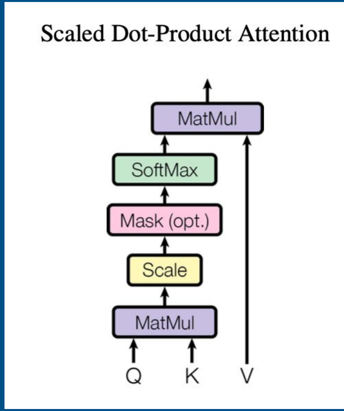
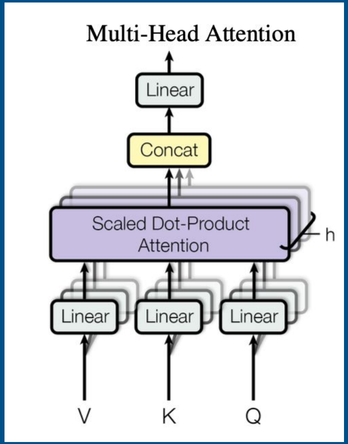
Attention as Context Summary

is all you need





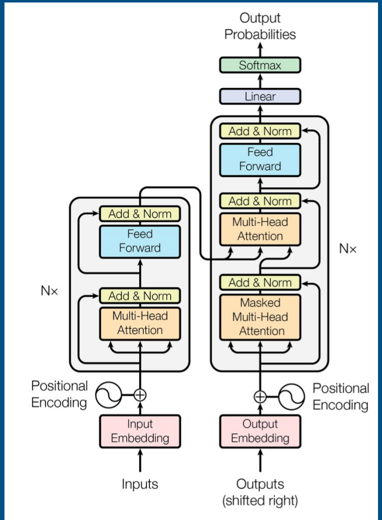


Figure 1: The Transformer - model architecture.

1. Motivation
2. Attention as context summary
3. How to calculate Attention?
4. Further map
5. Conclusion

1. **Motivation**
2. Attention as context summary
3. How to calculate Attention?
4. Further map
5. Conclusion

Motivation: review Markov property

📝Markov Property: given the present, the future does not depend on the past.

❓Is it sufficient in language modeling:

example:

Easy peasy lemon squeezy

If use markov model it would be:

Easy peasy lemon ❓

Bigram

Motivation

📝Markov:

Easy peasy lemon ❓

Bigram

🤔But what we want:

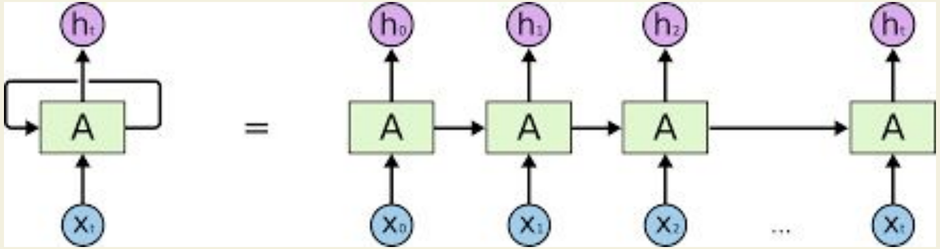
Easy peasy lemon ❓

context summary

Motivation: training

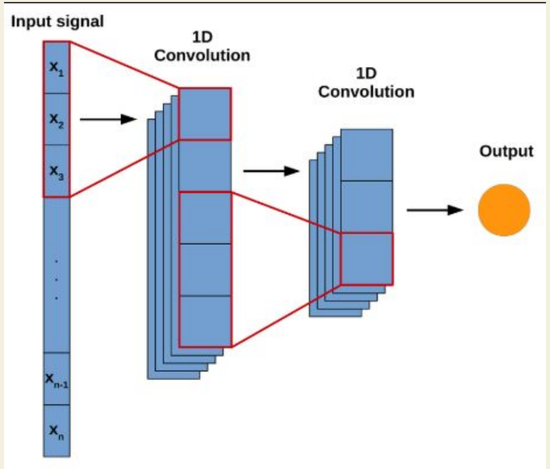
We have tried different methods to achieve it:

RNN
(Recurrent NN)



✅Long Dependency
❌not-Parallelizable at all

CNN
(Convolutional NN)

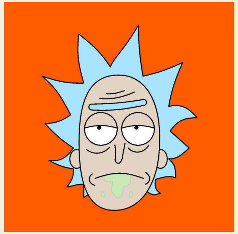


✅Parallelizable
❌weak Long Dependency unless many layers

Motivation: training

So here is the question:

: **Long Dependency(RNN)** or **Parallelizable(CNN)**?
That is the question. Whether 'tis nobler in the mind to suffer……



: Stop, enough. Here is the **Transformer**.
You have both now.

