raper: A...

- before appearence of transformers, Rnn, Lstm, gru were used a lot.

1

◆let dive into Rnn

- what is Rnn: Rnn is a kind of neural network which is useful for dealing with sequence data.
  - sequecne data: they all have order
    - like text data → you are great boy
    - if we destroy order between them, they dont make sense

★) Rnns work with text data most of time
- some application of Rnn =>
  1) machine translation
  2) sentiment analysis
  3) language model
  4) text topic

★) all of tasks above work with text data which have order
- ★) the movie was ~~was~~ so funny → positive
- ★ Hello, are you okey? → translation.
- ★) language model → predicting next word ★) the most important task in nlp
- ★) language model is basic, simple like image classification in cnn

- let dive into know more about language model.
  - predicting next word in sentence
  - backbone of nlp tasks
  - gives a probability of next word based on previous word

- example:

you are great → boy
→ girl
→ desk
→ actor
→ Football Player

- Probability of all words in vocabulary. based on previous

- So all of them have to keep somewhere to know about next word

$$P(X^{t+1} \mid \underbrace{X^{t}, X^{t-1}, X^{t-2}, \dots, X^{0}}_{\text{Previous word}})$$

X^{t+1}: next word or Real word

◆ big difference between Rnn with other kinds of neural network:

- neural networks like cnn, Ann are not able to consider time
- there is no difference between input data
- they are not able to understand order between data

you are good boy

- their input must be Fixed → what does it mean?

cnn → 48×48×1 } → input data must be same size.
ann → (24,)

- using Padding for text data? → sequences are not same size

increasing number of input data → Parameters ★

For cnn, and, it does not matter which one comes first

you great are boy ≈ dont make sense
are boy you great

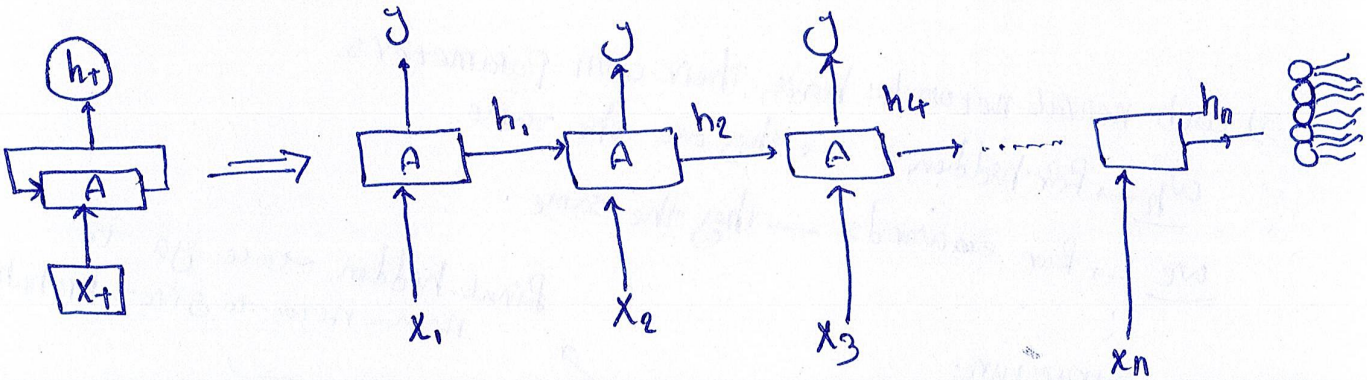
→ because they have order → Rnn comes to tackle problem.

- they can process words with order through time → in order

First: you goes to network
second: are goes to network
third: great goes ↑ ...

- For processing each word, we need previous to make sens
  For next word prediction

- lets see structure of Rnn:



more general:

*) all of units have two input, however first one

**) we insert a random vector to first one → to make them similar

***) some tips:

1) words are not able to enter neural network directly:

↳ they go to somewhere called embedding factory

to make a array of them.

you → embedding → [vector]

are → ~ → ~

great → ~ → ~

2) what is hidden_state: it is a context vector which stores information from each timestep to predict next word

