National Taiwan University of Science and Technology
Computer Science and Information Engineering

Text Mining Tutorial
Course ID: CS4813304

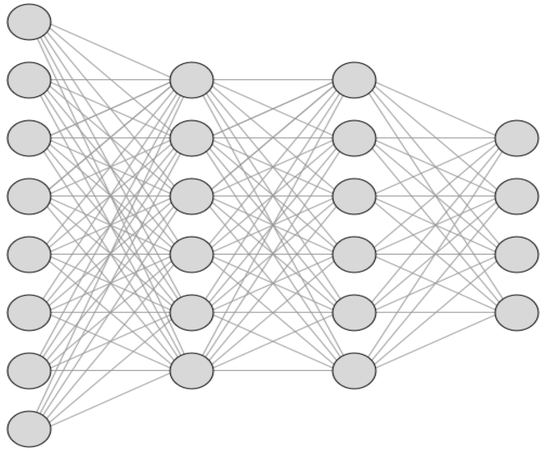
TUTORIAL 10: TRANSFORMER
PART 1: ENCODER PART, ATTENTION

Instructor: Prof. Hsing-Kuo Pao
TA: Zolnamar Dorjsembe (Zola)

Date: April 30, 2024

Contents

- Recap: LSTM
- Attention
- Transformer’s encoder part in detail



LET'S RECAP: RNN & LSTM

Remember the last week’s activity

Solving Transformer by Hand: A Step-by-Step Math Example

Source: https://levelup.gitconnected.com/understanding-transformers-from-start-to-end-a-step-by-step-math-example-16d4e64e6eb1

Step 1. Defining Our Dataset

Dataset (corpus)

I drink and I know things.
When you play the game of thrones, you win or you die.
The true enemy won't wait out the storm, He brings the storm.

Our entire dataset containing only three sentences

Step 2. Finding Vocab Size

$$vocab\ size = count(set(N))$$

vocab_size formula where N is total number of words

Dataset (Corpus)

I drink and I know things.
When you play the game of thrones, you win or you die.
The true enemy won't wait out the storm, He brings the storm.

N = [
I, drink, and, I, Know, things,
When, you, play, the, game, of, thrones, you, win, or, you, die,
The, true, enemy, won't, wait, out, the, storm, He, brings, the, storm
]

calculating variable **N**

$$vocab\ size = count(set(N))$$

set (I, drink, and, I, Know, things, When, you, play, the, game, of, thrones, you, win, or, you, die, The, true, enemy, won't, wait, out, the, storm, He, brings, the, storm)

count (I, drink, and, Know, things, When, you, play, the, game, of, thrones, win, or, die, true, enemy, won't, wait, out, storm, He, brings)

