



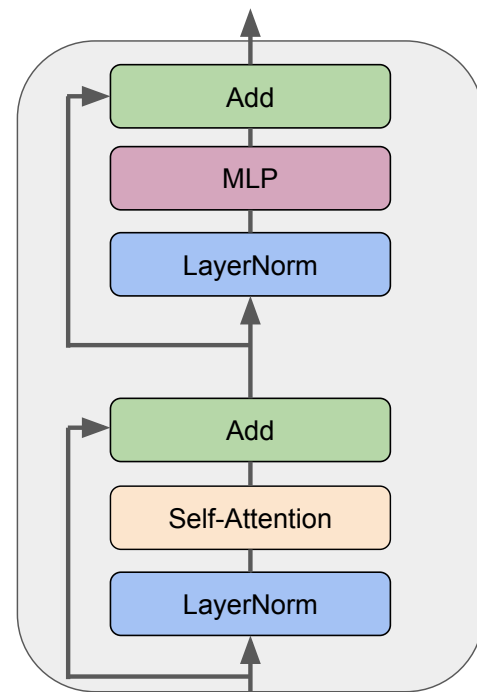
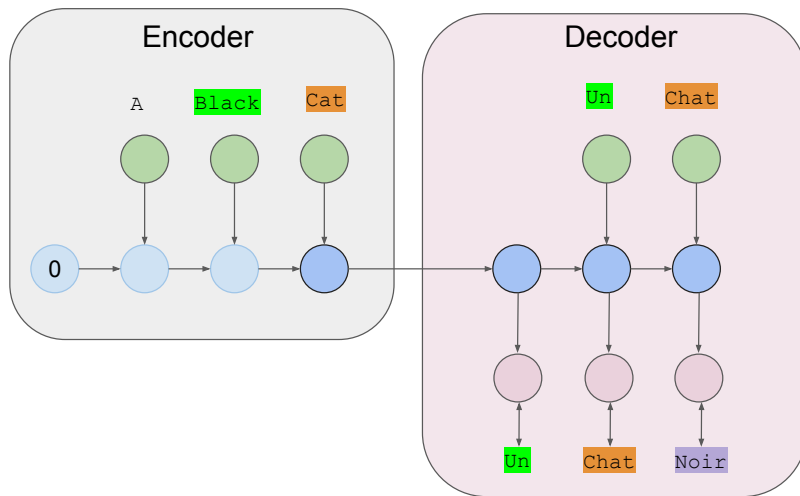
NYU CS-GY 6923

Machine Learning

Prof. Pavel Izmailov

Today

- RNNs beyond language modeling
- Transformers
- Post-Training





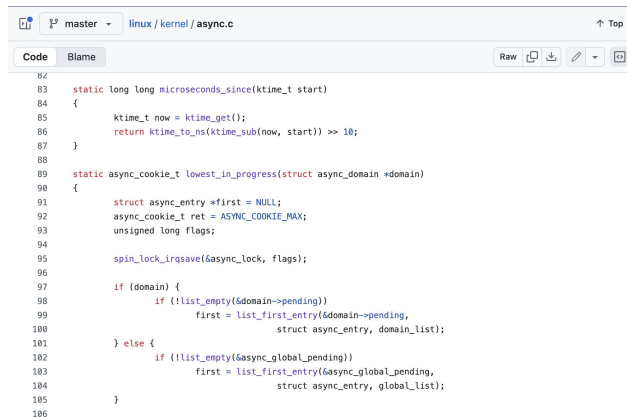
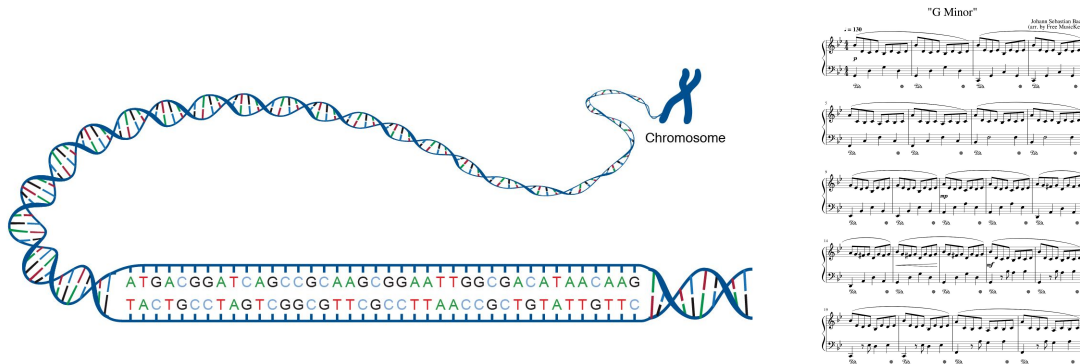
RNNs Beyond Language Modeling

0.36

0.34

Sequence Modeling

Last week we talked about language modeling



4

TERENCE TAO

The random set A we will propose for Theorem 1.3 will then be defined as the truncated random parabola

$$A := \{(x, y) \in \{0, \dots, n-1\}^2 : (ax+by)^2 = cx+dy+e \bmod p\}. \quad (1.2)$$

The standard truncated parabola $S := \{(x, y) \in \{0, \dots, n-1\}^2 : y = x^2 \bmod p\}$ is essentially the example considered in [16], [4], and is the special case of (1.2) when $a = d = 1$ and $b = c = e = 0$. Geometrically, A is formed by applying a random invertible affine transformation¹ to the parabola $\{(x, x^2) : x \in \mathbb{F}_p\}$ and then restricting to the grid $\{0, \dots, n-1\}^2$. While the construction (1.2) is not nearly as random as a completely random subset of \mathbb{F}_p^2 of density $\asymp 1/p$, we shall see that there will (barely) still be enough “entropy” in the five random parameters a, b, c, d, e for the probabilistic method to be effective². In particular, we avoid the difficult number-theoretic questions of trying to count the number of occurrences of the patterns $\pi_1, \pi_2, \dots, \pi_k$ in the standard truncated parabola S (cf. [5, Problem 2]).

A routine application of the second moment method reveals that A usually has the “right” cardinality up to acceptable errors:

Lemma 1.5. *With probability at least 0.9 (say), the set A has cardinality $n^2/p + O(\sqrt{n})$. In particular, for n large enough, the cardinality of A is $\asymp n$ with probability at least 0.9 .*

Proof. Let us temporarily remove the non-degeneracy condition (1.1), so that a, b, c, d, e now become independent random variables. It is clear that any point $(x, y) \in \{0, \dots, n-1\}^2$ will now lie in A with probability $1/p$, just from the randomness of e alone. In fact, any two distinct points $(x, y), (x', y') \in \{0, \dots, n-1\}^2$ will both lie in A with a joint probability of $1/p^2$, from the randomness of c, d, e (since $(x, y, 1)$ and $(x', y', 1)$ are linearly independent in \mathbb{F}_p^3); thus the events $\{(x, y) \in A\}$ are pairwise independent. This implies that the cardinality of A has mean $n^2/p \asymp n$ and variance $O(n^2/p) = O(n)$, and hence from Chebyshev’s inequality, A will have cardinality $n^2/p + O(\sqrt{n})$ with probability at least 0.95 (say). Conditioning to the event (1.1), we obtain the claim. \square

Sequence Modeling

How do we model a time series?

+49.77 (35.98%) ↑ year to date

Closed: Nov 6, 7:59 PM EST • [Disclaimer](#)

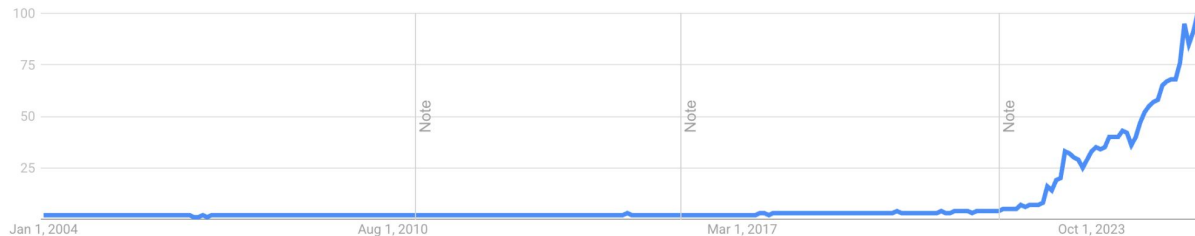
After hours 189.68 +1.60 (0.85%)



Interest over time ?

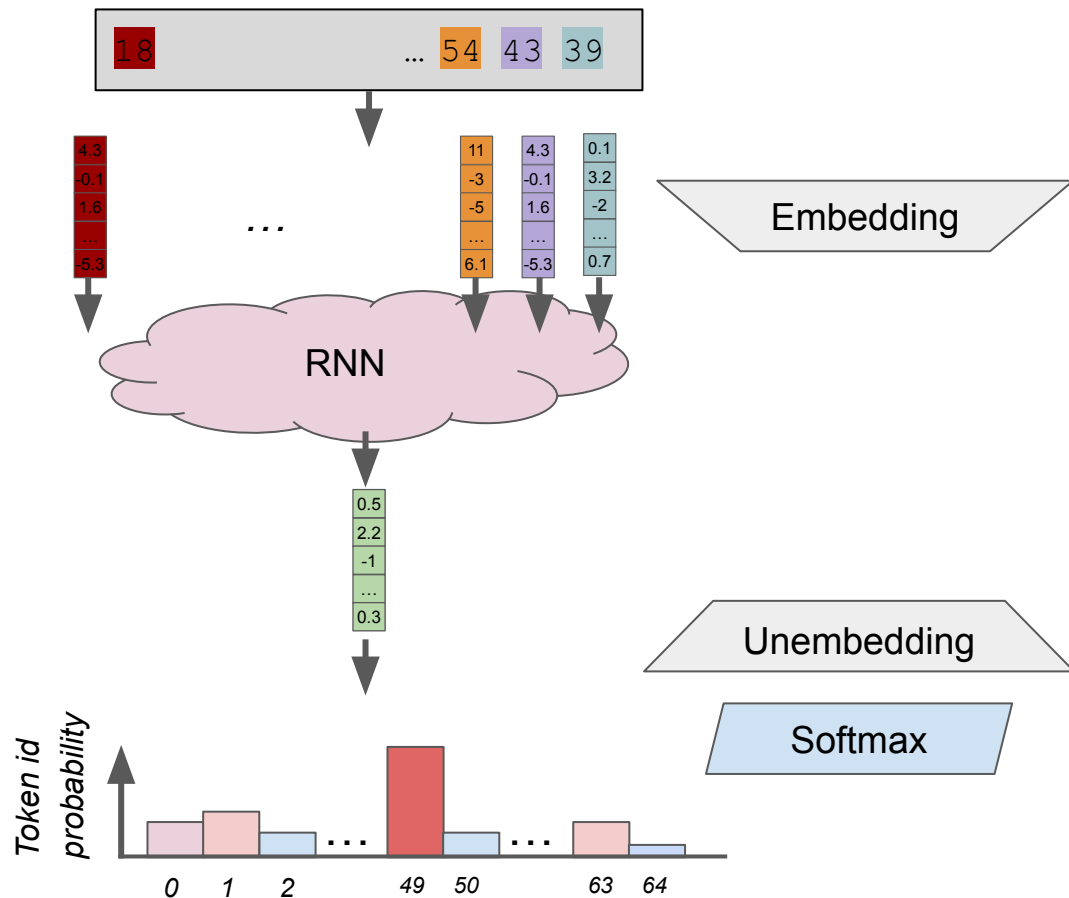
Google Trends

Searches for “AI”



Sequence Modeling

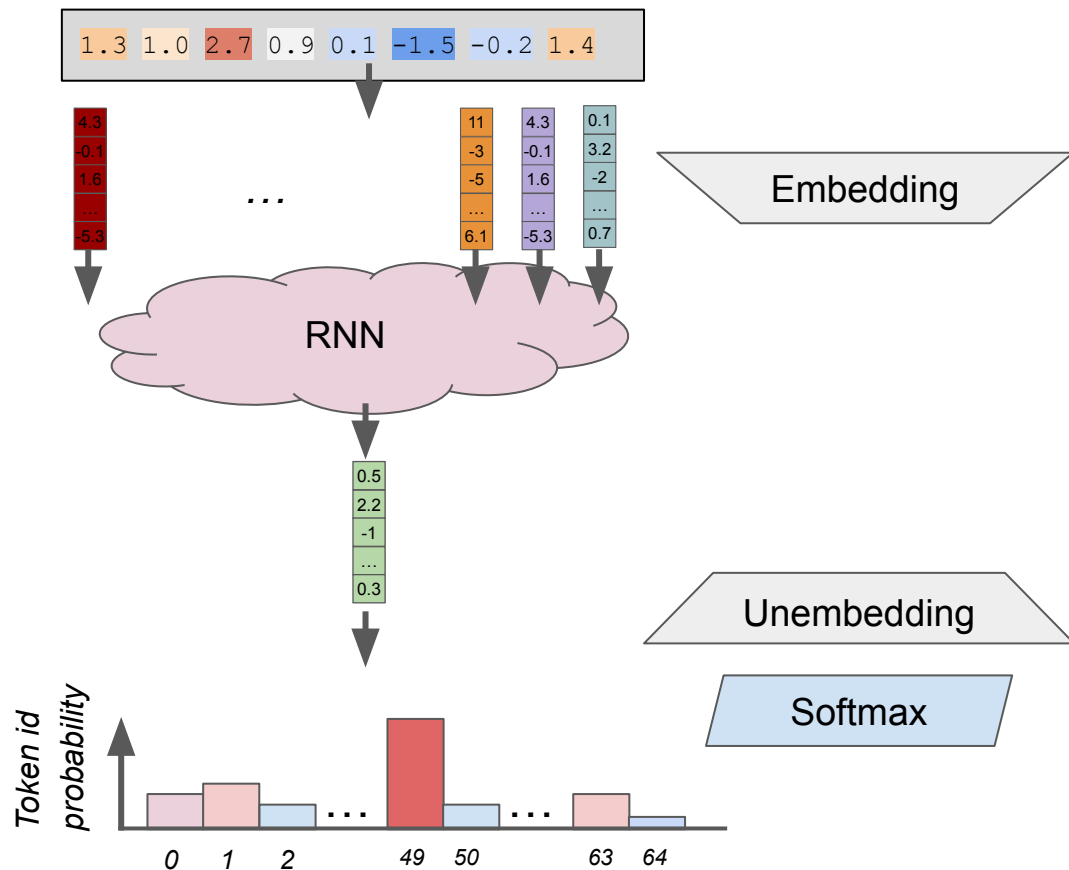
Question: What do we need to change?



Sequence Modeling

Tokens \rightarrow numbers

Question: What do we need to change?

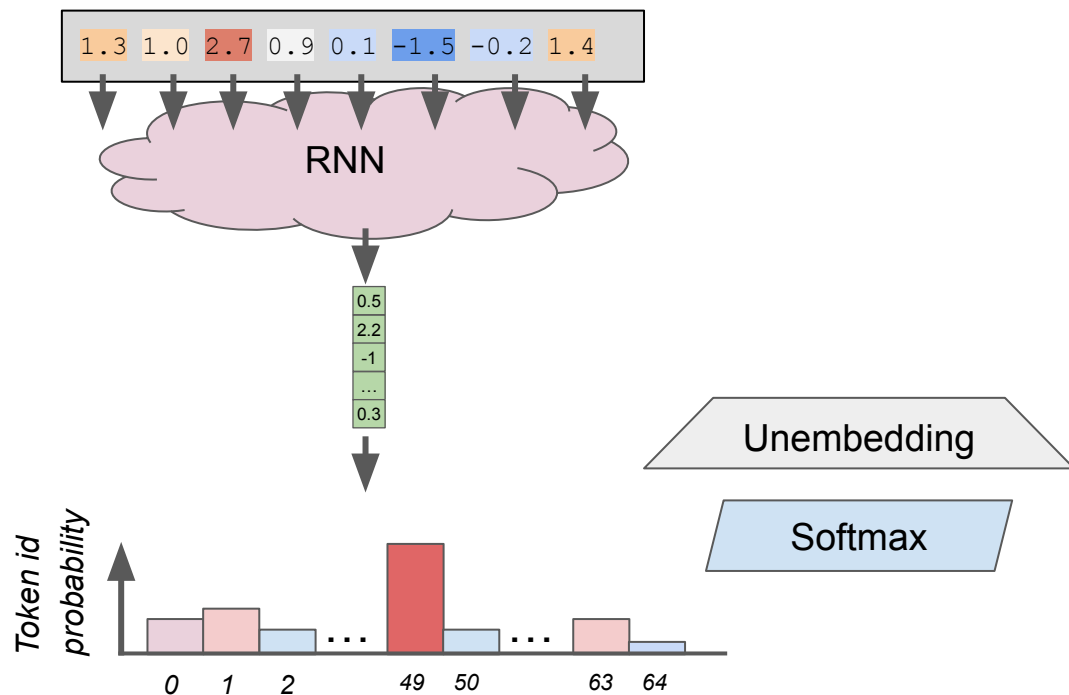


Sequence Modeling

Tokens \rightarrow numbers

We can remove the embedding layer

Question: What do we need to change?