- combined with word at time t, h_{t-1} → to predict X_{t+1}

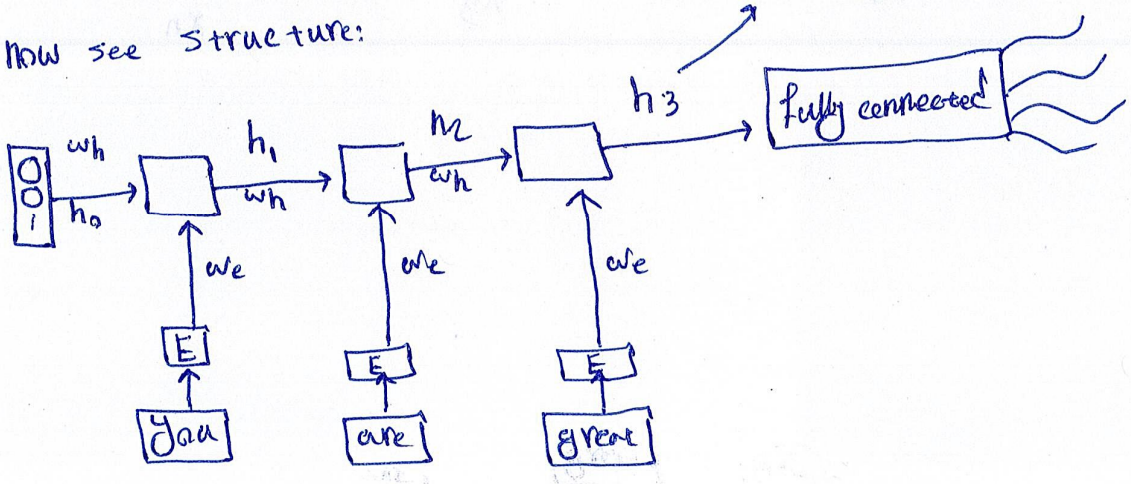
h_t → carries information from previous part

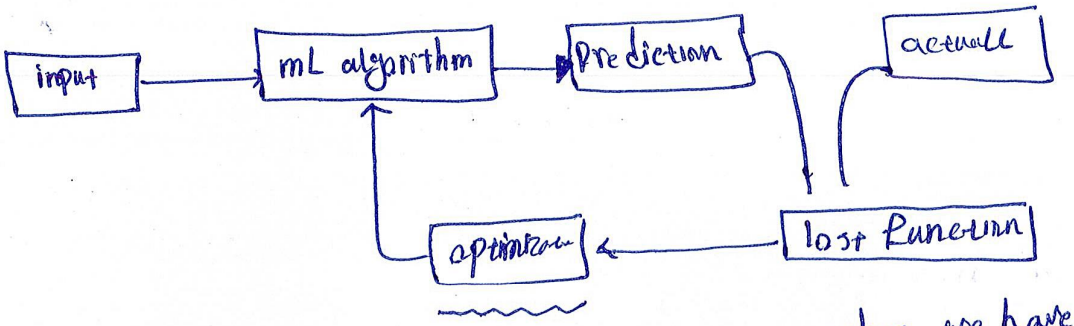
3) each neural networks have their own parameters.

w_h → for hiddens → they are the same.

w_e → for words → they the same.

- now see structure:





*) there is big problem in optimization ⟹ when we have long input word

For the first time all weights are random. worse

- we must use backpropagation to update our weights. → w_h

- after some step → gradient will disappear → we can't update our weights → train will stop → forget data which comes first

*) Vanishing gradient decent

*) exploding gradient decent

LSTM comes to address problem by ... gates.

* LSTM is special version of Rnn → solves problem through concept of gates

* language model → predicting next word is so challenging when rnn comes to handle propagate → vanishing gradient

A) Lstm has 3 kinds of gate
- forget gate
- input gate
- output gate.

- Lstm's are capable of learning long-term dependencies.