= 23

finding vocab size

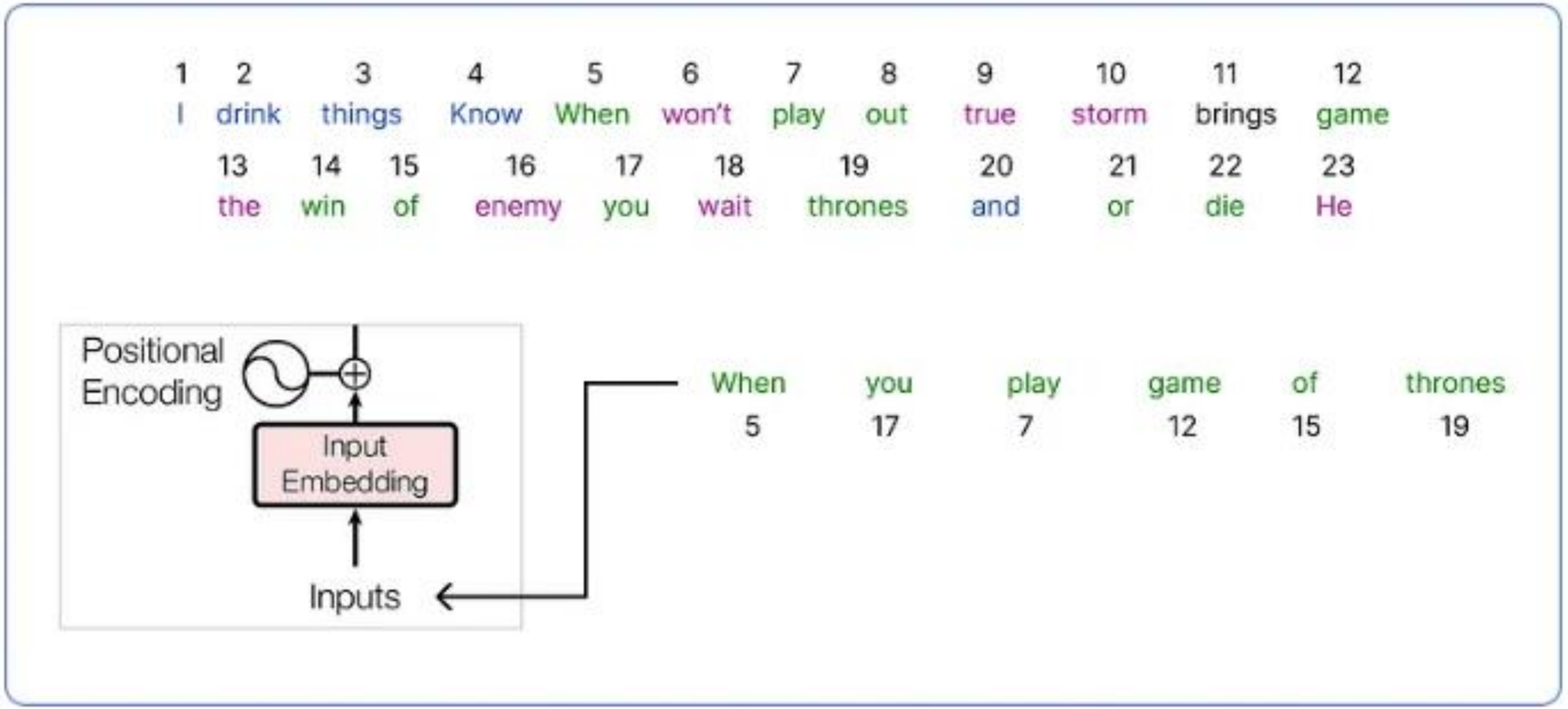
Step 3. Encoding

1	2	3	4	5	6	7	8	9	10	11	12
I	drink	things	Know	When	won't	play	out	true	storm	brings	game