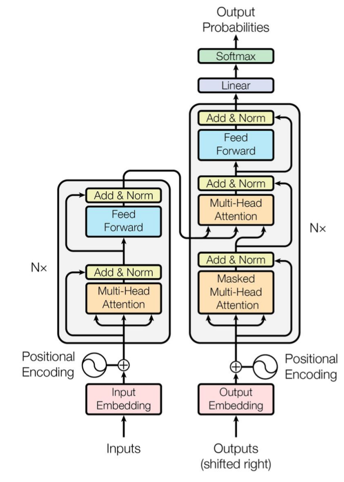
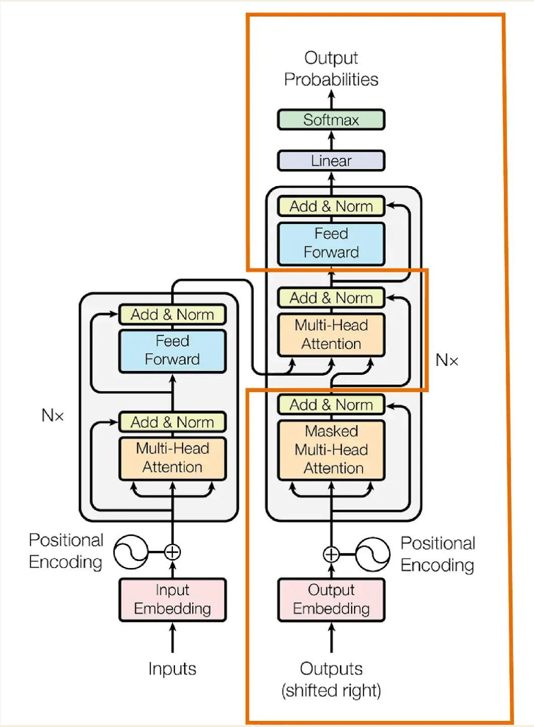
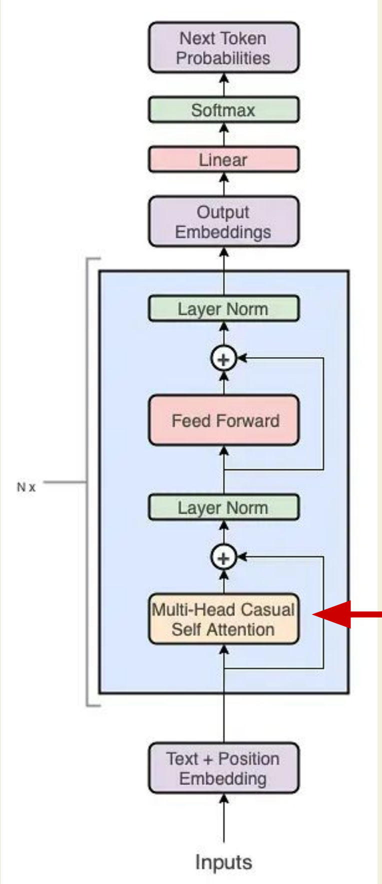


Figure 1: The Transformer - model architecture.



Encoder(left) + Decoder(right)

abstract the right part



Decoder Only(e.g. GPT-like)

We only focus on **Attention** today.

Motivation

📝Markov:

Easy peasy lemon ❓

Bigram

But what we want:

Easy peasy lemon ❓

context summary

RNN 🙁
CNN ☹️
Transformer(Attention) ✅

1. Motivation
2. **Attention as context summary**
3. How to calculate Attention?
4. Further map
5. Conclusion

Attention as context summary

Use ChatGPT to generate the example, it should looks like:

Easy

🧠 **Step 1: Understand what happened under the hood — *in Inference***

- **Auto-regression**: generate one token at a time.
- **Next token prediction**: predict the next token based on all tokens before it.

example: 1️⃣: [Easy] ----------predict----------> peasy

2️⃣: [Easy peasy] --------------------> lemon

3️⃣: [Easy peasy lemon] -----------> squeezy

🎯Final output : [Easy peasy lemon squeezy]

Attention as context summary

🧠 **Step 1: Understand what happened under the hood — *in Inference***

example: 1️⃣ [Easy] ----------predict----------> peasy

2️⃣ [Easy peasy] --------------------> lemon

3️⃣ [Easy peasy lemon] -----------> squeezy

🎯Final output : [Easy peasy lemon squeezy]