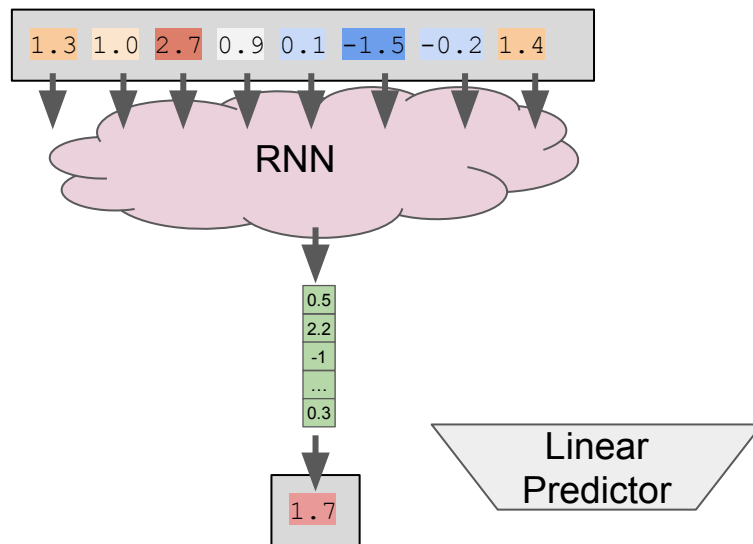
Sequence Modeling

Tokens \rightarrow numbers

We can remove the embedding layer

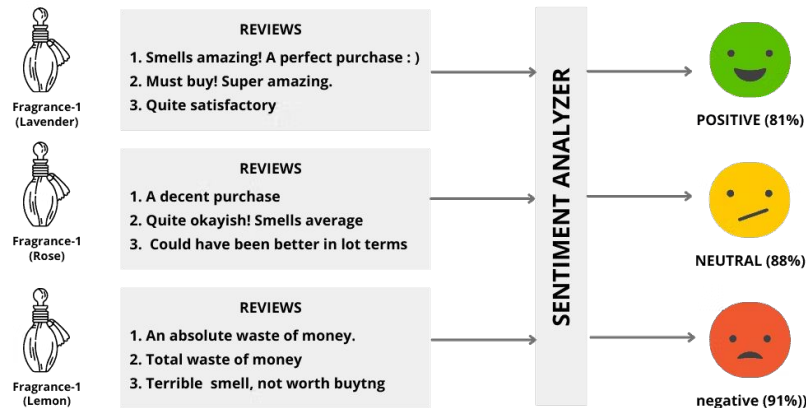
Question: What do we need to change?

Unembedding is now just a linear layer with one output, and we use regression loss

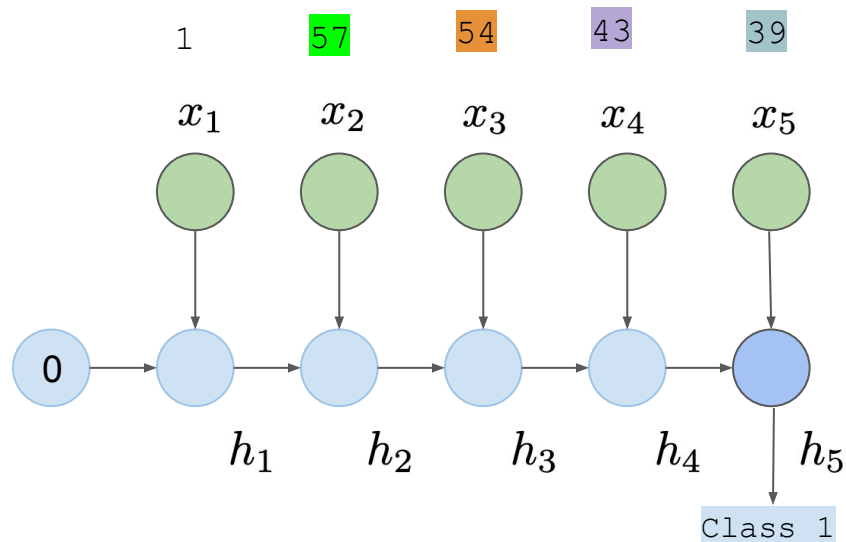


Sequence Classification

- Sentiment analysis: is this a positive review?
- Language detection: which language is this?
- Does this python program compile?
- ...

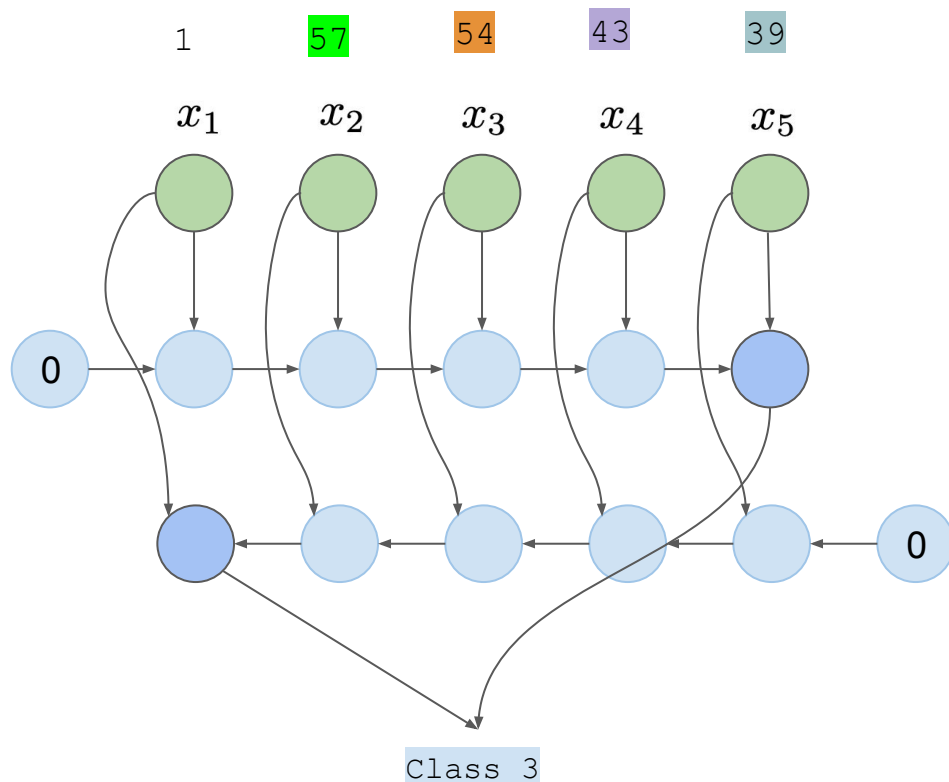


Sequence Classification



We can just use the hidden representation for the last token to make predictions

Sequence Classification



We can generalize the idea to
bidirectional RNNs: use the two
nodes with full context

Sequence to Sequence

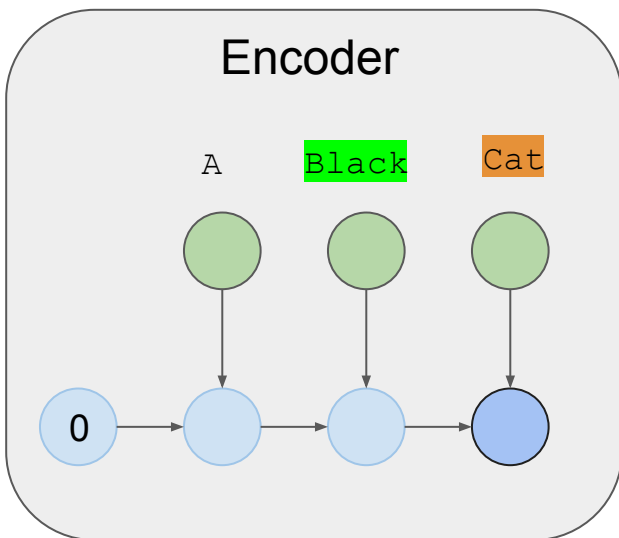
- Machine Translation: Translating text from one language to another
- Text Summarization
- Speech Recognition
- Image Captioning



Captioning Model

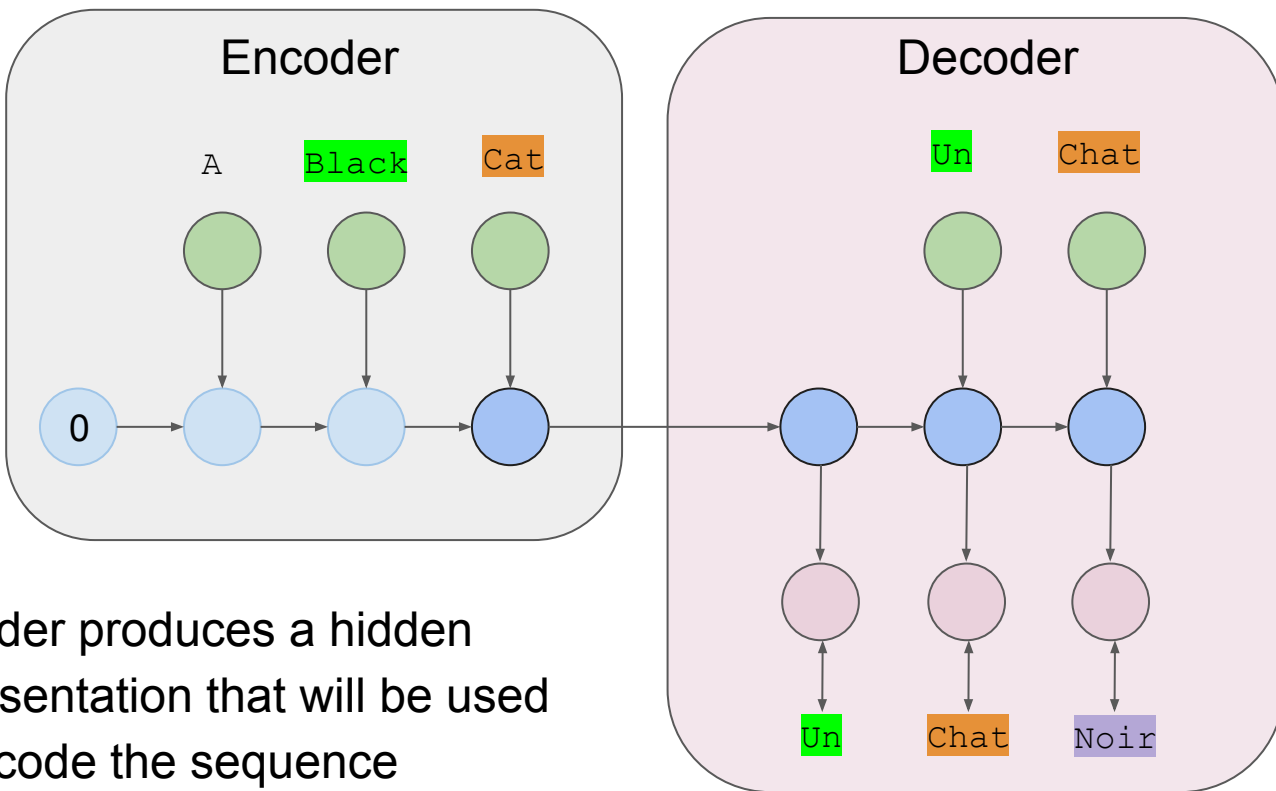
A happy dog is standing in the ocean

Sequence to Sequence



Encoder produces a hidden representation that will be used to decode the sequence

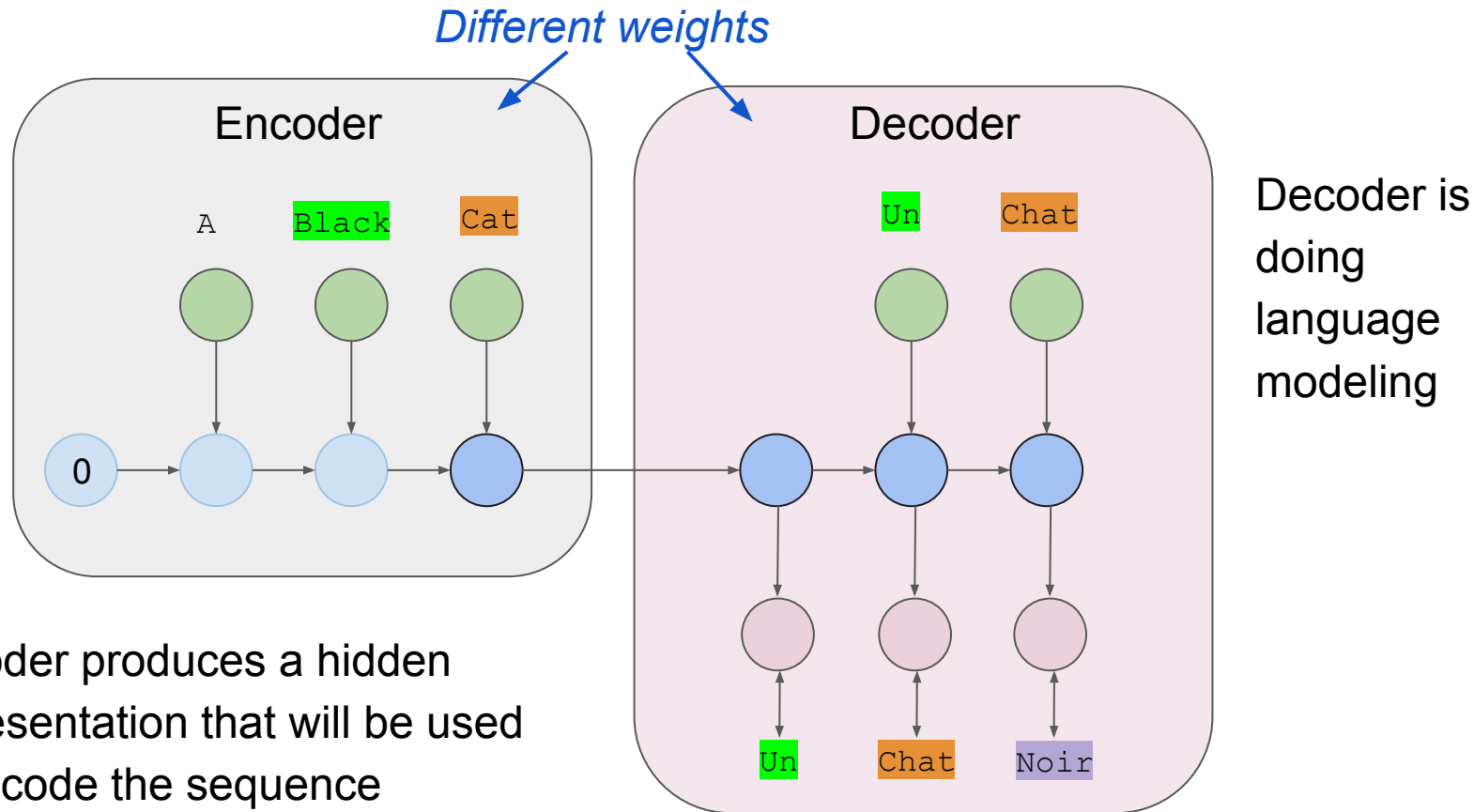
Sequence to Sequence



Decoder is
doing
language
modeling

Encoder produces a hidden
representation that will be used
to decode the sequence

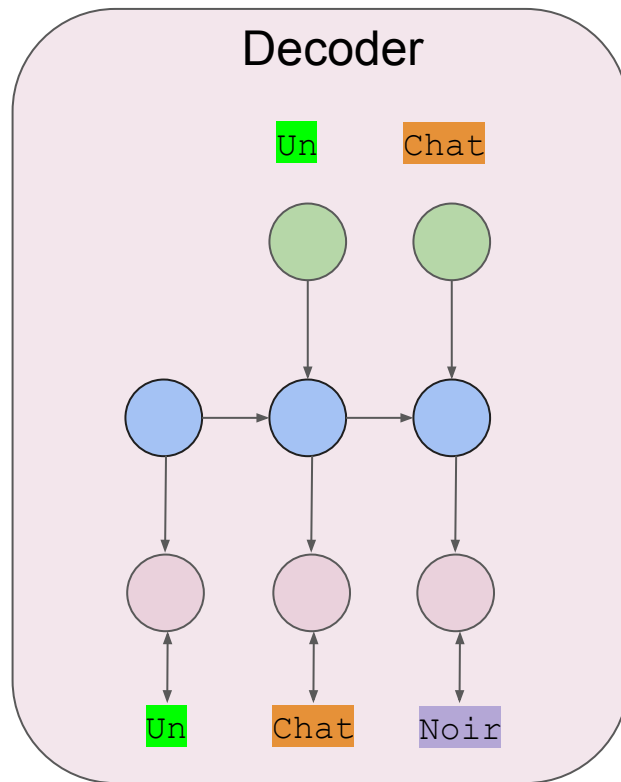
Sequence to Sequence



Encoder produces a hidden representation that will be used to decode the sequence