13	14	15	16	17	18	19	20	21	22	23
the	win	of	enemy	you	wait	thrones	and	or	die	He

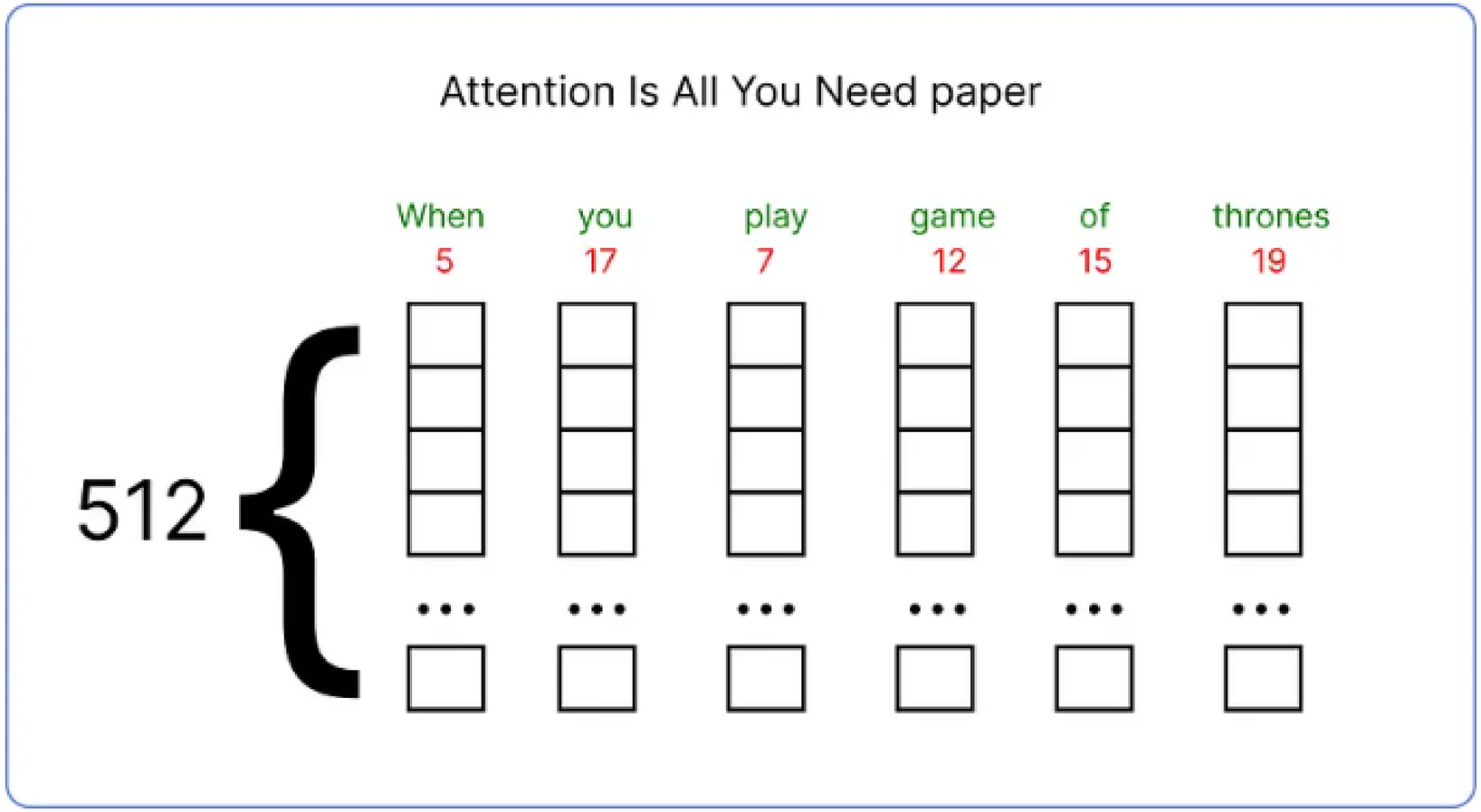
encoding our unique words

Step 4. Calculating Embedding



Input sentence for transformer

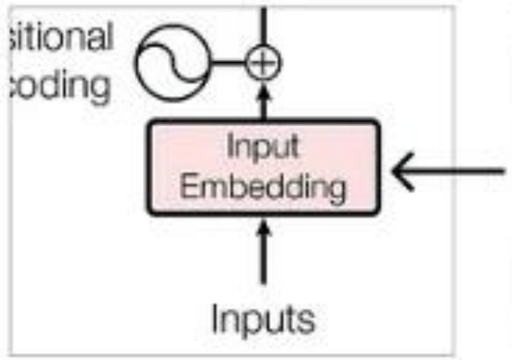
Step 4. Calculating Embedding



Original Paper uses 512 dimension vector

Step 4. Calculating Embedding

(For demonstration, use an embedding vector with a dimension of 6)



When	you	play	game	of	thrones
5	17	7	12	15	19

e1	e2	e3	e4	e5	e6
0.79	0.38	0.01	0.12	0.88	0.6
0.6	0.12	0.51	0.6	0.41	0.33
0.96	0.06	0.27	0.65	0.79	0.75
0.64	0.79	0.31	0.22	0.62	0.48
0.97	0.9	0.56	0.07	0.5	0.94
0.2	0.74	0.59	0.37	0.7	0.21

Embedding vectors of our input

Step 5. Calculating Positional Embedding

Embedding vector for any word

even position
odd position
even position
odd position
even position

...

For even position

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

For odd position

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Positional Embedding formula

Step 5. Calculating Positional Embedding

When
5

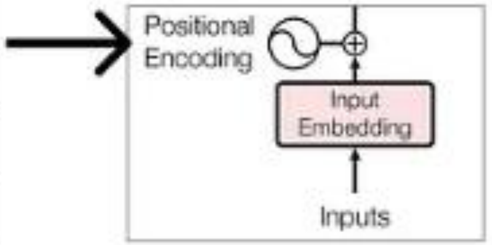
i	e1	Position	Formula	p1
0	0.79	Even	$sin(0/10000^{(2*0/6)})$	0
1	0.6	Odd	$cos(0/10000^{(2*1/6)})$	1
2	0.96	Even	$sin(0/10000^{(2*2/6)})$	0
3	0.64	Odd	$cos(0/10000^{(2*3/6)})$	1
4	0.97	Even	$sin(0/10000^{(2*4/6)})$	0
5	0.2	Odd	$cos(0/10000^{(2*5/6)})$	1
d (dim)	6			
POS	0			

Positional Embedding for word: **When**

Step 5. Calculating Positional Embedding

When	you	play	game	of	thrones
5	17	7	12	15	19

i	p1	p2	p3	p4	p5	p6
0	0	0.8415	0.9093	0.1411	-0.7568	-0.9589
1	1	0.0464	0.9957	0.1388	0.1846	0.9732
2	0	0.0022	0.0043	0.0065	0.0086	0.0108
3	1	0.0001	1	0.0003	0.0004	1
4	0	0	0	0	0	0
5	1	0	1	0	0	1
d (dim)	6	6	6	6	6	6
POS	0	1	2	3	4	5