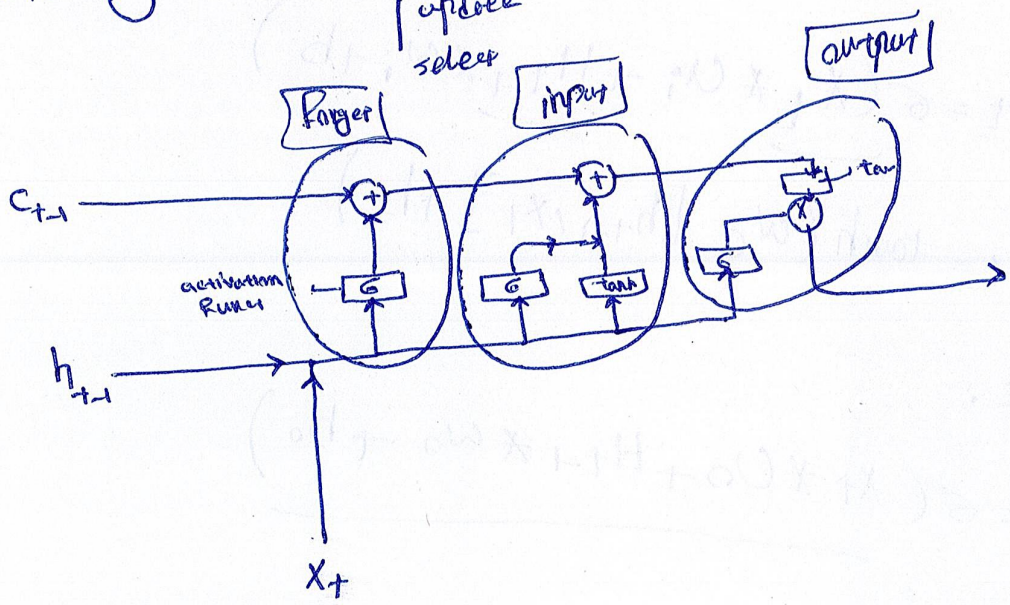
- Lstm introduced 1997

- they work well on wide variety of sequential problems.

* remembering the context of the sentence for long priod of time is their default behaviour. → they have memory

* memory is used to
- delete
- insert
- update
- select

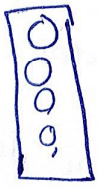
data

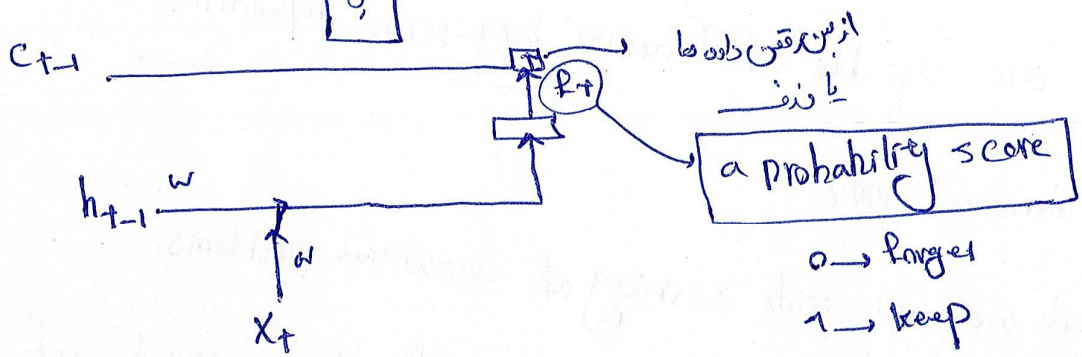


Forget gate

example of text prediction:

- Prince was feeling uncomfortable in the party while chiray enjoyed the party.
- there some important word's here → subject, feeling / adjective...
- in forget gate all ~~relevant~~ irrelevant information is removed via filter
- filter → f_t → it is created by multiplying x_t and h_t → Passing sigmoid