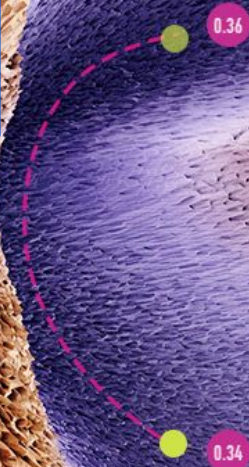
Sequence to Sequence

Last week, we were looking at
decoder-only models



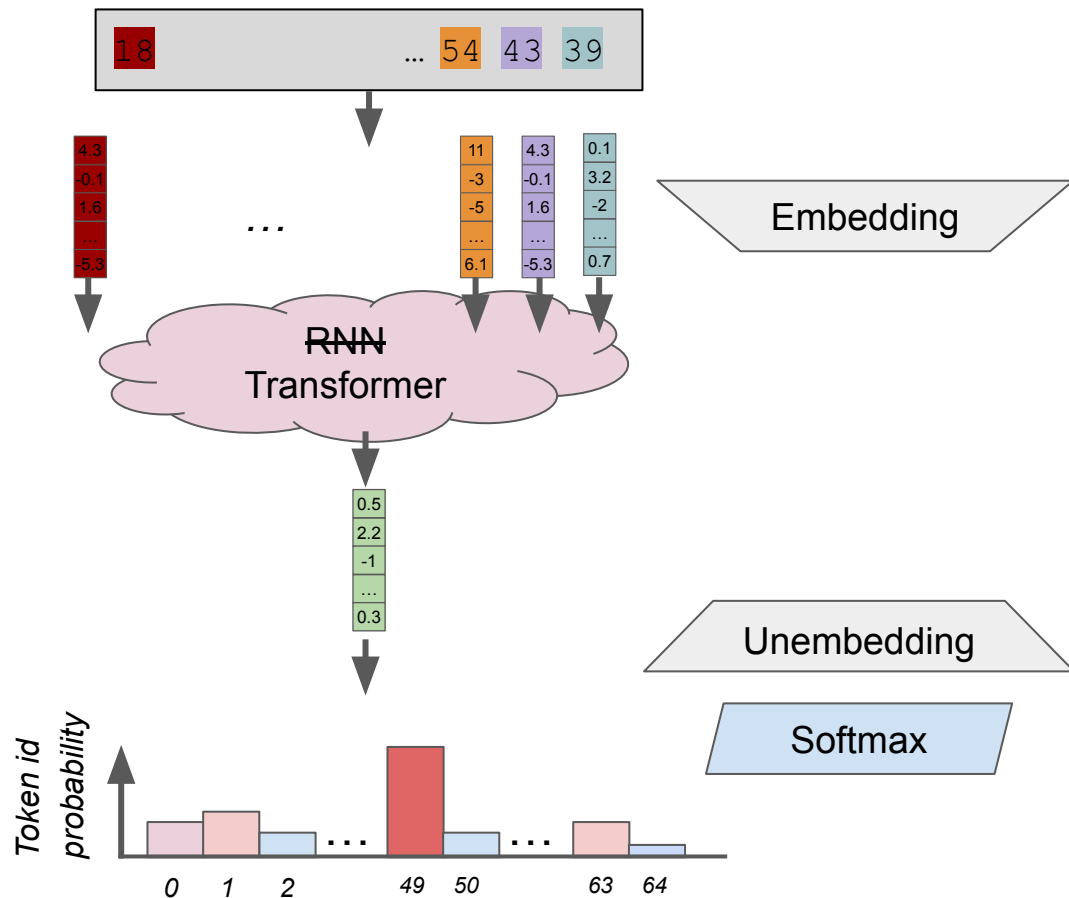
Decoder is
doing
language
modeling

Transformers

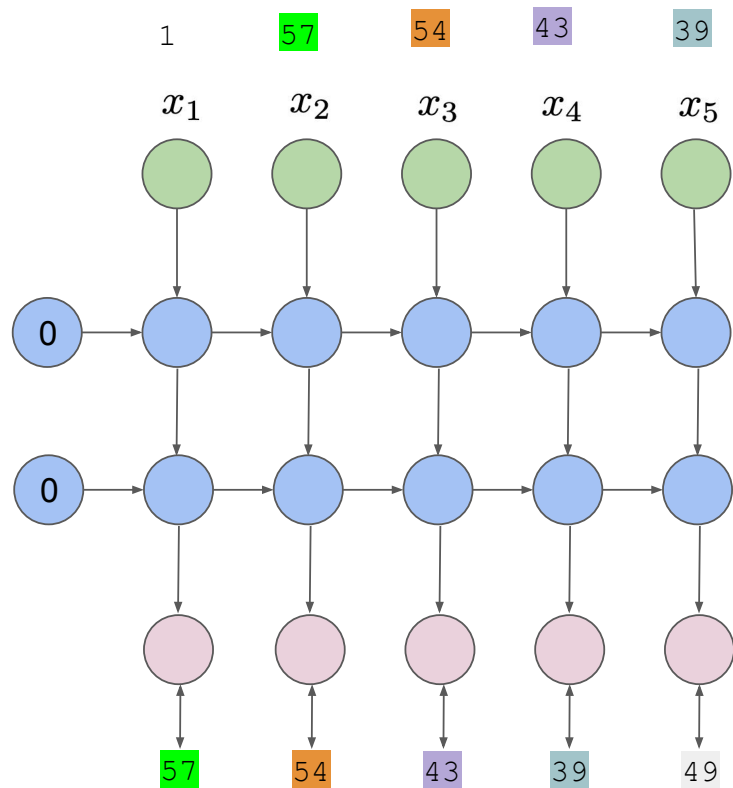


Transformers

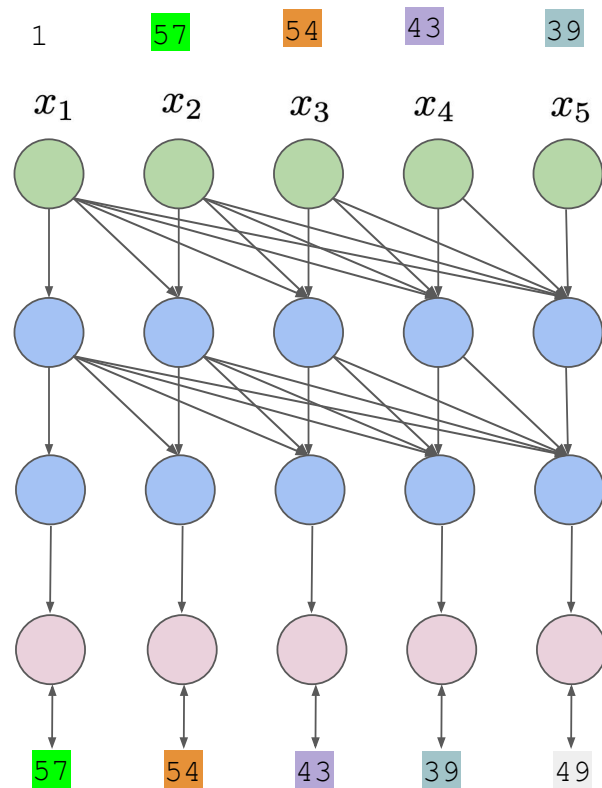
We will go back to our decoder-only language modeling setting.



Transformers

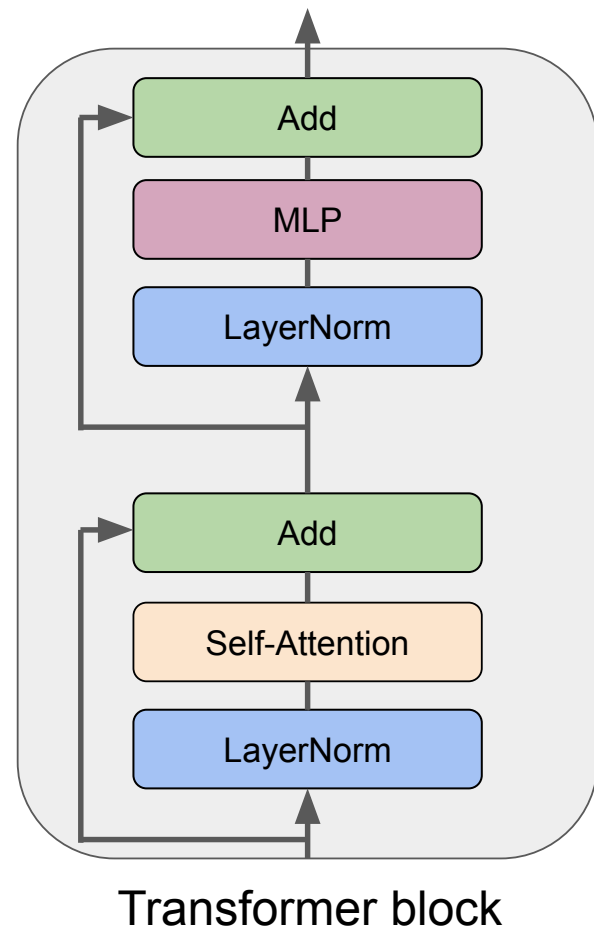


RNN



Transformer

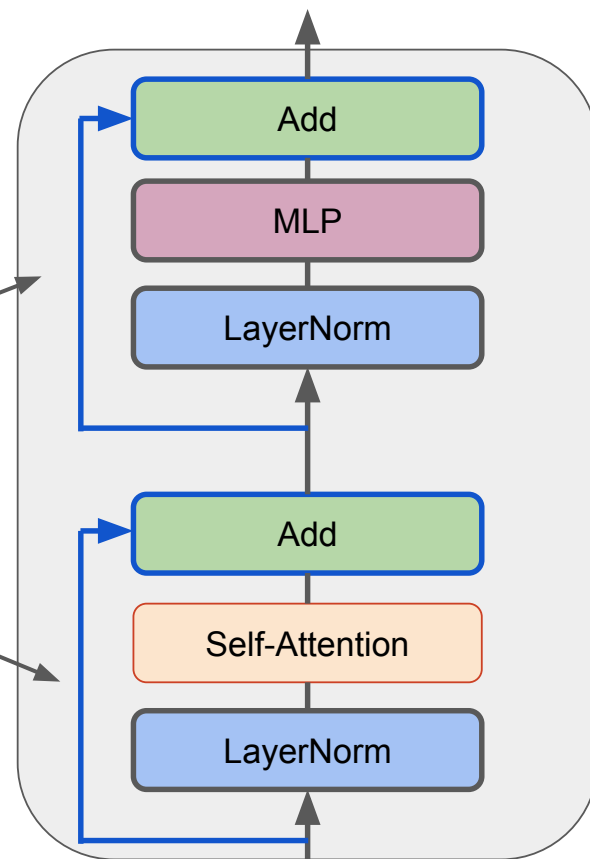
Transformers



Transformers

We know almost all of the components!

Skip Connections



Transformer block

Layernorm

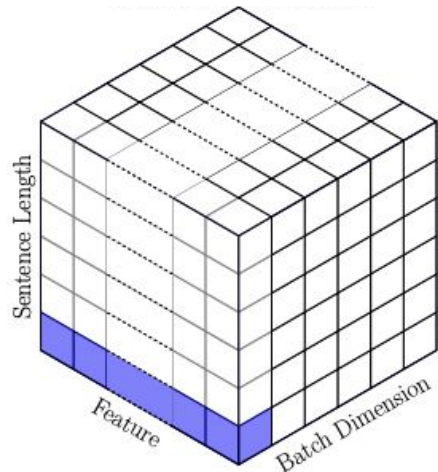
Diagram illustrating the Layer Normalization (Layernorm) operation:

Input x is processed to produce the Layernorm output y using the formula:

$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta$$

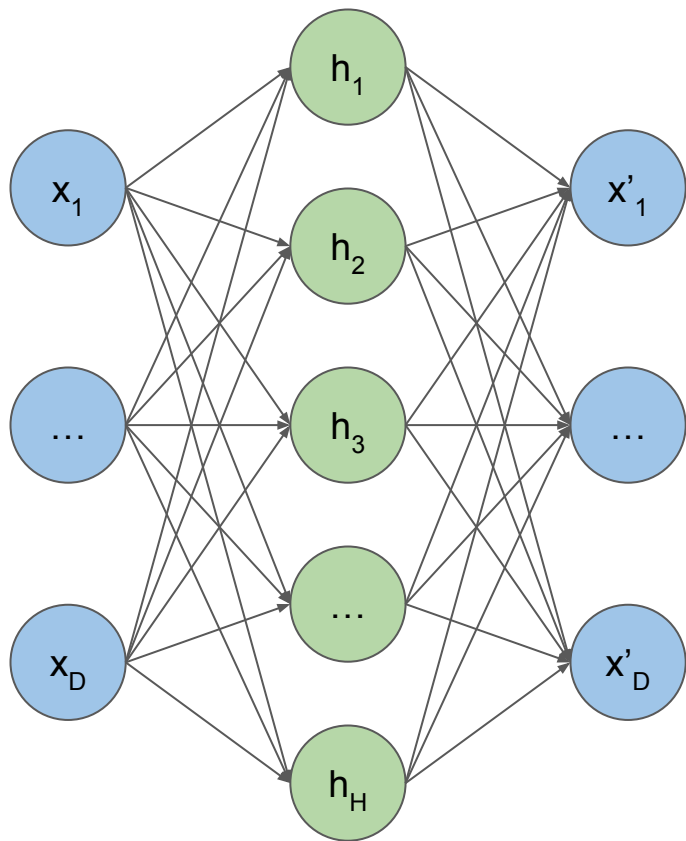
Where:

- $\mathbb{E}[x]$ is the mean of the input.
- $\text{Var}[x]$ is the variance of the input.
- ϵ is a small constant added to the denominator for numerical stability.
- γ is the trainable scale.
- β is the trainable shift.



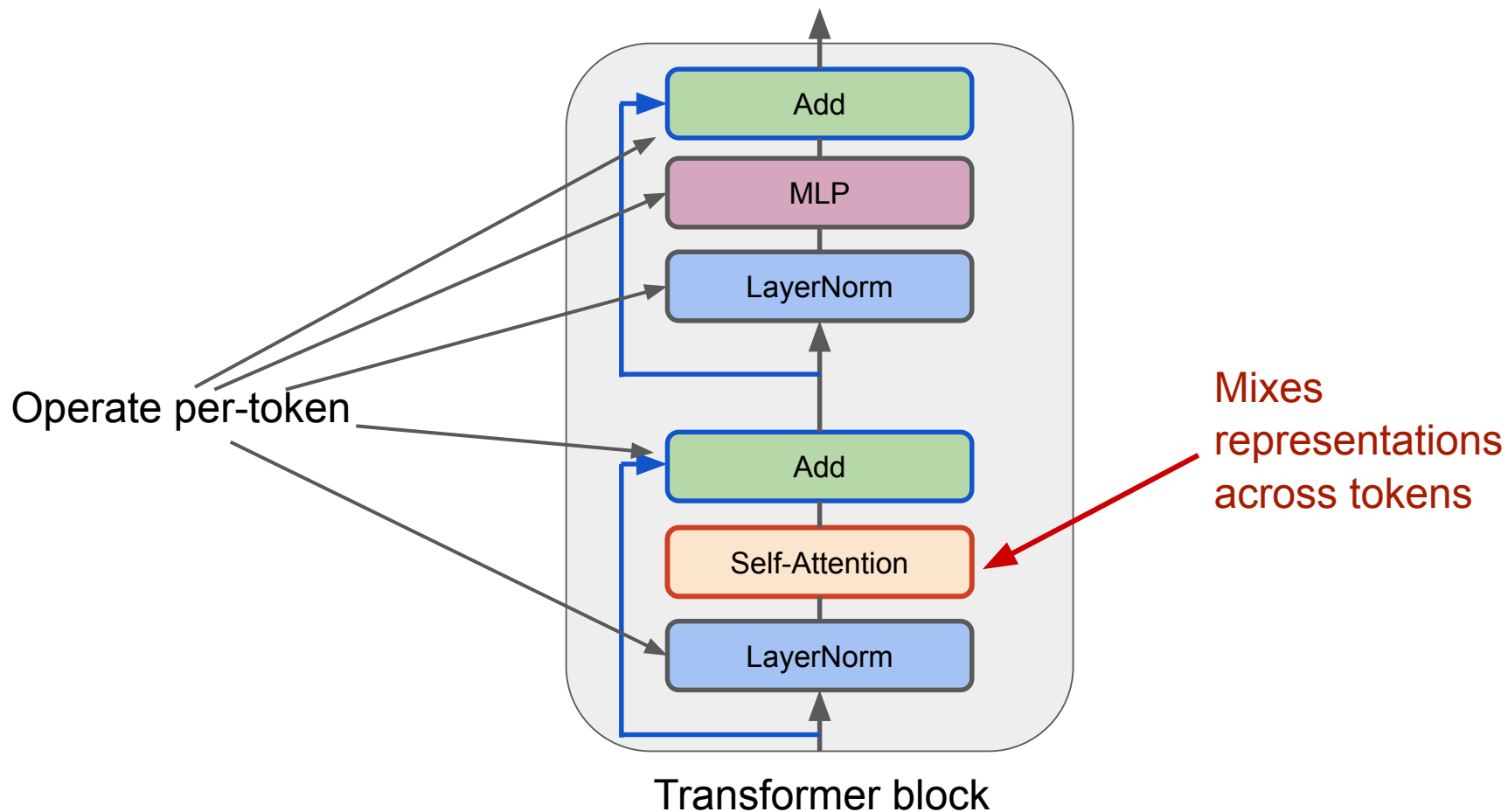
- We normalize the activations per token
- Mean and variance are computed over all dimensions of the input