Calculating Positional Embeddings of our input **(The calculated values are rounded)**

Step 6. Concatenating Positional and Word Embeddings

When	you	play	game	of	thrones
5	17	7	12	15	19

Position Embedding Matrix

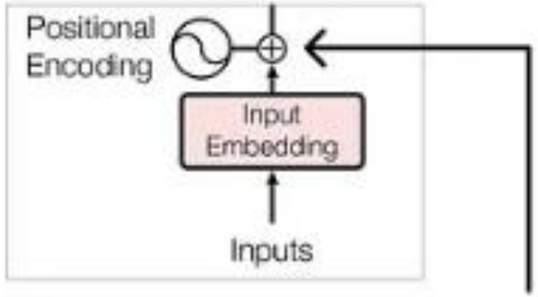
p1	p2	p3	p4	p5	p6
0	0.8415	0.9093	0.1411	-0.7568	-0.9589
1	0.0464	0.9957	0.1388	0.1846	0.9732
0	0.0022	0.0043	0.0065	0.0086	0.0108
1	0.0001	1	0.0003	0.0004	1
0	0	0	0	0	0
1	0	1	0	0	1

+

Word Embedding Matrix

e1	e2	e3	e4	e5	e6
0.79	0.38	0.01	0.12	0.88	0.6
0.6	0.12	0.51	0.6	0.41	0.33
0.96	0.06	0.27	0.65	0.79	0.75
0.64	0.79	0.31	0.22	0.62	0.48
0.97	0.9	0.56	0.07	0.5	0.94
0.2	0.74	0.59	0.37	0.7	0.21