🤔 **Step 2: So how should we achieve it? — *in training***

We train it to mimic that behavior:

- Train it to **predict the next token**, but only allow it to **look back** — we call it "Causal" or "Masked"
- Calculate loss, back propagation, etc…… — we are all familiar with this, ignore

Attention as context summary

🧠 **Step 1: Understand what happened under the hood — *in Inference***

example: 1️⃣ [Easy] -----------------------------> peasy

2️⃣ [Easy peasy] --------------------> lemon

3️⃣ [Easy peasy lemon] -----------> squeezy

🎯Final output : [Easy peasy lemon squeezy]

🤔 **Step 2: So how should we achieve it? — *in Training***

We train it to mimic that behavior:

- Train it to **predict the next token**, but only allow it to **look back** we call it "Causal" or "Masked"

Use matrix to visualize it: "Easy peasy lemon squeezy"---training data, what we've already knew

"Easy" [Easy, ❓ , ❌ , ❌],-----------see: Easy------------------------>peasy

"peasy" [Easy, peasy, ❓ , ❌],----------see: Easy, peasy---------------->lemon

"lemon" [Easy, peasy, lemon, ❓],-----------see: Easy, peasy, lemon----->squeezy

✅ Easy peasy lemon ❓

Attention as context summary

🧠 **Step 1: Understand what happened under the hood — *in Inference***

🤔 **Step 2: So how should we achieve it? — *in Training***

"Easy" [Easy, ❓ , ❌ , ❌],-----------see: Easy------------------------->peasy

"peasy" [Easy, peasy, ❓ , ❌],----------see: Easy, peasy--------------->lemon

"lemon" [Easy, peasy, lemon, ❓],-----------see: Easy, peasy, lemon----->squeezy

✅ Easy peasy lemon ❓

🤔**Step 3: Task is how to better summarize history context?— *in Training***

: All tokens are equal, but…but some tokens are **more equal** than others.
Yes, I'm saying **Attention** mechanism.---------like weighted sum

Easy peasy lemon ❓

context summary

Attention as context summary

🧠 **Step 1: Understand what happened under the hood — *in Inference***

1️⃣ : [Easy] ----------------------------> peasy

2️⃣ : [Easy peasy] --------------------> lemon

3️⃣ : [Easy peasy lemon] -----------> squeezy

🎯 Final output : [Easy peasy lemon squeezy]

🤔 **Step 2: So how should we achieve it? — *in training***

"Easy" [Easy, ❓ , ❌ , ❌],-----------see: Easy------------------------>peasy

"peasy" [Easy, peasy, ❓ , ❌],----------see: Easy, peasy---------------->lemon

"lemon" [Easy, peasy, lemon, ❓],-----------see: Easy, peasy, lemon----->squeezy

✅ Easy peasy lemon ❓

🤔**Step 3: Task is how to better summarize history context?— *in training***

: **Attention** mechanism.---------like weighted sum

✅ Easy peasy lemon ❓

context summary

🔁 **Summary:**

Inference defines the goal → Training mimics the goal → **Attention** as a context summary shows the way!!

1. Motivation
2. Attention as context summary
3. **How to calculate Attention?**
4. Further map
5. Conclusion

How to calculate Attention?

$$\text{Attention}(Q,K,V) = \boxed{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})}V$$

weight

Every token has its own **q, k, v**:

🙋‍♀️q: query-------> “What am I looking for?”

🪪k: key----------->”A brief intro about me”

📝v: value-------->”The real info of me”

q*k: “How much does your info match my need? ”

softmax(q_0*k_0, q_1*k_1, …, q_n*k_n**)**: attention weight

softmax(...)*v: context summary

Q, K, V are matrix of q, k, v, for calculation efficiency. And divided by sqrt(d_k) is for scaling, ensuring training stable.

	Easy	peasy	lemon	squeezy
q:	q_0	q_1	q_2	q_3
k:	k_0	k_1	k_2	k_3
v:	v_0	v_1	v_2	v_3

How to calculate Attention?

$$\text{Attention}(Q,K,V) = \boxed{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})}V$$

weight

Every token has its own **q, k, v**:

🙋‍♀️q: query-------> “What am I looking for?”