MLP

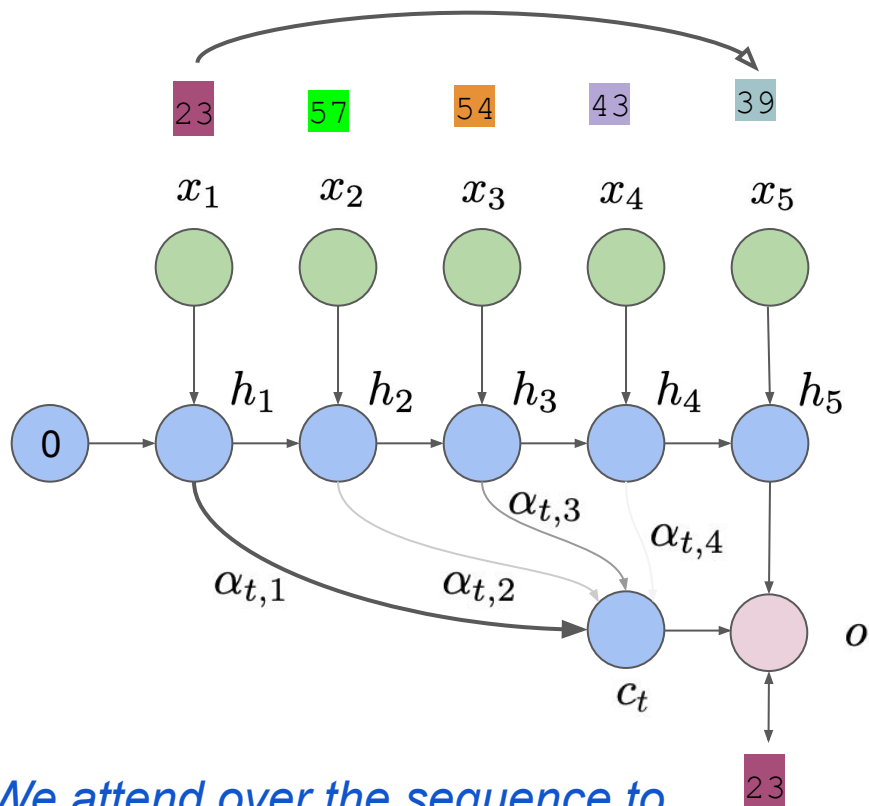


- Same MLP as we discussed before
- Transforms the features of each token separately

Transformers



RNN Variants: Attention



We attend over the sequence to look for relevant information!

We will try to fix the long-range dependence issue.

$$e_{t,i} = \text{score}(h_t, h_i), \quad i < t$$
$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j < t} \exp(e_{t,j})}$$

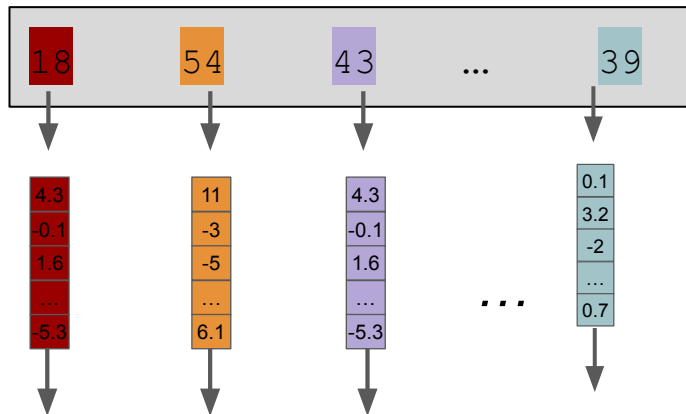
Compute *attention weights* for each previous token

$$c_t = \sum_{i < t} \alpha_{t,i} h_i$$

Use a weighted sum of all representations to make the prediction

Self-Attention

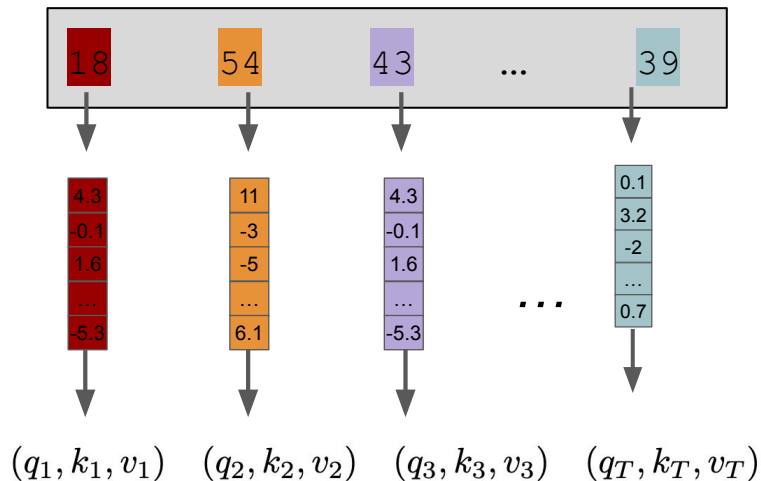
We want a transformation that can mix representations over multiple tokens.



We have a sequence of hidden representations

Self-Attention

We want a transformation that can mix representations over multiple tokens.

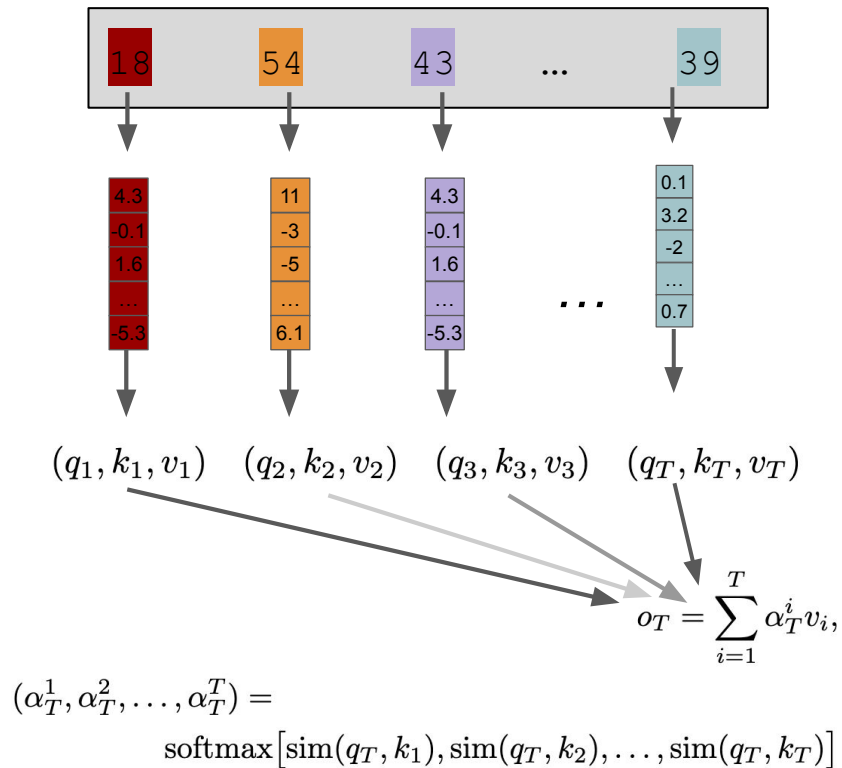


We have a sequence of hidden representations

(q=query, k=key, v=value)

Self-Attention

We want a transformation that can mix representations over multiple tokens.



We have a sequence of hidden representations

(q=query, k=key, v=value)

Output is a mix of values with weights based on similarity(query_T, key_i)

Key-Query-Value example

Item	Key (car features)	Value (car mpg)
Honda Civic	HP: 158, Engine: 2.0L, Year: 2020	33
Ford Mustang	HP: 310, Engine: 5.0L, Year: 2019	18
Toyota Prius	HP: 121, Engine: 1.8L, Year: 2021	52
...
Chevy Silverado	HP: 285, Engine: 4.3L, Year: 2020	20

Item	Query
Mystery Sedan	HP: 150, Engine: 1.9L, Year: 2020

Key-Query-Value example

Item	Key (car features)	Value (car mpg)	Similarity(key, query)
Honda Civic	HP: 158, Engine: 2.0L, Year: 2020	33	0.92
Ford Mustang	HP: 310, Engine: 5.0L, Year: 2019	18	0.31
Toyota Prius	HP: 121, Engine: 1.8L, Year: 2021	52	0.78
...	
Chevy Silverado	HP: 285, Engine: 4.3L, Year: 2020	20	0.35

Item	Query
Mystery Sedan	HP: 150, Engine: 1.9L, Year: 2020

Key-Query-Value example

Item	Key (car features)	Value (car mpg)	Similarity(key, query)
Honda Civic	HP: 158, Engine: 2.0L, Year: 2020	33	0.92
Ford Mustang	HP: 310, Engine: 5.0L, Year: 2019	18	0.31
Toyota Prius	HP: 121, Engine: 1.8L, Year: 2021	52	0.78
...	
Chevy Silverado	HP: 285, Engine: 4.3L, Year: 2020	20	0.35

Item	Query
Mystery Sedan	HP: 150, Engine: 1.9L, Year: 2020

$$o = \sum_{i=1}^T \alpha^i v_i,$$

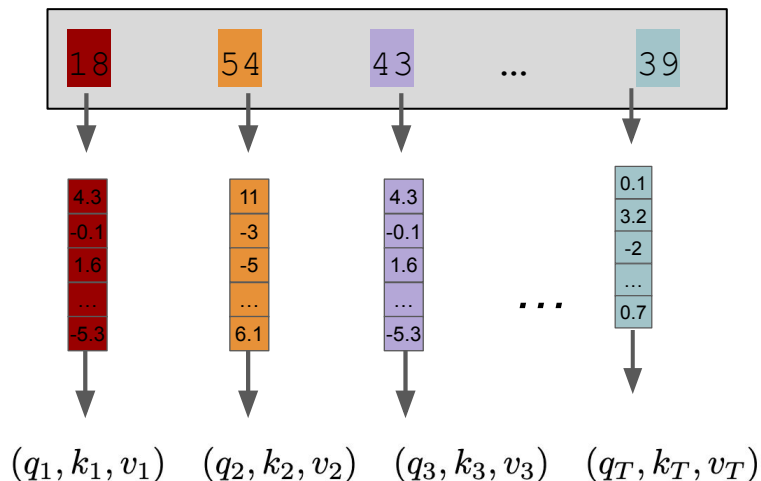
$(\alpha^1, \alpha^2, \dots, \alpha^T) =$
 $\text{softmax}[\text{sim}(q, k_1), \text{sim}(q, k_2), \dots, \text{sim}(q, k_T)]$

o = 34.2 MPG

Key-Query-Value example

For each token:

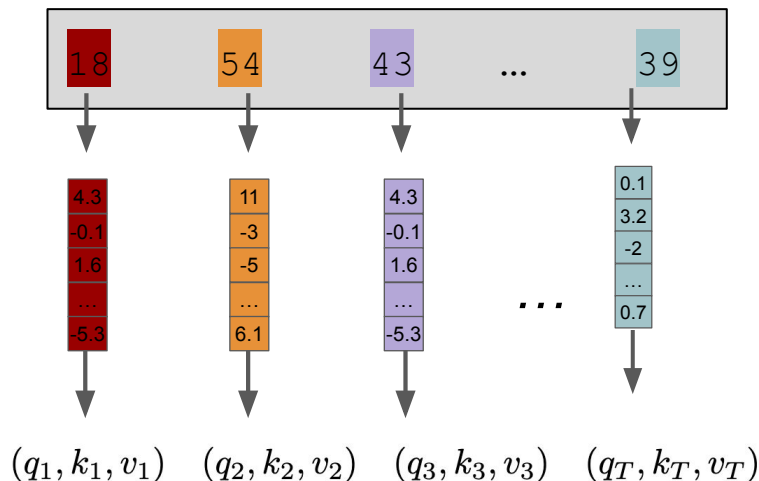
- k: key, a description of the information in this token, used to find this token
- v: value, output associated with the key
- q: query, what we want to find in other tokens



Key-Query-Value example

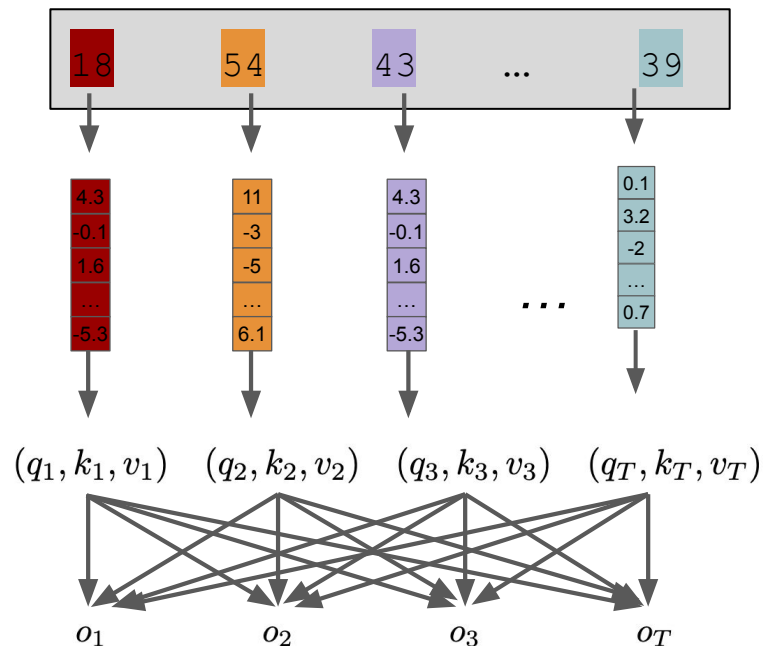
For each token:

- k: key, a description of the information in this token, used to find this token
- v: value, output associated with the key
- q: query, what we want to find in other tokens



Unlike our previous example, in transformers (q, k, v) are abstract and learned by the model; they will not have an interpretable meaning

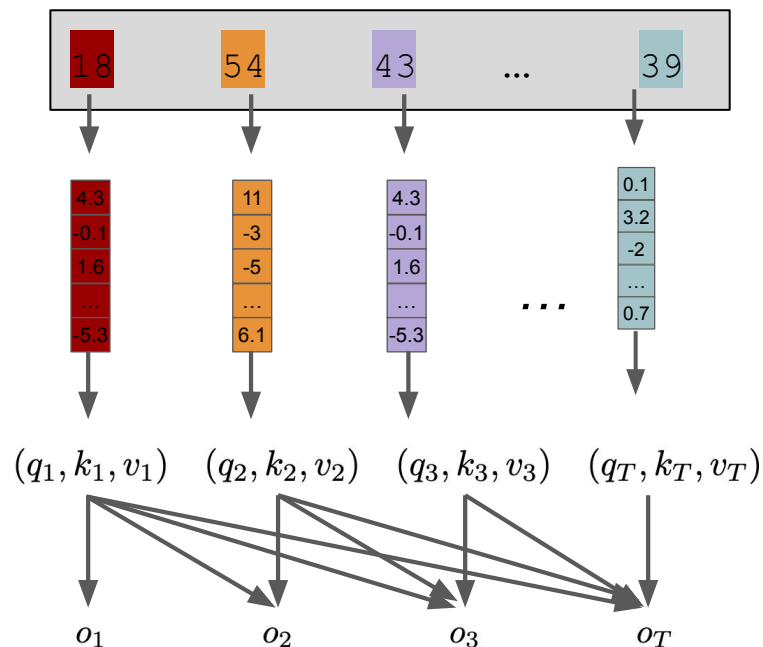
Key-Query-Value attention



We get an output for each token, which depends on all other tokens

$$o_j = \sum_{i=1}^T \alpha_j^i v_i,$$
$$(\alpha_j^1, \alpha_j^2, \dots, \alpha_j^T) = \text{softmax}[\text{sim}(q_j, k_1), \text{sim}(q_j, k_2), \dots, \text{sim}(q_j, k_T)]$$