function →  → data is between 0 — 1



forget gate $\Rightarrow f_t = \sigma(X_t * U_f + h_{t-1} * W_f + b_f)$

وزن وزن

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

input gate: $i_t = \sigma(X_t * U_i + H_{t-1} * W_i + b)$

$$C_t = \tanh(W_C [h_{t-1}, X_t] + b)$$

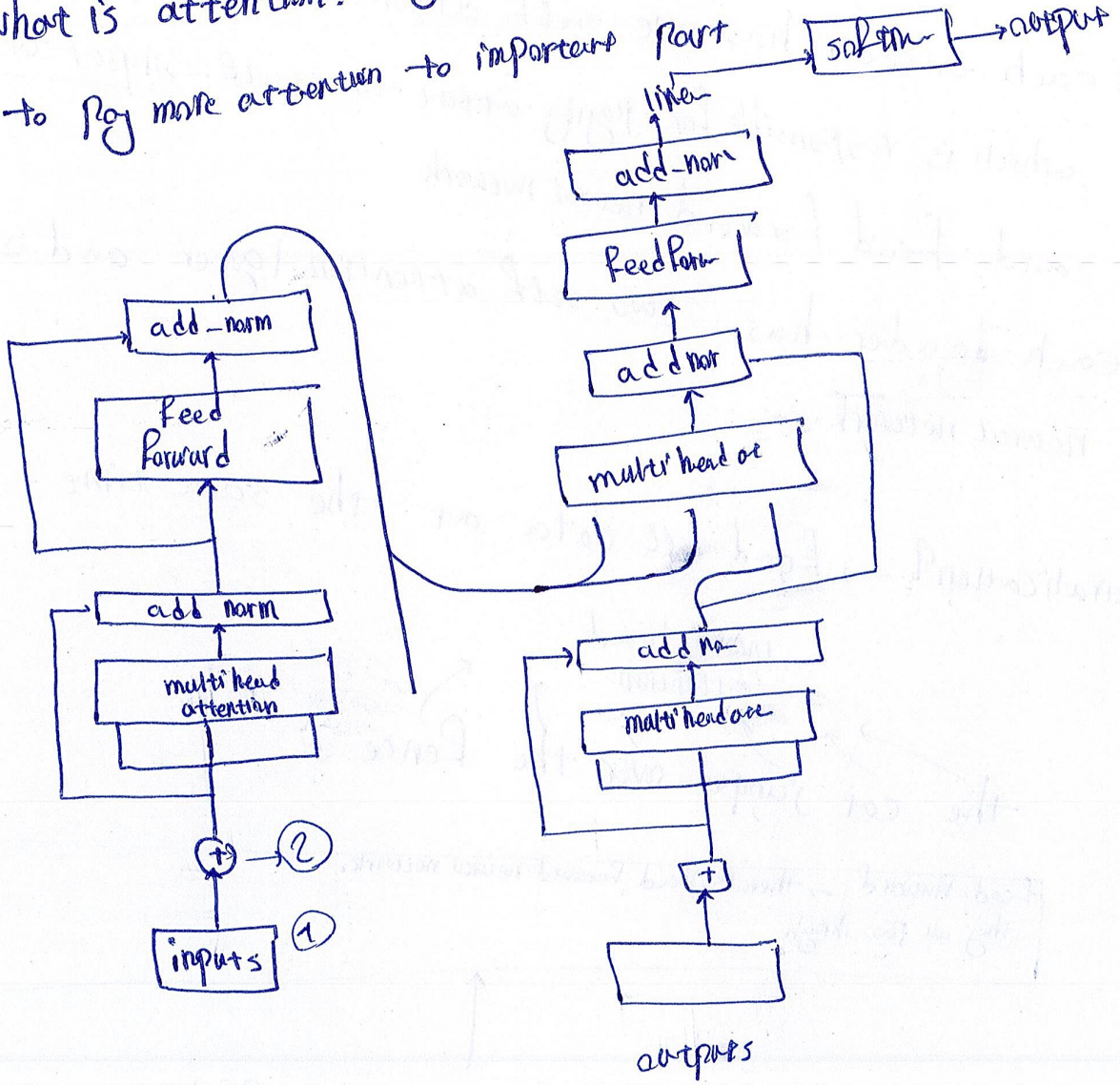
output-gate:

$$o_t = \sigma(X_t * U_o + H_{t-1} * W_o + b_o)$$

Transformers:

- Transformers structure only rely on attention mechanism.
- they do all work Parralel.
- they don't use any Rnn, Lstm, ---- → their problem.
- they more faster, smarter and quicker than → Parallele way.

*) what is attention? in general, attention is the ability of a model to pay more attention to important part