🪪k: key----------->”A brief intro about me”

📝v: value-------->”The real info of me”

q*k: “How much does your info match my need? ”

softmax(q_0*k_0, q_1*k_1, …, q_n*k_n**)**: attention weight

softmax(...)*v: context summary

	Easy	peasy	lemon	squeezy
q:	q_0	q_1	q_2	q_3
k:	k_0	k_1	k_2	k_3
v:	v_0	v_1	v_2	v_3

1️⃣**q*k**:

q \ k	**Easy(k_0)**	**peasy(k_1)**	**lemon(k_2)**	**squeezy(k_3)**
Easy(q_0)	q_0*k_0	❌	❌	❌
peasy(q_1)	q_1*k_0	q_1*k_1	❌	❌
lemon(q_2)	q_2*k_0	q_2*k_1	q_2*k_2	❌
squeezy(q_3)	q_3*k_0	q_3*k_1	q_2*k_2	q_3*k_3

How to calculate Attention?

$$\text{Attention}(Q, K, V) = \boxed{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})}V$$

weight

Every token has its own **q, k, v**:

🙋‍♀️q: query-------> “What am I looking for?”

🪪k: key----------->”A brief intro about me”

📝v: value-------->”The real info of me”

q*k: “How much does your info match my need? ”

softmax(q_0*k_0, q_1*k_1, …, q_n*k_n**)**: attention weight

softmax(...)*v: context summary

	Easy	peasy	lemon	squeezy
q:	q_0	q_1	q_2	q_3
k:	k_0	k_1	k_2	k_3
v:	v_0	v_1	v_2	v_3

2️⃣**softmax**(q*k), it might be:

q \ k	**Easy(k_0)**	**peasy(k_1)**	**lemon(k_2)**	**squeezy(k_3)**
Easy(q_0)	1.0	❌	❌	❌
peasy(q_1)	0.4	0.6	❌	❌
lemon(q_2)	0.3	0.2	0.5	❌
squeezy(q_3)	0.2	0.1	0.3	0.4

How to calculate Attention?

$$\text{Attention}(Q, K, V) = \underbrace{\boxed{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)}}_{\text{weight}} V$$

Every token has its own **q, k, v**:

🙋‍♀️q: query-------> “What am I looking for?”

🪪k: key----------->”A brief intro about me”

📝v: value-------->”The real info of me”

q*k: “How much does your info match my need? ”

softmax(q_0*k_0, q_1*k_1, …, q_n*k_n**)**: attention weight

softmax(...)*v: context summary

	Easy	peasy	lemon	squeezy
q:	q_0	q_1	q_2	q_3
k:	k_0	k_1	k_2	k_3
v:	v_0	v_1	v_2	v_3

3️⃣softmax(q*k)***v**, it might be:

q \ k	**Easy(k_0)**		**peasy(k_1)**		**lemon(k_2)**		**squeezy(k_3)**
Easy(q_0)	1.0*v_0		❌		❌		❌
peasy(q_1)	0.4*v_0	+	0.6*v_1		❌		❌
lemon(q_2)	0.3*v_0	+	0.2*v_1	+	0.5*v_2		❌
squeezy(q_3)	0.2*v_0	+	0.1*v_1	+	0.3*v_2	+	0.4*v_3

How to calculate Attention?

$$\text{Attention}(Q, K, V) = \boxed{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})}V$$

weight

3 softmax(q*k)***v**, it might be:

q \ k	**Easy(k_0)**		**peasy(k_1)**		**lemon(k_2)**		**squeezy(k_3)**
Easy(q_0)	1.0*v_0		❌		❌		❌
peasy(q_1)	0.4*v_0	+	0.6*v_1		❌		❌
lemon(q_2)	0.3*v_0	+	0.2*v_1	+	0.5*v_2		❌
squeezy(q_3)	0.2*v_0	+	0.1*v_1	+	0.3*v_2	+	0.4*v_3

🤩What we just did:

Easy peasy lemon ❓

lemon's
Attention score

How to calculate Attention?

🤩What we just did:

Transformer(Attention) ✅

Easy ❓

Easy's
Attention score

Easy peasy ❓

peasy's
Attention score

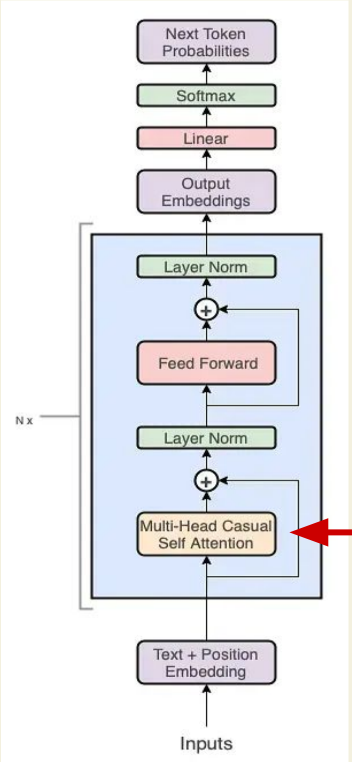
Easy peasy lemon ❓

lemon's
Attention score

Easy peasy lemon squeezy ❓

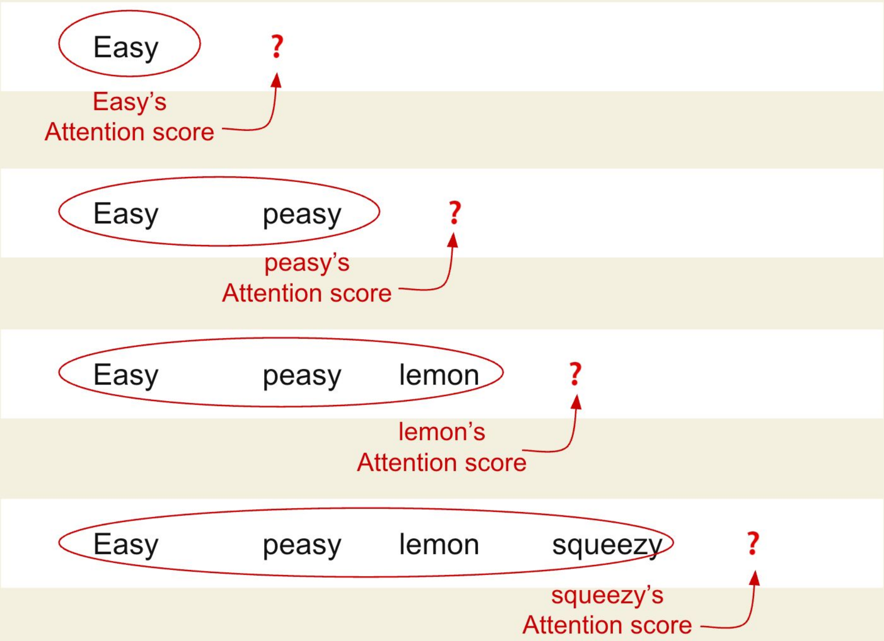
squeezy's
Attention score

To be continued…



Decoder Only(e.g. GPT-like)

🤩What we just did:



1. Motivation
2. Attention as context summary
3. How to calculate Attention?
4. **Further map**
5. Conclusion

Further map

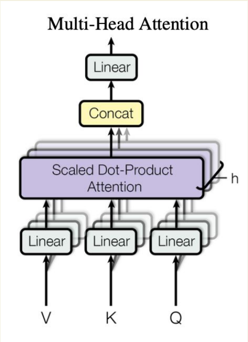
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\boxed{\sqrt{d_k}}}\right)V$$

🙋‍♀️Where does q, k, v come from❓

🙋‍♂️Why we scale by sqrt(d_k) exactly❓

🙋What is Multi-Head Attention❓

🙋What are the rest of the Transformer❓



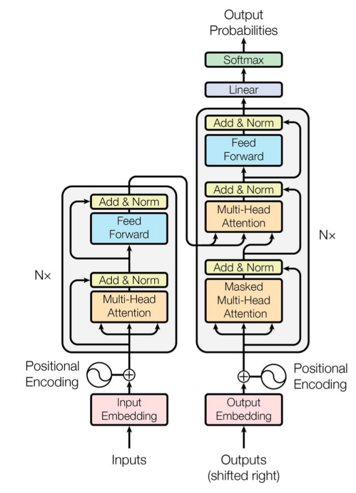


Figure 1: The Transformer - model architecture.

Further map

🙋‍♀️Why we scale by sqrt(d_k) exactly❓
🙋‍♂️What is Multi-Head Attention❓
🙋What are the rest of Transformer❓

: If you want it, you can have it !!
I left all the secrets of Attention…… in this following Map🗺️:

📺***Deep Dive into LLMs like ChatGPT.*** *By Andrej Karpathy*

📺***Let's build GPT: from scratch, in code, spelled out.*** *By Andrej Karpathy*

📜***Attention Is All You Need***, *Ashish Vaswani et al., 2017, v7*

1. Motivation
2. Attention as context summary
3. How to calculate Attention?
4. Further map
5. **Conclusion**

Conclusion

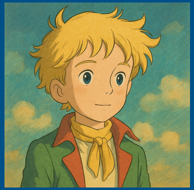
Attention as Context Summary
is all you need

Attention

Easy peasy lemon ❓

Easy peasy lemon ❓

lemon's
Attention score



: It is only with the **Attention** that one can see rightly.

The End

Thank U