Key-Query-Value attention



We get an output for each token, which depends on all other tokens

For language modeling we apply a causal mask: outputs can not depend on future tokens

$$o_j = \sum_{i=1}^T \alpha_j^i v_i,$$
$$(\alpha_j^1, \alpha_j^2, \dots, \alpha_j^T) = \text{softmax}[\text{sim}(q_j, k_1), \text{sim}(q_j, k_2), \dots, \text{sim}(q_j, k_T)]$$

Dot-Product Self Attention

$$W_Q x = q$$

x
4.3
-0.1
1.6
...
-5.3

q

$$W_V x = v$$

x
4.3
-0.1
1.6
...
-5.3

v

$$W_K x = k$$

x
4.3
-0.1
1.6
...
-5.3

k

(q, k, v) are all produced as linear transformations of the hidden vector

W_Q , W_K , W_V are learned, shared across all tokens

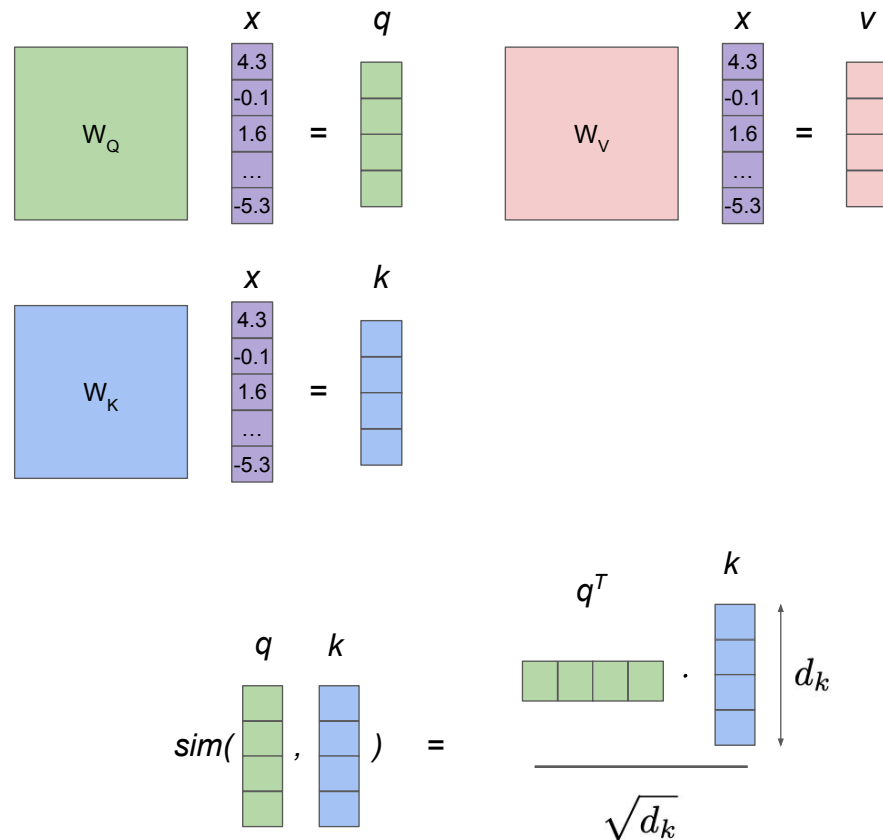
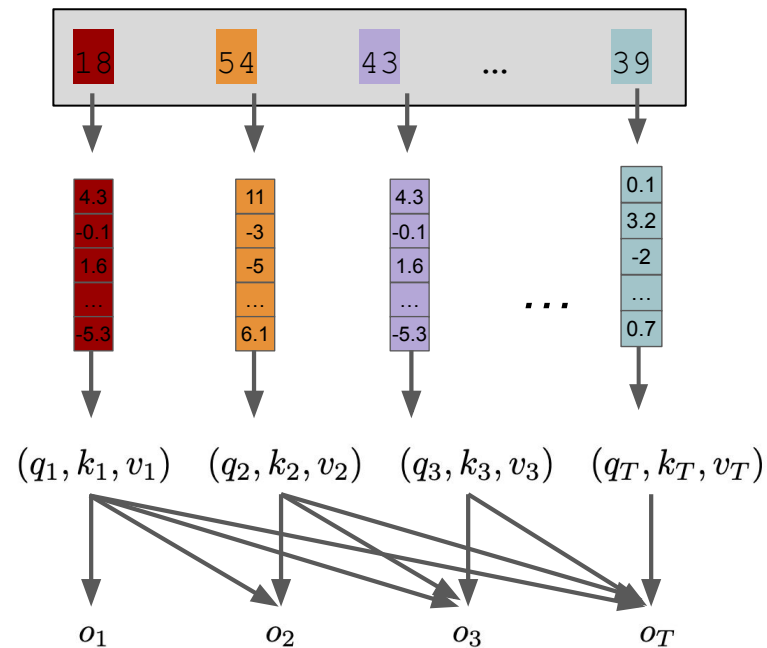
Dot-Product Self Attention

The diagram illustrates the calculation of a similarity score for dot-product self-attention. On the left, a green vertical vector q (4x1) and a blue vertical vector k (4x1) are shown. The similarity function $sim(q, k)$ is applied to these vectors. This is equated to a scaled dot product on the right. The dot product is represented by a green horizontal vector q^T (1x4) multiplied by a blue vertical vector k (4x1). The result of the dot product is divided by the square root of the dimension d_k , which is indicated by a vertical double-headed arrow next to the vector k .

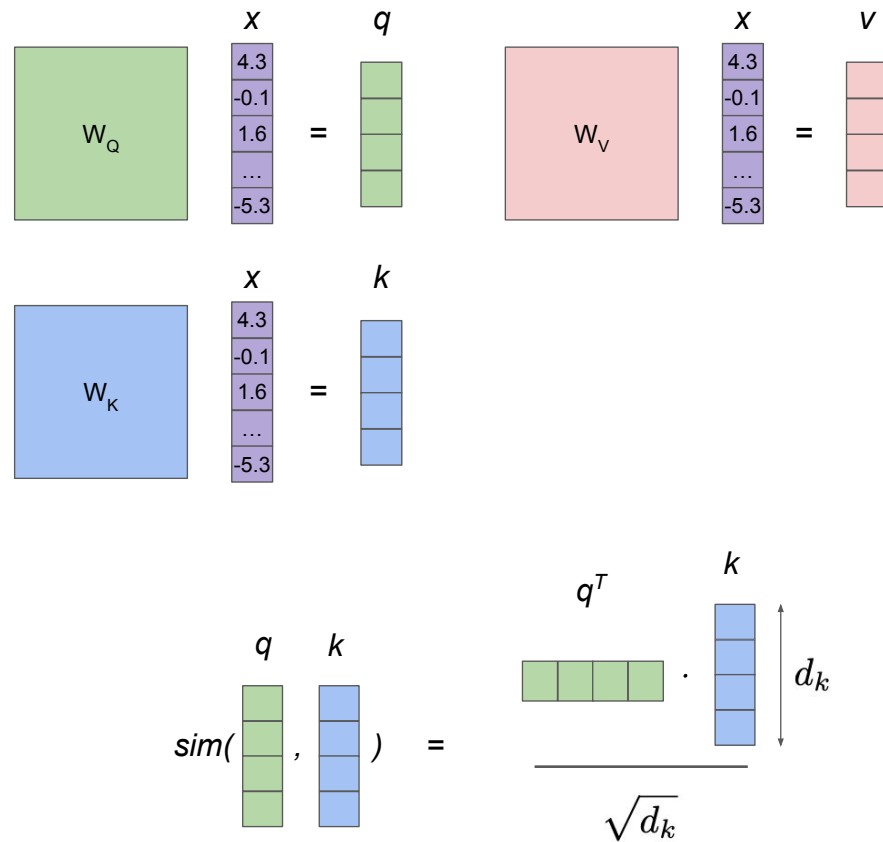
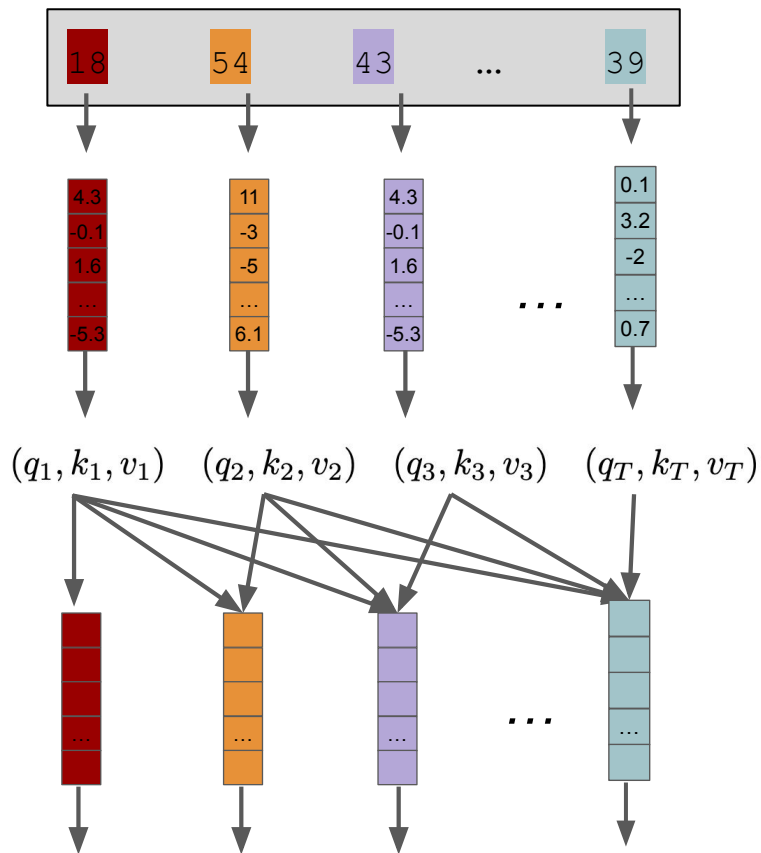
$$sim(q, k) = \frac{q^T \cdot k}{\sqrt{d_k}}$$

Our similarity score is a scaled dot product. We will explain the scale later.

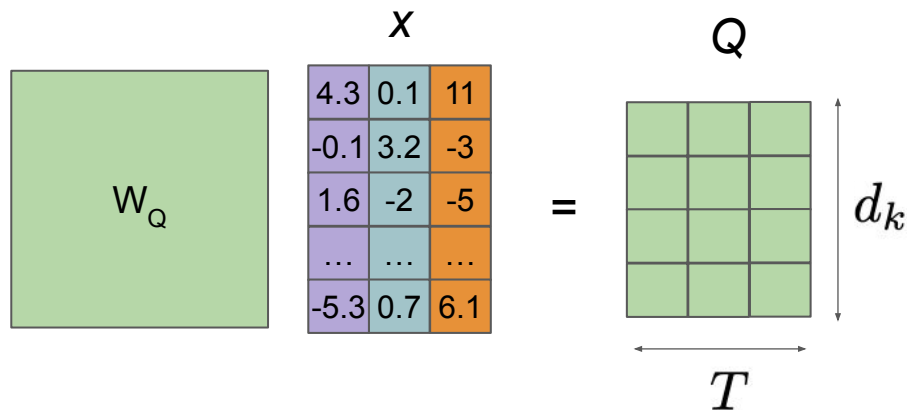
Dot-Product Self Attention



Dot-Product Self Attention

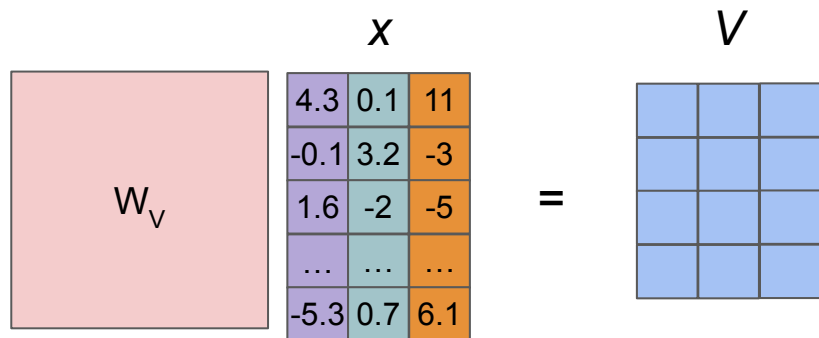
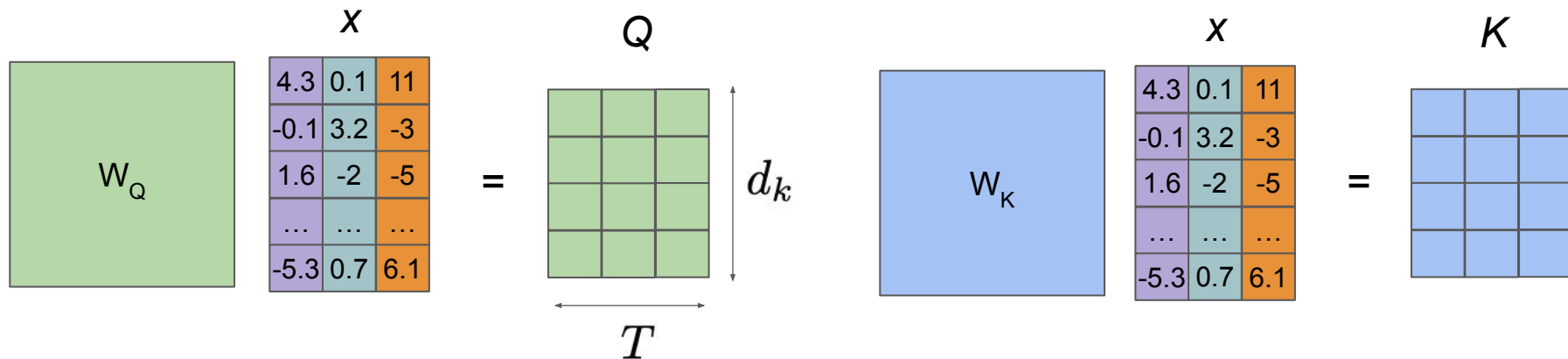


Dot-Product Self Attention



Let's process all T tokens in parallel.

Dot-Product Self Attention



Let's process all T tokens in parallel.

Dot-Product Self Attention

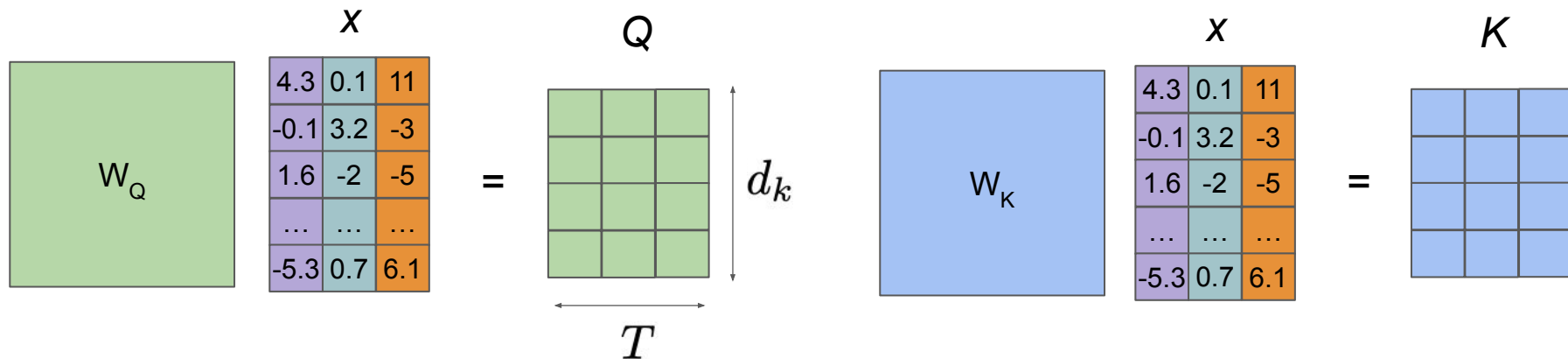


Diagram illustrating the construction of the similarity matrix S from the Query matrix Q^T and the Key matrix K .

The similarity matrix S is computed as:

$$\text{softmax} \left[\frac{Q^T \cdot K}{\sqrt{d_k}} \right] = S$$

The resulting similarity matrix S (yellow 3x3 grid) has dimensions T (horizontal axis) and T (vertical axis).

Then, construct a T by T similarity matrix.