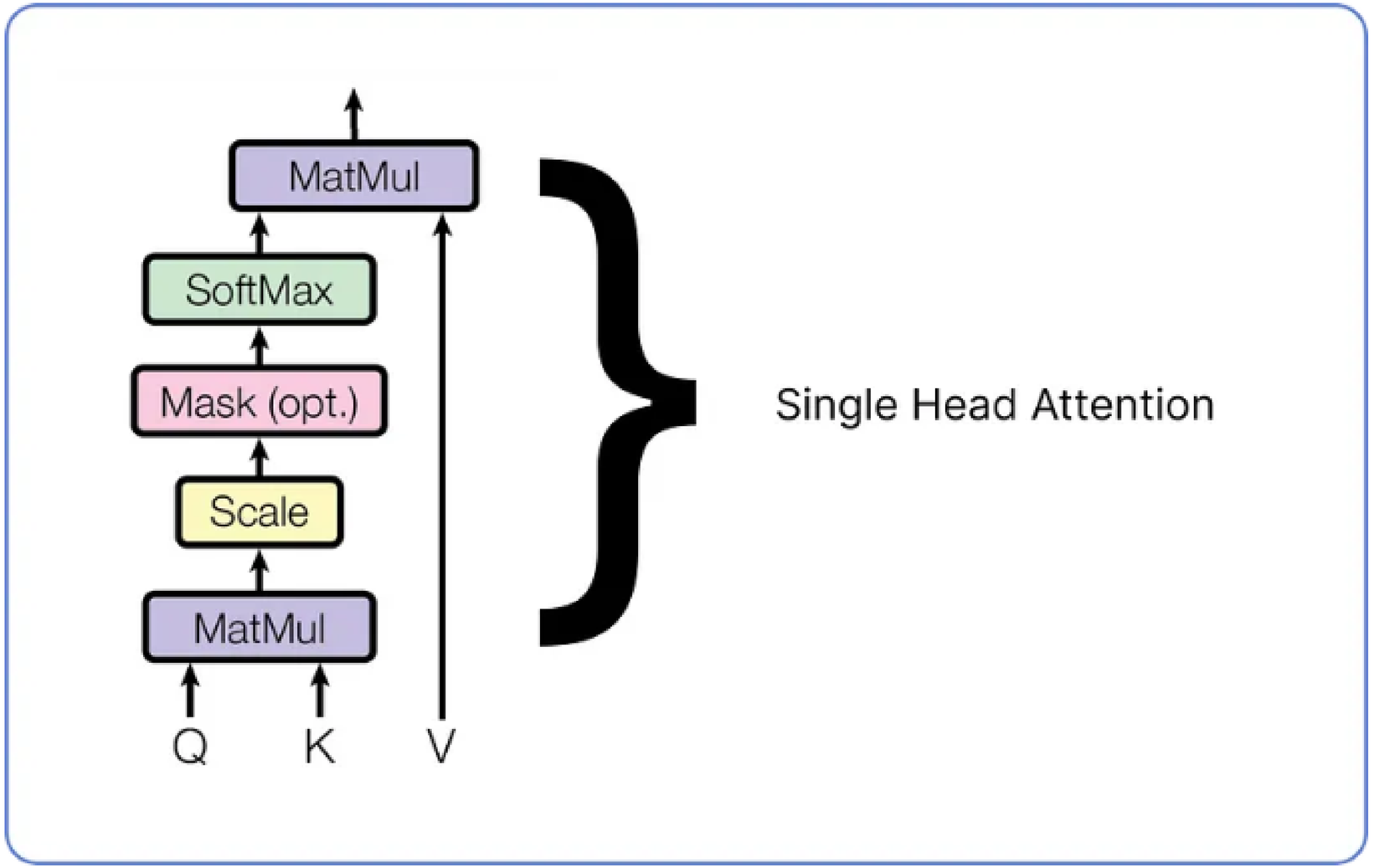
=



ep1	ep2	ep3	ep4	ep5	ep6
0.79	1.22	0.92	0.26	0.12	-0.36
1.6	0.17	1.51	0.74	0.59	1.3
0.96	0.06	0.27	0.66	0.8	0.76
1.64	0.79	1.31	0.22	0.62	1.48
0.97	0.9	0.56	0.07	0.5	0.94
1.2	0.74	1.59	0.37	0.7	1.21

concatenation step

Step 7. Multi Head Attention



Single Head attention in Transformer

Step 7. Multi Head Attention: Query matrix

Word Embedding + Positional Embedding

When	0.79	1.6	0.96	1.64	0.97	1.2
you	1.22	0.17	0.06	0.79	0.9	0.74
play	0.92	1.51	0.27	1.31	0.56	1.59
game	0.26	0.74	0.66	0.22	0.07	0.37
of	0.12	0.59	0.8	0.62	0.5	0.7
thrones	-0.36	1.3	0.76	1.48	0.94	1.21

6 x 6

X

Linear weights for query

0.52	0.45	0.91	0.69
0.05	0.85	0.37	0.83
0.49	0.1	0.56	0.61
0.71	0.64	0.4	0.14
0.76	0.27	0.92	0.67
0.85	0.56	0.57	0.07

6 x 4

calculating Query matrix

Step 7. Multi Head Attention: Key and Value Matrices

Word Embedding + Positional Embedding

When	0.79	1.6	0.96	1.64	0.97	1.2
you	1.22	0.17	0.06	0.79	0.9	0.74
play	0.92	1.51	0.27	1.31	0.56	1.59
game	0.26	0.74	0.66	0.22	0.07	0.37
of	0.12	0.59	0.8	0.62	0.5	0.7
thrones	-0.36	1.3	0.76	1.48	0.94	1.21

6 x 6

X

Linear weights for key

0.74	0.57	0.21	0.73
0.55	0.16	0.9	0.17
0.25	0.74	0.8	0.98
0.8	0.73	0.2	0.31
0.37	0.96	0.42	0.08
0.28	0.41	0.87	0.86

6 x 4

Word Embedding + Positional Embedding

When	0.79	1.6	0.96	1.64	0.97	1.2
you	1.22	0.17	0.06	0.79	0.9	0.74
play	0.92	1.51	0.27	1.31	0.56	1.59
game	0.26	0.74	0.66	0.22	0.07	0.37
of	0.12	0.59	0.8	0.62	0.5	0.7
thrones	-0.36	1.3	0.76	1.48	0.94	1.21

6 x 6

X

Linear weights for value

0.62	0.07	0.7	0.95
0.2	0.97	0.61	0.35
0.57	0.8	0.61	0.5
0.67	0.35	0.98	0.54
0.47	0.83	0.34	0.94
0.6	0.69	0.13	0.98

6 x 4

Calculating Key and Value Matrices

Step 7. Multi Head Attention: Query, Key and Values

Query

3.88	3.8	4.08	3.42
2.55	1.86	2.77	1.78
3.39	3.6	3.49	2.72
1.02	1.18	1.24	1.3
1.9	1.56	1.88	1.53
3.04	2.9	2.73	2.22