Transformers

- encoder
- Decoder

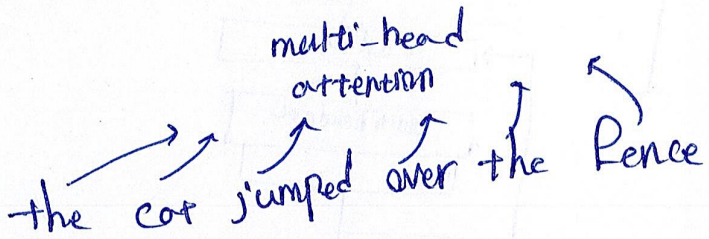
→ number of both are 6

- right hand side → Decoder
- left — — → encoder

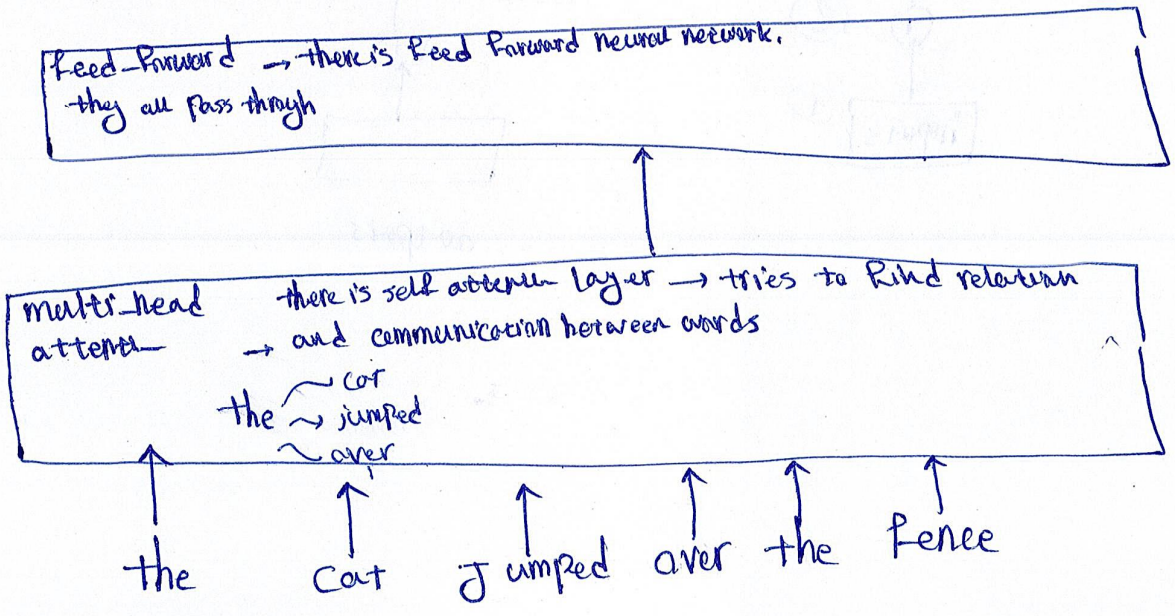
☆) each encoder has one self attention layer which is responsible for paying attention to itself → input text and Feed Forward neural network

☆) each decoder has two self attention layer and one neural network.

☆) Paralization? → Feed all data at the same time



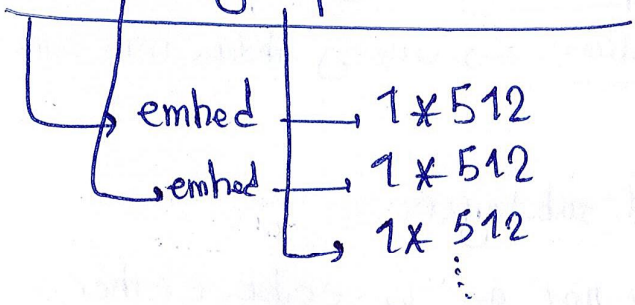
☆)



- we have six enceders. → each of them, they are diffe

*) all input either enceder or decoder are embedded

* the cat jumped over fence



what is embedding → this is a way for presenting words

* each word represents by 512 numbers.
↳ this is hyperparameter

2) Positional encoding:

- since the model do not any Rnn_lstm, how does it able to understand order of the data?
- by adding Positional encoding → let it know → where words come

*) output and linear →

output of decoder can be transformed into someth to be understand.

- Normalization layers:

  * they are between Feed Forward layer and self atten layer

  * what do they do?

  * Responsible For → normalize output of sub layer

  * an improvment over batch normalizon → collecoy data into same region

- skip Connection:

  - there are some arrows around sub_layers.

  - they carry inforna dosnt not go to either

    self or Feed Forward.

  - It helps model not forget things.

    helps data go thrugh model.

* **novel idea**

  - multi-head attention

  - Positional encoding.

Lets Dive into them

there only 2 kinds of multi-head attention

1) multi-head attention.

2) masked multi-head atten.

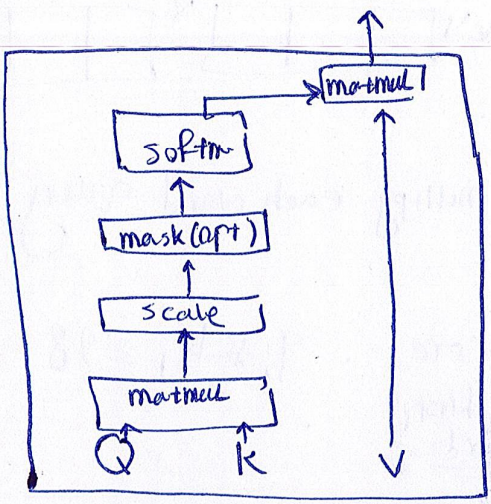
Difference:

- in multi head attention all the words compared with all the other words
- in masked multi only the words that are coming before a word compared to that word.