Dot-Product Self Attention

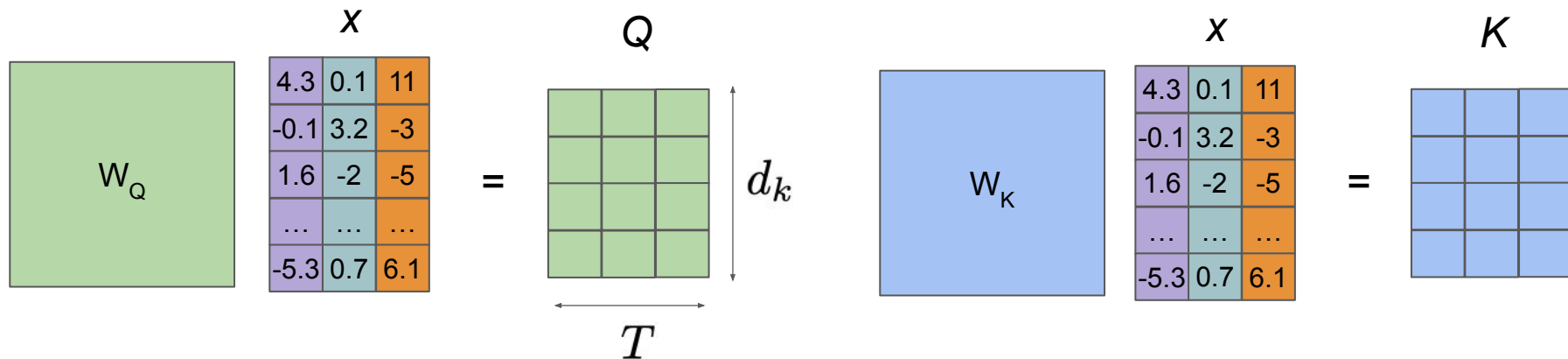


Diagram illustrating the construction of the similarity matrix S and the application of the softmax function.

The similarity matrix S is computed as:

$$\text{softmax} \left[\frac{Q^T K}{\sqrt{d_k}} \right] = S$$

The similarity matrix S is a 5x3 matrix:

S		
1.	0.	0.
0.7	0.3	0.
0.1	0.3	0.6

The similarity matrix S has dimensions T (horizontal) and T (vertical).

Then, construct a T by T similarity matrix.

Causal mask: cannot look at future tokens.

Dot-Product Self Attention

$$\text{softmax} \left[\frac{Q^T \cdot K}{\sqrt{d_k}} \right] = S$$

Diagram illustrating the first step of Dot-Product Self Attention. A query matrix Q^T (green, 3x4) is multiplied by a key matrix K (blue, 4x4). The result is passed through a softmax function to produce a weight matrix S (yellow, 3x3). The weight matrix S is shown with values: $\begin{bmatrix} 1. & 0. & 0. \\ 0.7 & 0.3 & 0. \\ 0.1 & 0.3 & 0.6 \end{bmatrix}$. The dimensions of S are indicated as T (rows) and T (columns).

Outputs are a mix of value vectors with weights given by S .

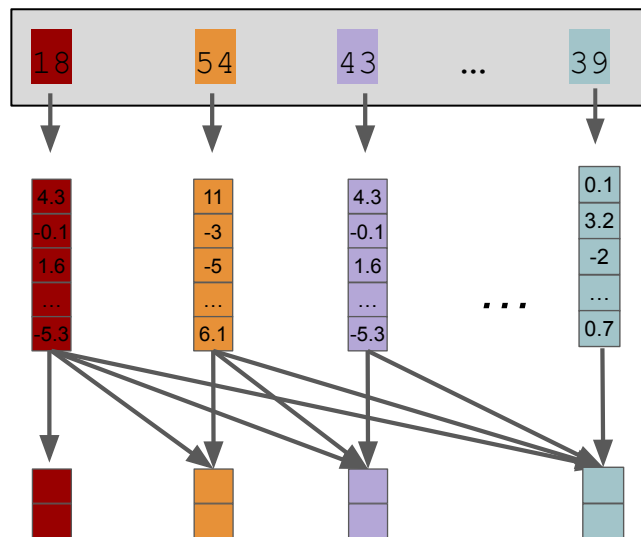
$$W_V \cdot X = V$$

Diagram illustrating the second step of Dot-Product Self Attention. A value matrix X (purple, 5x3) is multiplied by a weight matrix W_V (pink, 5x4) to produce a value matrix V (blue, 4x4). The dimensions of V are indicated as T (rows) and d_k (columns).

$$S \cdot V^T = O$$

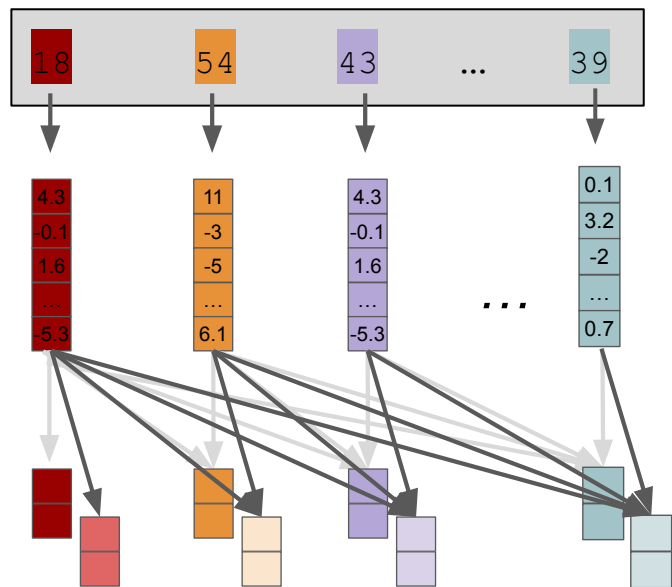
Diagram illustrating the final step of Dot-Product Self Attention. The weight matrix S (yellow, 3x3) is multiplied by the value matrix V^T (blue, 4x4) to produce the output matrix O (pink, 3x4). The dimensions of O are indicated as T (rows) and d_k (columns).

Multi-Head Attention



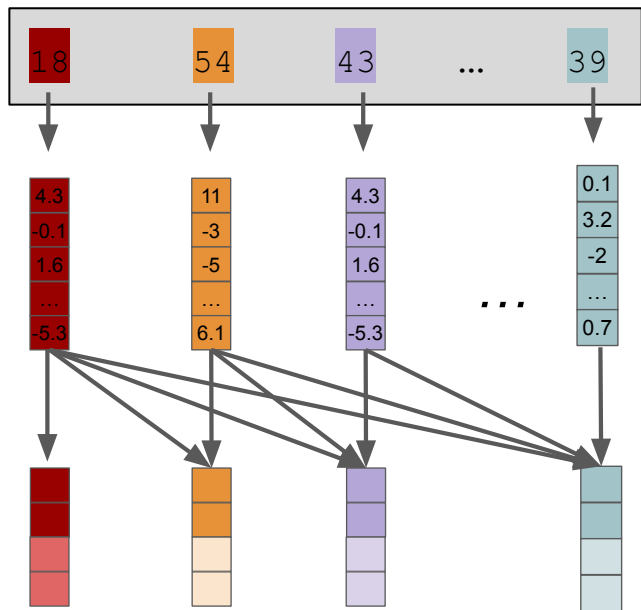
Let's just do attention twice with different projection matrices W_Q , W_K , W_V

Multi-Head Attention



Let's just do attention twice with different projection matrices W_Q , W_K , W_V

Multi-Head Attention



Let's just do attention twice with different projection matrices W_Q , W_K , W_V

We can just concatenate the outputs.

We now have two *attention heads*.

In particular, attention heads will use different attention weights.

