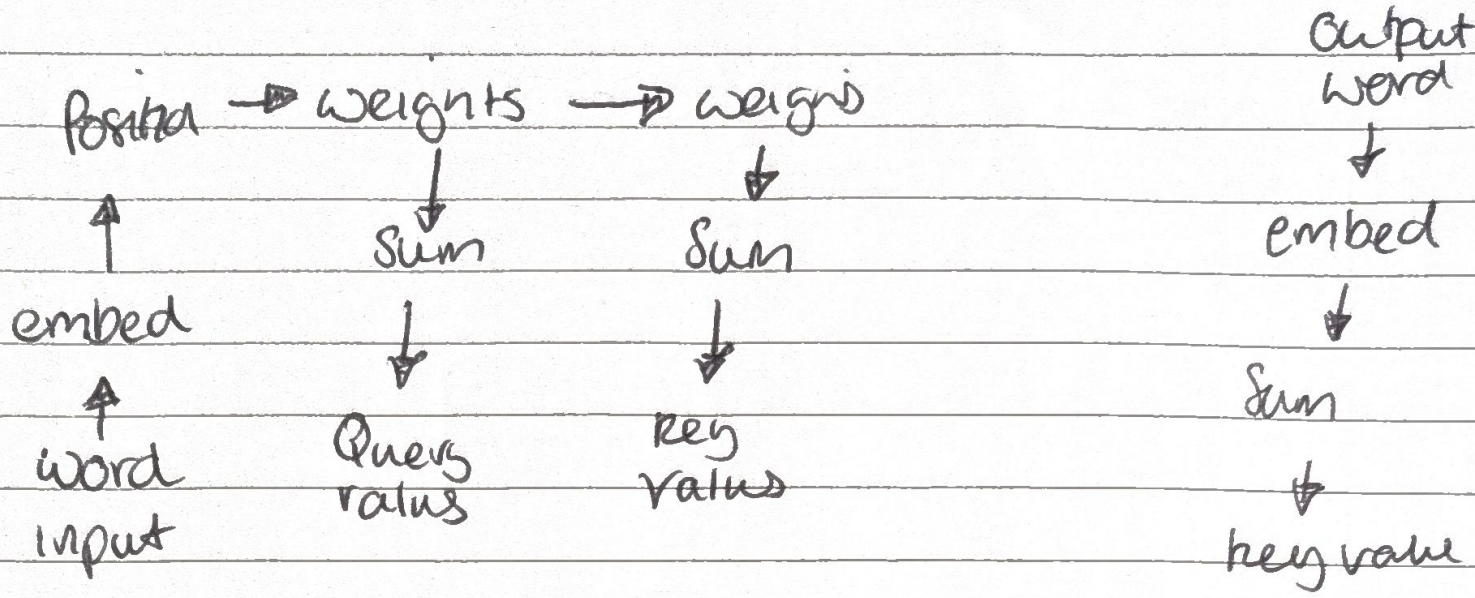
Positional encoding → retain order context

- take word embedding and apply tranformatio to get positional encode
  - ↳ usually cosine etc

this is how A transformer keeps track of word positions

How do transformers keep track of the relationships between words?

Self-Attention
- → keeps track of a words similarity to all other words in a sentance. including itself



use Key values to calc similarity w/ Query

(have use dot prod for sim)

① Query x input key = Sim

② Query x ouput key = Sim

large = more similar

→ has more influence on encoding using softmax (0, 1) to harvest word value

Decide the words similar to a word to influence its encoding

After a word in sentance has been compared to all other words

again (Pos vals x weights) x sum = value

value * softmax

do for input side & outside & scale

Result = Self-Attention values for words! from Sentance

Repeat Self-Attention Scores for each word

can re-use Query & value weights

can calc all words @ once, does need to be iterative

word → embed → Position → Self-Attion

↳ these are the progression of values that represent a word

Self-Attention values improve as they contain input from all other words
↳ this enables words to have context

this helps establish how each word in the input is related to the others

→ Self Attention Cell:
- think of are w! weights calcing the Queries, keys & values as cell
- layer → Stack cells to capture more relationships in complicated inputs
- cells start w/ positional encoded values

in the first transformer paper they stacked 8 self-Attention cells

↳ Multi-head attention

in the word value flow take the:
- Pos encoded values
- self-Attention values

& sum together

these are called residual connections

Allows self-Attention to focus on relationships

→ Position encodes are added back the residual values are the encoded inputs for the transformer

4 stages - embed, Position, Self-Att, residual

- encode into nums
- encode positions
- encode relationships
- easily & Quickly train in Parrell

→ there are lots more complex things that can be added to a transf

transformers are an encoder-decoder architechure

Previous has all ~~EA~~ been encoders into the resid connections

Decoder

Starts w/ input embeds - start w/ EOS

↓

Posit encodes

↓

Self-Attention (Diff weights to encode SA)

} all steps and words done @ same time

Previous looked @ Self-Att relations between Sentance words

But also need to track between input & output (Attention)

Decoder needs to keep track of signif words in the input (not loose any context)

"encoder-Decoder Attention"

- take word from Decode & take Query vale
- Calc key values from all words in encoder
- Calc all similarities (dot prod)
- Softmax to decide weight influence

Output = encoder-Decoder Attention value for A output word

E-D Attention has DIFF weights to Self Att

then calc a resid value from the Self-Attention + E-D Attention

Resid)
- E-D Attention - input/ouput relation
- Self-Att - intra sentance relation
- Positional encode
- word embed

finally resids go into a fully connected layer to predict the Decoded word from vocab

repeat until Decoder outputs EOS

TRANSFORMERS

1. embed
2. Position
3. Self-Attention
4. Encode-Decode Attention
5. Residual to chain together