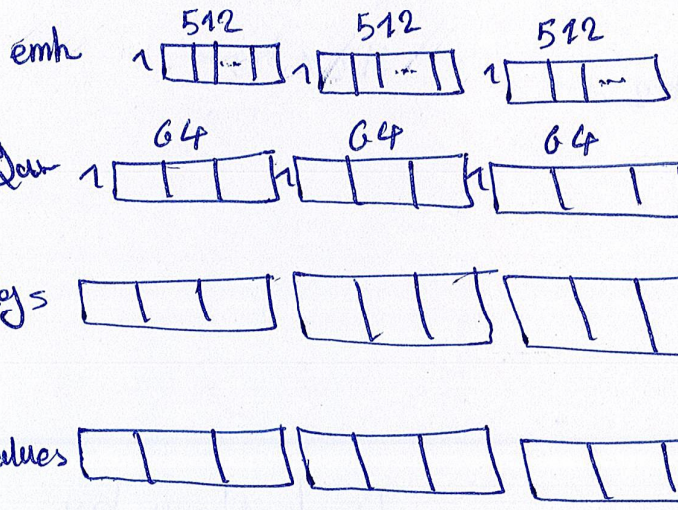
A) lets go to attention-layer

A) there is scaled dot product

★ How does attention calculate?



input that was Ran



Query

1) multiply this embeddy vector to these matrix

2) Values of mat are random, during train they will train to be learn

key

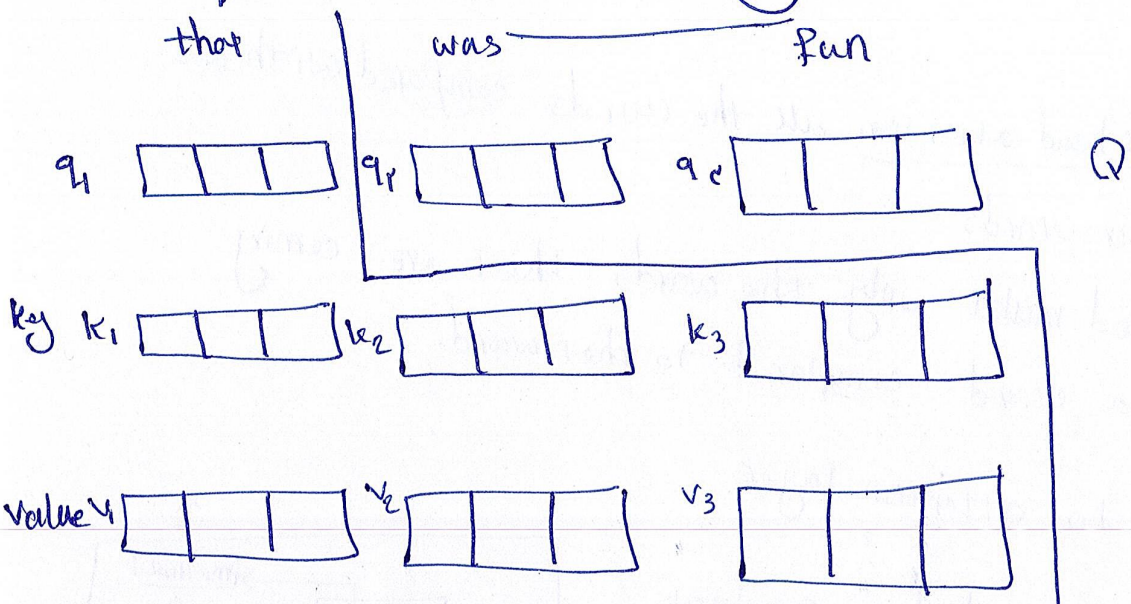
value

let go to calcu:

attention will give a score → it means [illegible]

between each words.

- Dot product ⟹ show similarity → [illegible]



multipy each word query with others word keys

score for first word: $q_1 * k_1 = 98$ $q_1 * k_2 = 46$ $q_1 * k_3 = 76$

second: $q_2 * k_1 = ?$ $q_2 * k_2 =$ $q_2 * k_3 =$

third

→ all of them are done together.