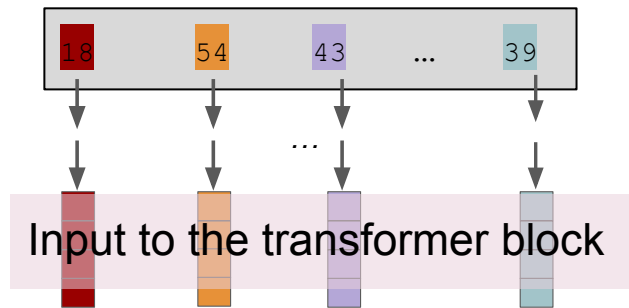
S_1

1.	0.	0.
0.7	0.3	0.
0.1	0.3	0.6

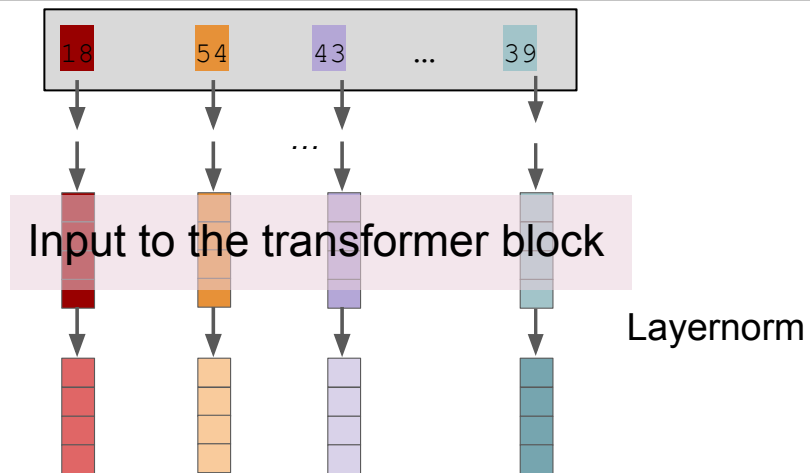
S_2

1.	0.	0.
0.4	0.6	0.
0.8	0.1	0.1

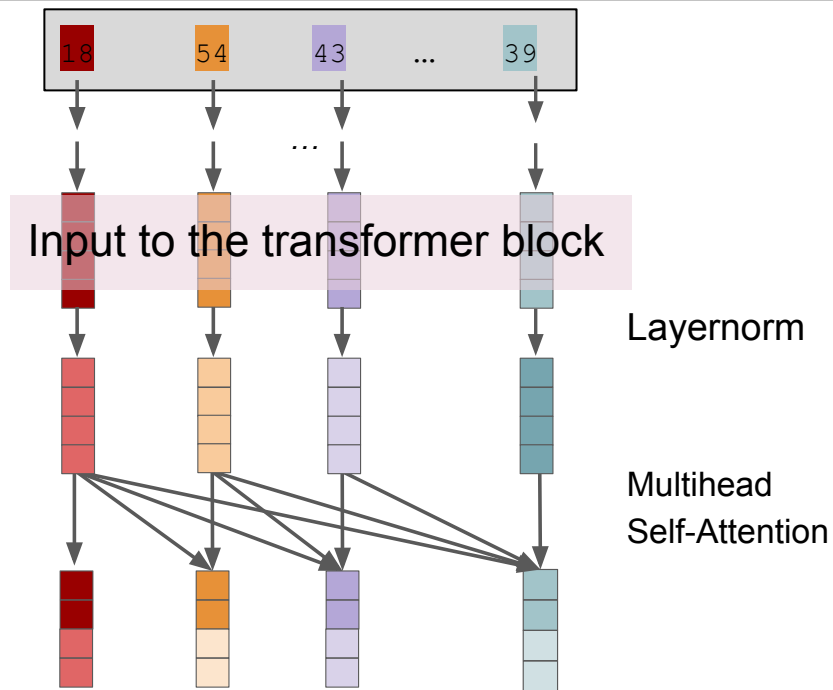
Transformer: Putting it all together



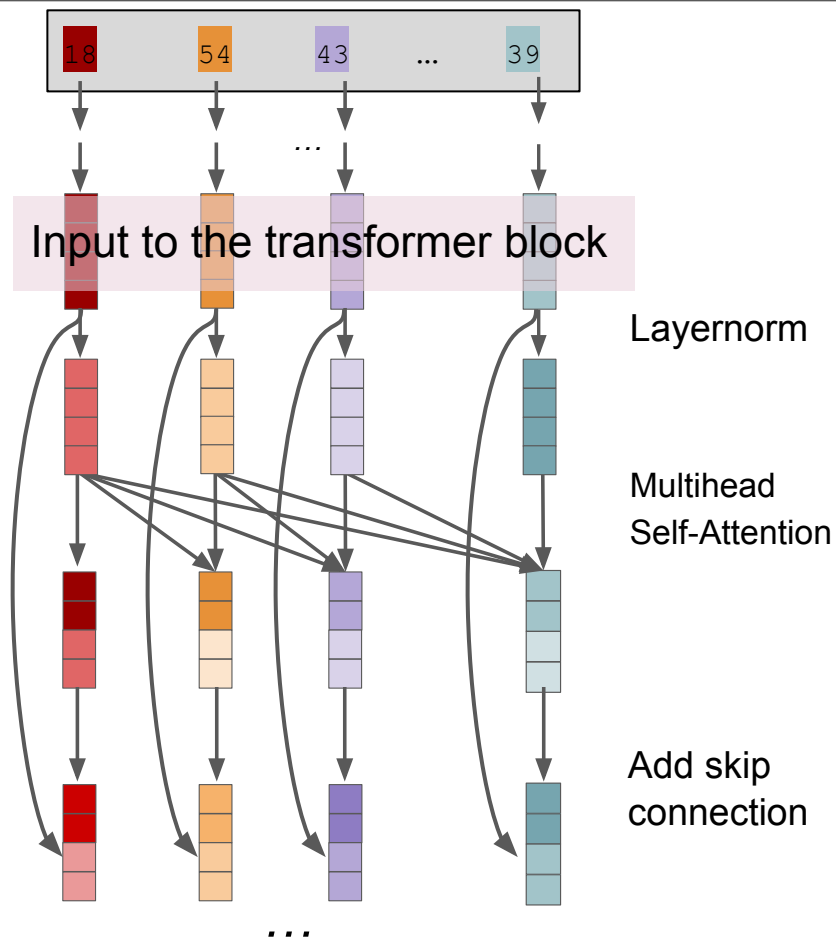
Transformer: Putting it all together



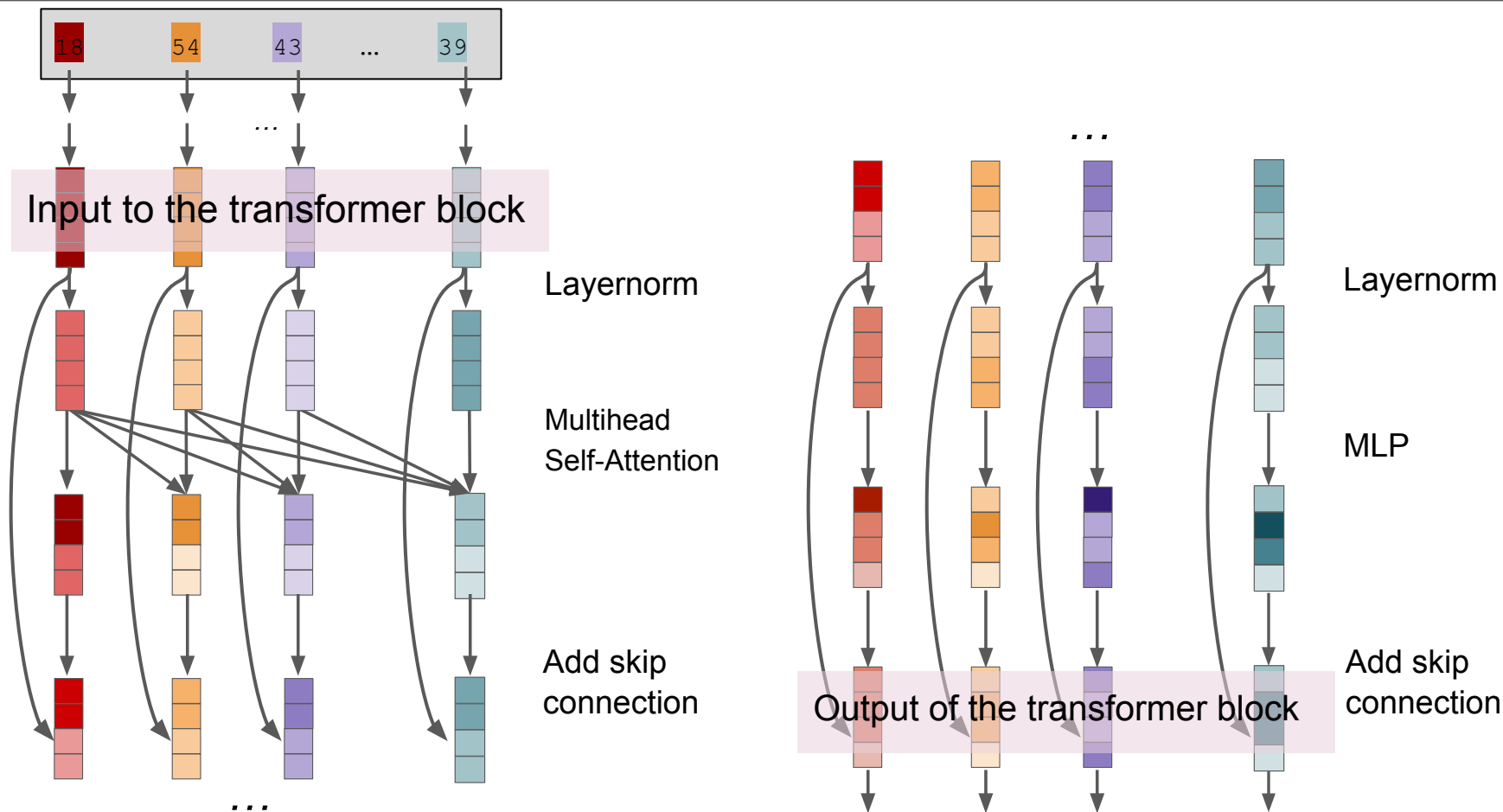
Transformer: Putting it all together



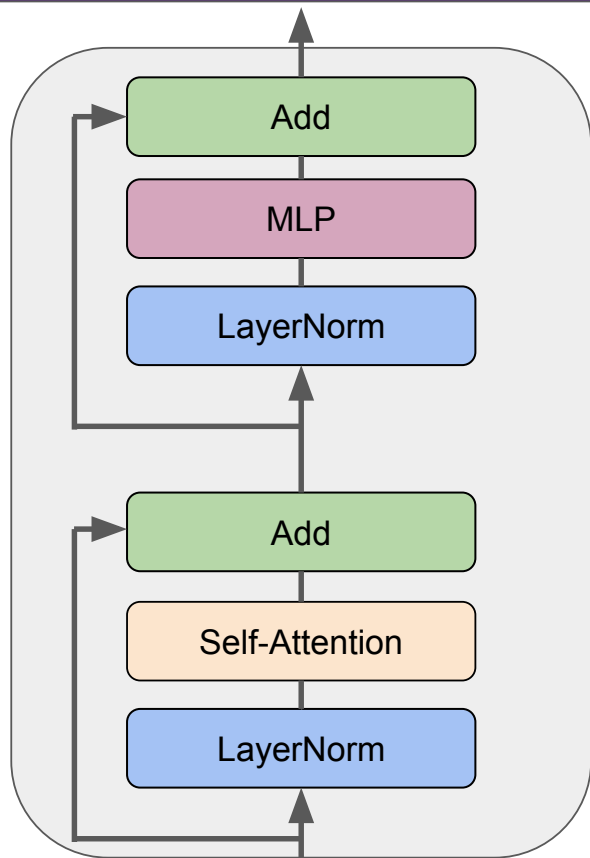
Transformer: Putting it all together



Transformer: Putting it all together



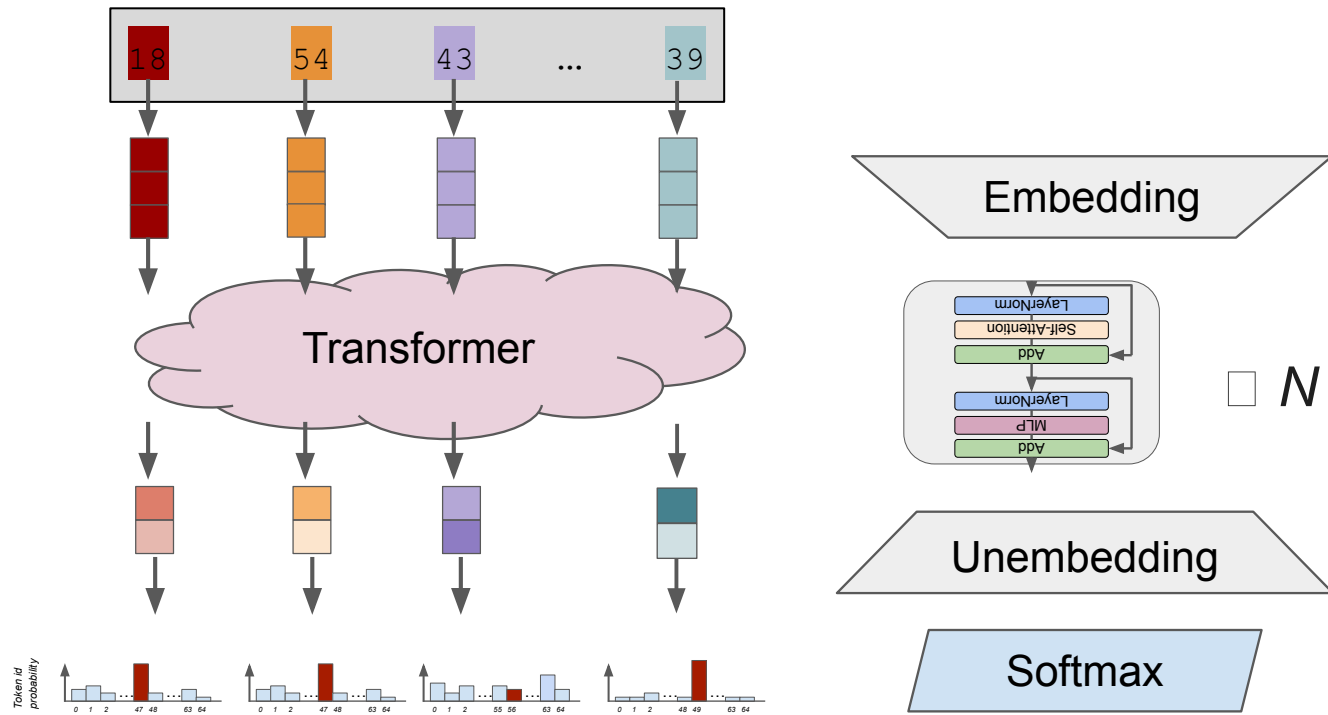
Transformer: Putting it all together



Transformer block

This is our new layer, can repeat!

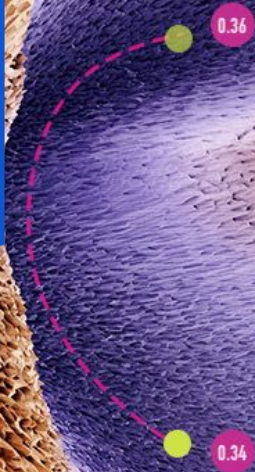
Transformer: Putting it all together



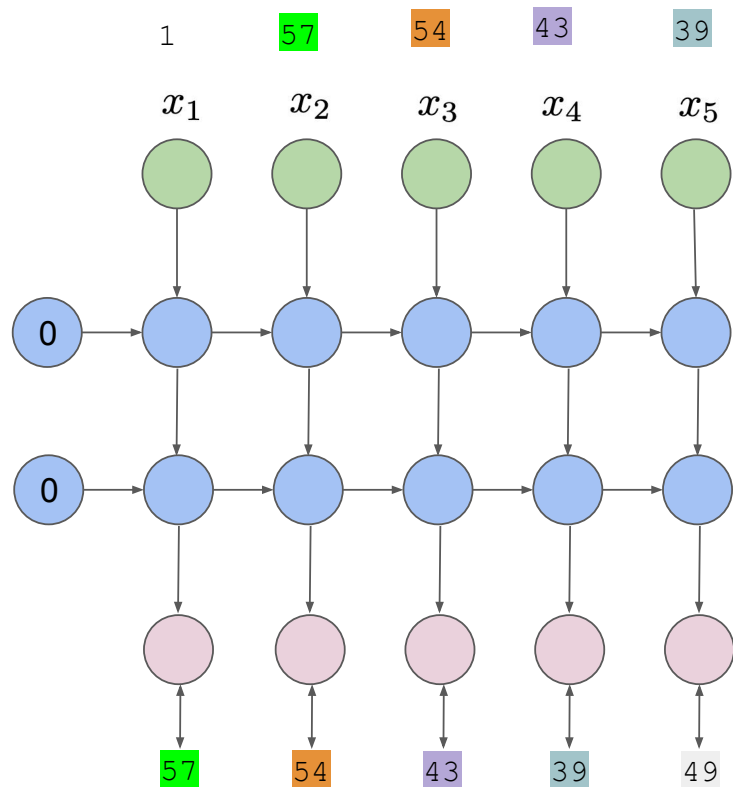
Attention: Temperature

Positional Encoding

Transformers vs RNNs



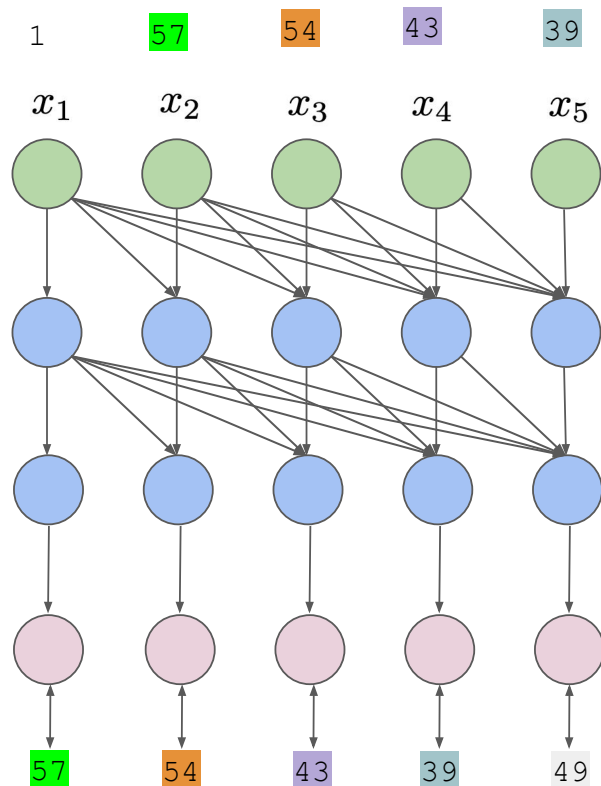
Transformers vs RNNs



RNN

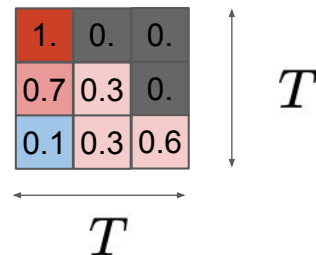
- Process inputs **sequentially**
- Memory
 - Linear in seq len T
- Compute
 - Linear in seq len T
- Time
 - Linear in seq len T
- Context compressed into a fixed-size hidden vector
- Bad at long-range dependencies

Transformers vs RNNs

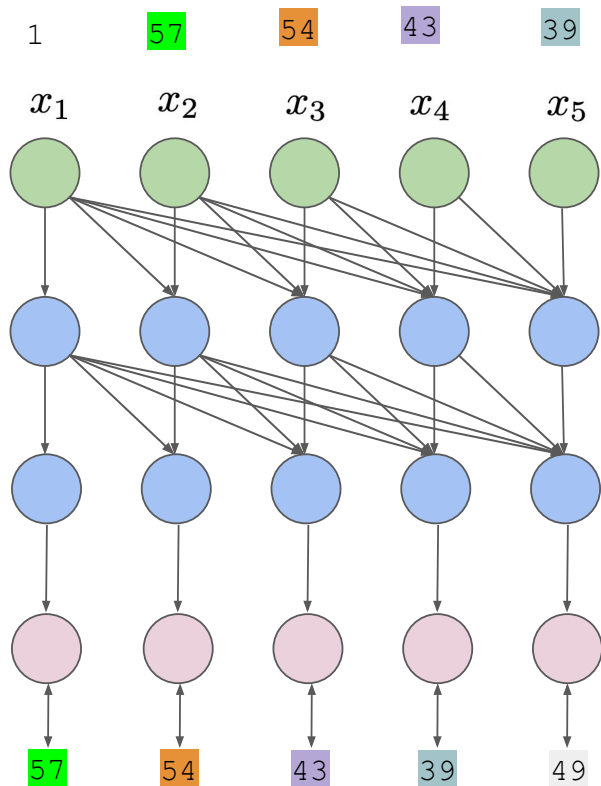


Transformer

- Process inputs in **parallel**
- Memory
 - Quadratic in seq len T
- Compute
 - Quadratic in seq len T
- Time
 - Linear in seq len T
- Good at long-range dependencies



Transformers vs RNNs

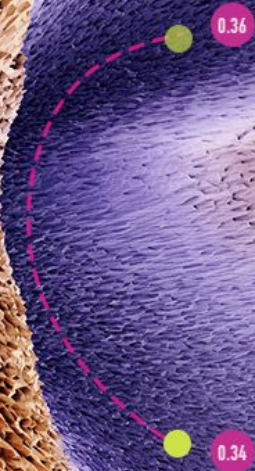


Transformer

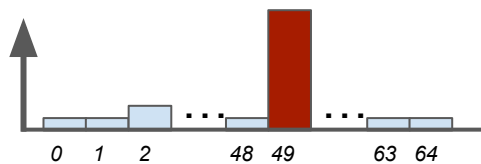
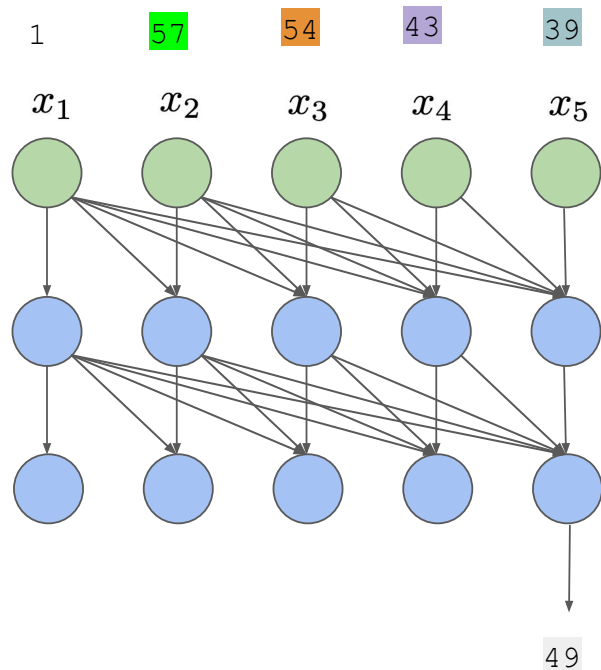
- Process inputs in **parallel**
- Can be scaled very efficiently on large GPU clusters



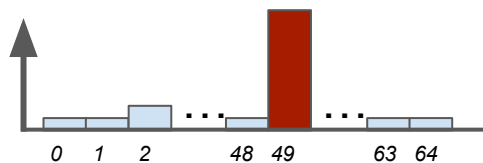
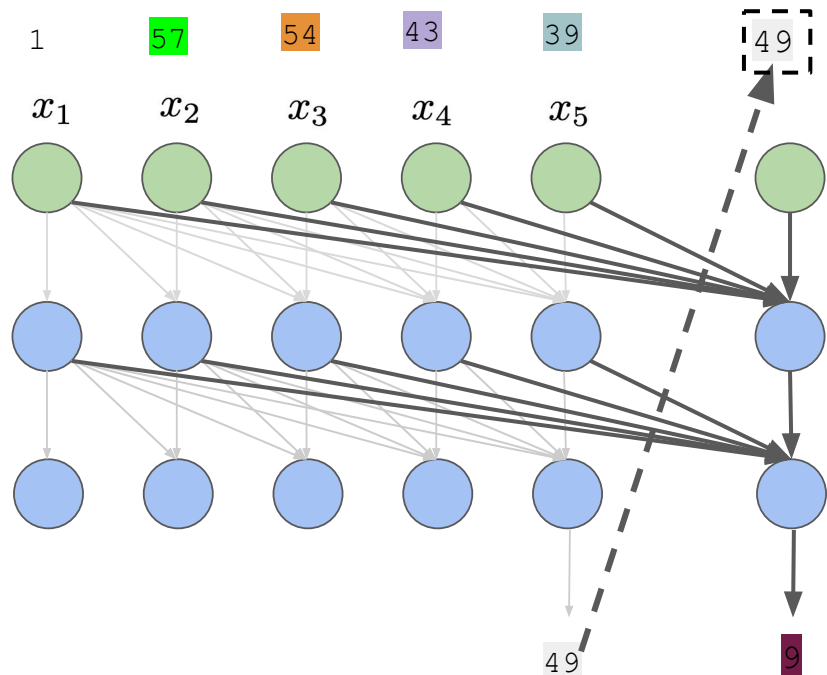
Sampling