△ when we have all the score → we divide them by 8 → why 8? roots of 64 → is lenght of key, query, value length

Score $q_1 * k_1 = 98$ $q_1 * k_2 = 46$ $q_1 * k_3 = 10$

Divide by 8 12,25 5,25 9,5

because it is root of keys values ; Query dimen

softmax 0,93 0,07 0,06

weighted wv1 wv2 wv3

sum z_1

→ تنها بردار کلمه دوم است

- باقی کلمات هم به همین شکل

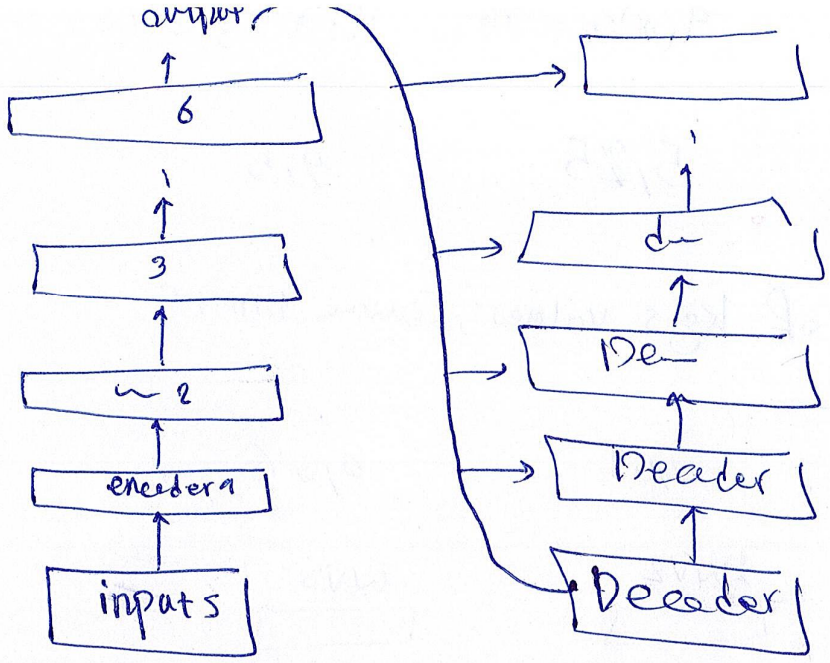
→ z_1 z_2 z_3 → output of attention layer

*) multi_head atten does 8 times

Δ) output of each multi_head attention

L_3 64

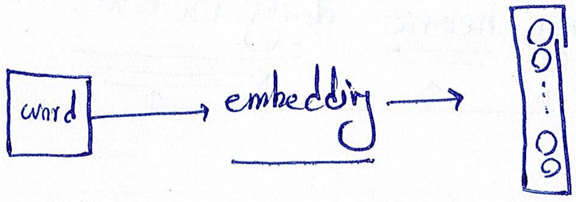
Δ) we have eight output → concat them to make orginal Dimention.



output goes to multi-head attention of decoder

another views:

- Embedding layer takes a word, gives a vector of number
  - they are const



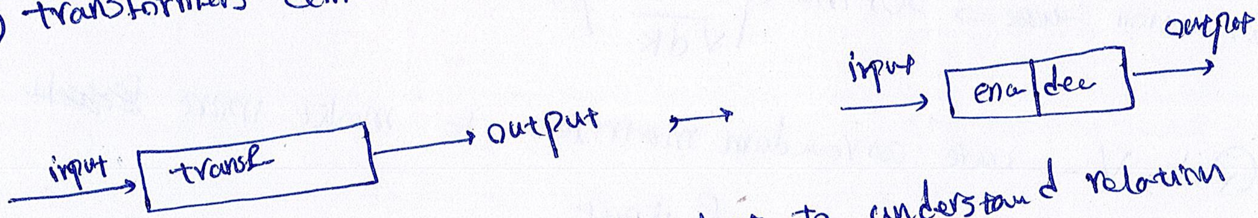
★ مثلا اگر شما بخش کلمه شیر را ببینی،
در هر جمله کلمه شیر را دقیقا باید
embed
نمایش می دهد.

- من شیر خوردم
- شیر حیوان خطرناکی است.
- شیر فلکه اصلی را ببند.
- تورو شیر کردند
:

برای تمامی این ها embedding یک vector ثابت می دهد

- ولی قبول کنین اینا باهم کاملا متفاوت هستن؟
- شیر در هر جمله کاملا متفاوت است.
- هیچ نگاهی به فضا، مفهوم و بار جمله ندارد.
- اینها بود که کار خراب شد ← یعنی ماشین درک مناسب نداشت.

★) transformers come to address.



in encoder → we use self attention mechanism to understand relations between data/text → this because word like book has different meaning in each sentence.

- by using/calculating attention score.

You are great boy ⟶ ت تو پسر خیلی خوبی هستی.

- transformers apply to machine translation task.

*) I am really hungry, because of my hectic daily routine.

- all of them go through embedding layer

then by using self attention ⟶ (explained) ⟶ calculates score

by dot product

⟶ scaled Dot product ⟶ find relationship and build up

a embedding layer.

- so, enceder builds up powerful embedding vector

including context.

- attention score $\Rightarrow$ $\text{softmax}\left(\frac{QK^T}{\sqrt{dk}}\right)V$

- Q, K, V ⟶ are random matrixes to make more flexible and reduce size of input

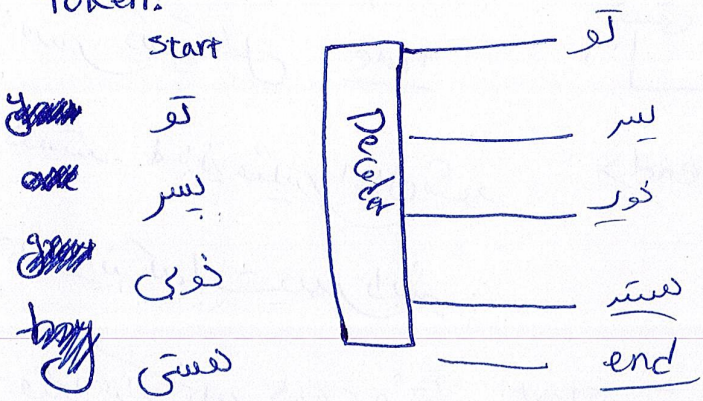
- $\sqrt{dk}$ ⟶ is root of matrixes dimen ⟶ 8

- transformers apply on machine translation

- it has difference between traing/testing process

- training part:

input of decoder => a shifted output, adding start token.



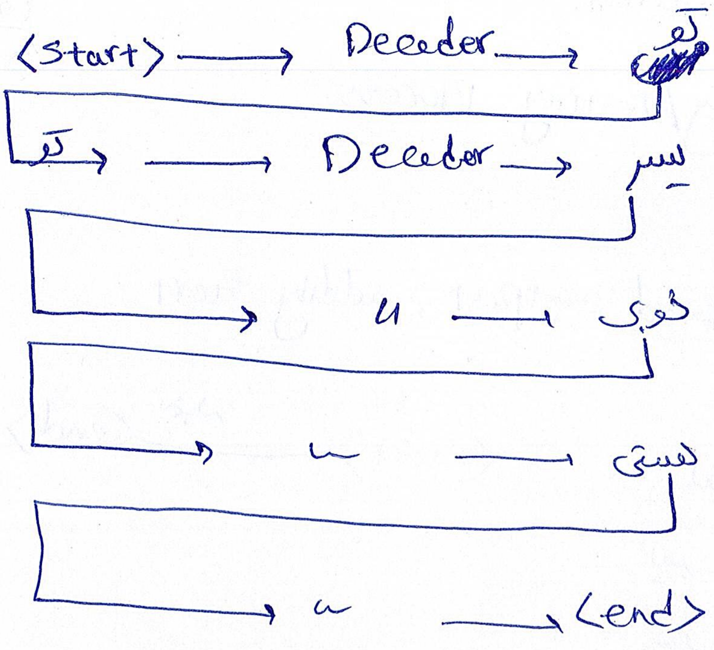
- in decoder, we have generating data

- decoder has two input because of two attention mechanism
input dec

1) is responsible for Lindy relation between (output)
2) encoder output with decoder input.

a) mask attention stop cheating structure

L, we prepared all information to produce next word. well done



۱۵ در مرحله test
تنهایی که داده ورودی بیانی کنند
به دنبال تولید داده است. آن داده
به عنوان ورودی بعدی مطرح است.

- در فرآیند train که آماده هستند
ورودی دیکودر کامل true
هستند. اما باید فیلتری اعمال شود
تا نتواند کلمات بعدی را ببیند.

- مثلا برای کلمه خوبی فقط start و تو را ببیند.

$$\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{فیلتری}\right) V$$

ماتریسی که عدد یا score کلمات بعدی را صفر می‌کند.

- multi_head attention → all embedy are availa
masked → some of them are are avaible