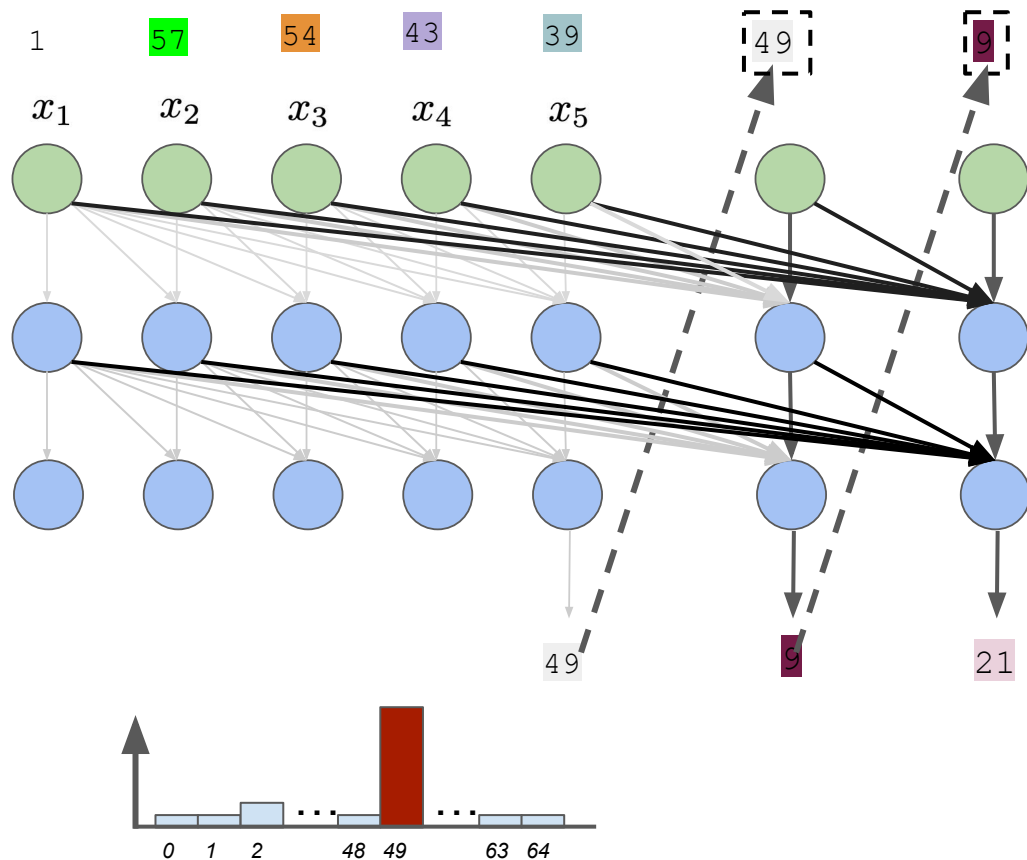
Transformers vs RNNs



Transformer Sampling

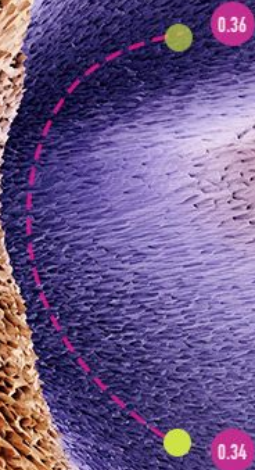


Transformer Sampling

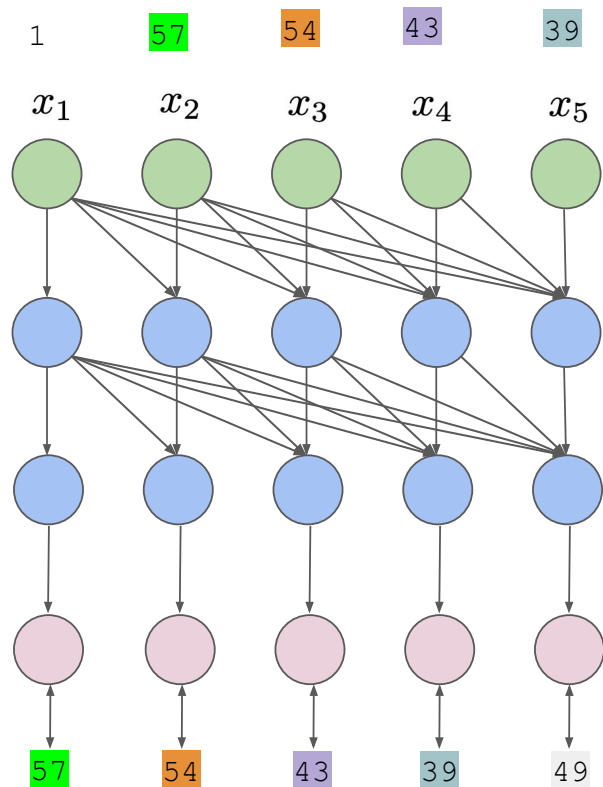


- Same as for RNNs, we can sample many tokens *autoregressively*
- Sampling is sequential

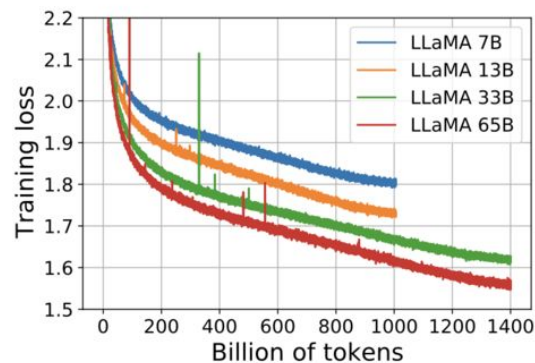
Post-Training



LLM Pretraining



Transformer



Base Language Models

- Next token prediction on the internet does not give us a chatbot
- They continue the text, not respond to your questions
- These are hard to work with

Prompt

Explain the moon landing to a 6 year old in a few sentences.

Completion

GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

Instruction Tuning

Step 1

Collect demonstration data,
and train a supervised policy.

A prompt is sample from
our prompt dataset.



Explain the moon
landing to a 6 year old

A labeler demonstrates
the desired output
behavior.



Some people went
to the moon...

This data is used to
fine-tune GPT-3 with
supervised learning.



SFT



We can just collect examples of
responses we like and finetune
on those

Issue: these examples might be
narrow and hard for the model to
imitate and generalize

<https://openai.com/index/instruction-following/>

Instruction Tuning

Step 1

Collect demonstration data,
and train a supervised policy.

A prompt is sample from
our prompt dataset.

🧠
Explain the moon
landing to a 6 year old

A labeler demonstrates
the desired output
behavior.

👤
Some people went
to the moon...

This data is used to
fine-tune GPT-3 with
supervised learning.

🔵
SFT
🧠
📄📄📄

Step 2

Collect comparison data, and
train a reward model.

A prompt and several
model outputs are
sampled.

🧠
Explain the moon
landing to a 6 year old

A B
Explain gravity... Explain war...
C D
Moon is natural satellite of... People went to the moon...

A labeler ranks the
outputs from best
to worst.

👤
D > C > A = B

This data is used to
train our reward model.

🔵
RM
🧠
D > C > A = B

We will train a model to assign
rewards (scores) to
model-generated responses

Humans give preferences to
model-generated responses

Reward model learns to
predict those preferences

Instruction Tuning

Step 1

Collect demonstration data,
and train a supervised policy.

RLHF: finetune the policy to maximize the
rewards predicted by the reward model
via *reinforcement learning* (RL).

A prompt is sampled from

A labeler demonstrates
the desired output
behavior.

Some people went
to the moon...



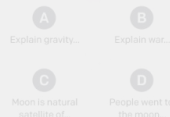
This data is used to
fine-tune GPT-3 with
supervised learning.



Step 2

Collect comparison data, and
train a reward model.

A prompt and several
outputs are sampled from
the SFT model.



These are the
outputs from best
to worst.

D > C > A = B



This data is used to
train our reward model.

D > C > A = B

Step 3

Optimize a policy against the
reward model using
reinforcement learning.

A new prompt is sampled
from the dataset.

Write a story
about frogs

The policy generates an
output.



Once upon a time...

The reward model
calculates a reward for
the output.



The reward is used to
update the policy using
PPO.

r_k

RL for Reasoning

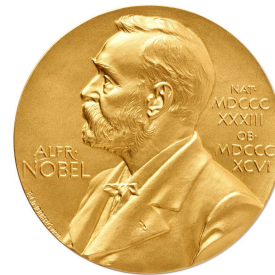
Environment
state \mathbf{s}



Action
 \mathbf{a}



Reward
 \mathbf{r}

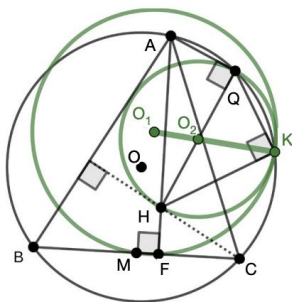


Next
environment \mathbf{s}'



RL for Reasoning

Environment
state \mathbf{s}



$$s \sim \mathcal{D}$$

Math Problem

Action
 \mathbf{a}

...
Construct D: midpoint BH [a]
[a], O₂ midpoint HQ \Rightarrow BQ // O₂D [20]
...
Construct G: midpoint HC [b] ...
 $\angle GMD = \angle GO_2D \Rightarrow M O_2 G D$ cyclic [26]
...
[a], [b] \Rightarrow BC // DG [30]
...
Construct E: midpoint MK [c]
..., [c] $\Rightarrow \angle KFC = \angle KO_1E$ [104]
...

$$a \sim \pi_\theta$$

*Text: Solution
Attempt*

Reward
 \mathbf{r}

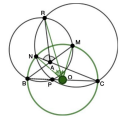
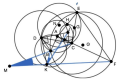
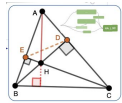
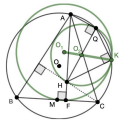


1 if correct else 0

$$\mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\theta} r$$

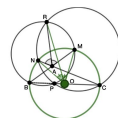
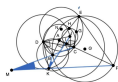
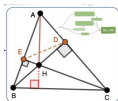
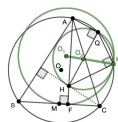
RL for Reasoning

$$s \sim \mathcal{D}$$



RL for Reasoning

$$s \sim \mathcal{D}$$



$$a \sim \pi_\theta$$

Step 1. $AO = A_2O$, $AO = BO$ and $BO = CO \Rightarrow A, A_2, B, C$ are cyclic.

Step 2. A, A_2, B, C are cyclic $\Rightarrow \angle AA_2C = \angle ABC$.

Step 3. A, A_1, A_2 are collinear, A_1, Q, Q_1 are collinear, $\angle AA_2C = \angle ABC$ and $\angle ABC = \angle QQ_1C \Rightarrow \angle A_1A_2C = \angle A_1Q_1C$.

$$\det \begin{bmatrix} -rs & ru & st \\ ru & -rt & st \\ -(u+s) & u+t & 0 \end{bmatrix} = rst[u(u+t) - t(u+s)] + [s(u+t) - u(u+s)] \\ = rst[(u^2 - st) + (st - u^2)] = 0.$$

Let PB_1 and QA_1 meet line AB at X and Y .

Since $\overline{XY} \parallel \overline{PQ}$ it is equivalent to show P_1XYQ_1 is cyclic (Reim's theorem).

Note the angle condition implies P_1CXA and Q_1CYB are cyclic.

By the external version of Pitot theorem, the existence of ω implies that

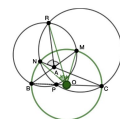
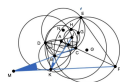
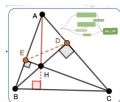
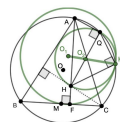
$$BA + AD = CB + CD.$$

Let \overline{PQ} and \overline{ST} be diameters of ω_1 and ω_2 with $P, T \in \overline{AC}$. Then the length relation on $ABCD$ implies that P and T are reflections about the midpoint of \overline{AC} .

Now orient AC horizontally and let K be the "uppermost" point of ω , as shown.

RL for Reasoning

$$s \sim \mathcal{D}$$



$$a \sim \pi_\theta$$

Step 1. $AO = A_2O$, $AO = BO$ and $BO = CO \Rightarrow A, A_2, B, C$ are cyclic.

Step 2. A, A_2, B, C are cyclic $\Rightarrow \angle AA_2C = \angle ABC$.

Step 3. A, A_1, A_2 are collinear, A_1, Q, Q_1 are collinear, $\angle AA_2C = \angle ABC$ and $\angle ABC = \angle QQ_1C \Rightarrow \angle A_1A_2C = \angle A_1Q_1C$.

$$\det \begin{bmatrix} -rs & ru & st \\ ru & -rt & st \\ -(u+s) & u+t & 0 \end{bmatrix} = rst[u(u+t) - t(u+s)] + [s(u+t) - u(u+s)] \\ = rst[(u^2 - st) + (st - u^2)] = 0.$$

Let PB_1 and QA_1 meet line AB at X and Y .
Since $\overline{XY} \parallel \overline{PQ}$ it is equivalent to show P_1XYQ_1 is cyclic (Reim's theorem).
Note the angle condition implies P_1CXA and Q_1CYB are cyclic.

By the external version of Pitot theorem, the existence of ω implies that

$$BA + AD = CB + CD.$$

Let \overline{PQ} and \overline{ST} be diameters of ω_1 and ω_2 with $P, T \in \overline{AC}$. Then the length relation on $ABCD$ implies that P and T are reflections about the midpoint of \overline{AC} .

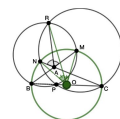
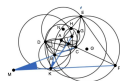
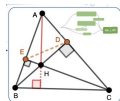
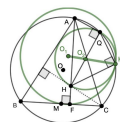
Now orient AC horizontally and let K be the "uppermost" point of ω , as shown.

$$r(s, a)$$



RL for Reasoning

$$s \sim \mathcal{D}$$



$$a \sim \pi_\theta$$

Step 1. $AO = A_2O$, $AO = BO$ and $BO = CO \Rightarrow A, A_2, B, C$ are cyclic.

Step 2. A, A_2, B, C are cyclic $\Rightarrow \angle AA_2C = \angle ABC$.

Step 3. A, A_1, A_2 are collinear, A_1, Q, Q_1 are collinear, $\angle AA_2C = \angle ABC$ and $\angle ABC = \angle QQ_1C \Rightarrow \angle A_1A_2C = \angle A_1Q_1C$.

$$\det \begin{bmatrix} -rs & ru & st \\ ru & -rt & st \\ -(u+s) & u+t & 0 \end{bmatrix} = rst[u(u+t) - t(u+s)] + [s(u+t) - u(u+s)] \\ = rst[(u^2 - st) + (st - u^2)] = 0.$$

Let PB_1 and QA_1 meet line AB at X and Y .
Since $\overline{XY} \parallel \overline{PQ}$ it is equivalent to show P_1XYQ_1 is cyclic (Reim's theorem).
Note the angle condition implies P_1CXA and Q_1CYB are cyclic.

By the external version of Pitot theorem, the existence of ω implies that

$$BA + AD = CB + CD.$$

Let \overline{PQ} and \overline{ST} be diameters of ω_1 and ω_2 with $P, T \in \overline{AC}$. Then the length relation on $ABCD$ implies that P and T are reflections about the midpoint of \overline{AC} .

Now orient AC horizontally and let K be the "uppermost" point of ω , as shown.

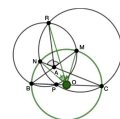
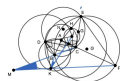
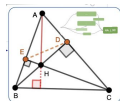
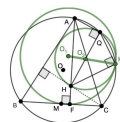
$$r(s, a)$$



Reinforce: SFT policy on successful solutions

RL for Reasoning

$$s \sim \mathcal{D}$$



$$a \sim \pi_\theta$$

Step 1. $AO = A_2O$, $AO = BO$ and $BO = CO \Rightarrow A, A_2, B, C$ are cyclic.

Step 2. A, A_2, B, C are cyclic $\Rightarrow \angle AA_2C = \angle ABC$.

Step 3. A, A_1, A_2 are collinear, A_1, Q, Q_1 are collinear, $\angle AA_2C = \angle ABC$ and $\angle ABC = \angle QQ_1C \Rightarrow \angle A_1A_2C = \angle A_1Q_1C$.

$$\det \begin{bmatrix} -rs & ru & st \\ ru & -rt & st \\ -(u+s) & u+t & 0 \end{bmatrix} = rst[u(u+t) - t(u+s)] + [s(u+t) - u(u+s)] \\ = rst[(u^2 - st) + (st - u^2)] = 0.$$

Let PB_1 and QA_1 meet line AB at X and Y .
Since $\overline{XY} \parallel \overline{PQ}$ it is equivalent to show P_1XYQ_1 is cyclic (Reim's theorem).
Note the angle condition implies P_1CXA and Q_1CYB are cyclic.

By the external version of Pitot theorem, the existence of ω implies that

$$BA + AD = CB + CD.$$

Let \overline{PQ} and \overline{ST} be diameters of ω_1 and ω_2 with $P, T \in \overline{AC}$. Then the length relation on $ABCD$ implies that P and T are reflections about the midpoint of \overline{AC} .

Now orient AC horizontally and let K be the "uppermost" point of ω , as shown.

$$r(s, a)$$



Reinforce: SFT policy on successful solutions

Summary: How to Train a ChatGPT

- Pretrain a large transformer language model on all of internet
- Post-train it to make it useful:
 - Instruction tuning via RLHF
 - Reasoning RL with verifiable rewards

What are you working on?

+ Ask anything