6 x 4

key

3.71	4.04	4.15	3.41
2.18	2.51	1.64	1.93
3.28	3.11	3.65	3.01
1.07	1.13	1.64	1.35
1.49	1.97	2.14	1.81
2.51	3.04	3.45	2.28

6 x 4

value

3.88	3.8	4.08	3.42
2.55	1.86	2.77	1.78
3.39	3.6	3.49	2.72
1.02	1.18	1.24	1.3
1.9	1.56	1.88	1.53
3.04	2.9	2.73	2.22

6 x 4

Query, Key, Value matrices

Step 7. Multi Head Attention:
Calculating single-head attention

Query

3.88	3.8	4.08	3.42
2.55	1.86	2.77	1.78
3.39	3.6	3.49	2.72
1.02	1.18	1.24	1.3
1.9	1.56	1.88	1.53
3.04	2.9	2.73	2.22

6 x 4

X

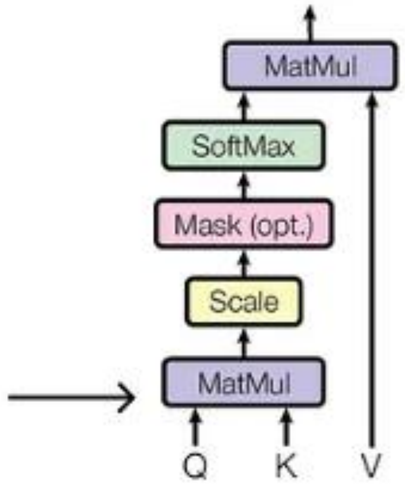
Transpose (Key)

3.71	2.18	3.28	1.07	1.49	2.51
4.04	2.51	3.11	1.13	1.97	3.04
4.15	1.64	3.65	1.64	2.14	3.45
3.41	1.93	3.01	1.35	1.81	2.28

4 x 6

=

58.341	31.2882	49.7306	19.7538	28.1886	43.1644
34.5402	18.2058	29.6169	11.7761	16.6133	25.6698
50.8796	27.3994	43.2409	17.0909	24.5349	37.695
18.1304	9.728	15.4544	6.2134	8.851	13.3894
26.3707	14.0937	22.5509	8.9445	12.6967	19.4858
41.8941	22.668	35.6369	14.004	20.103	30.9265



matrix multiplication between Query and Key

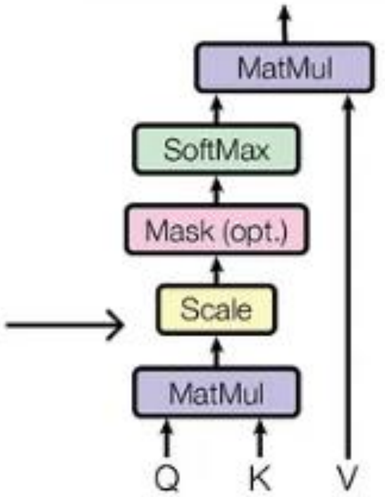
Step 7. Multi Head Attention:
Calculating single-head attention

58.341	31.2882	49.7306	19.7538	28.1886	43.1644
34.5402	18.2058	29.6169	11.7761	16.6133	25.6698
50.8796	27.3994	43.2409	17.0909	24.5349	37.695
18.1304	9.728	15.4544	6.2134	8.851	13.3894
26.3707	14.0937	22.5509	8.9445	12.6967	19.4858
41.8941	22.668	35.6369	14.004	20.103	30.9265

$\sqrt{d_k}$ where d (dimension) is 6

=

23.81721	12.77409	20.30219	8.062904	11.50852	17.62
14.1009	7.434201	12.09231	4.809165	6.781004	10.47973
20.77167	11.186	17.65266	6.976963	10.01433	15.39096
7.401542	3.972256	6.307436	2.535222	3.612997	5.466445
10.76551	5.752218	9.205999	3.64974	5.184753	7.956759
17.10152	9.254989	14.54997	5.715476	8.205791	12.62712



scaling the resultant matrix with dimension **5**

Step 7. Multi Head Attention:
Calculating single-head attention

23.82	12.77	20.3	8.06	11.51	17.62
14.1	7.43	12.09	4.81	6.78	10.48
20.77	11.19	17.65	6.98	10.01	15.39
7.4	3.97	6.31	2.54	3.61	5.47
10.77	5.75	9.21	3.65	5.18	7.96
17.1	9.25	14.55	5.72	8.21	12.63

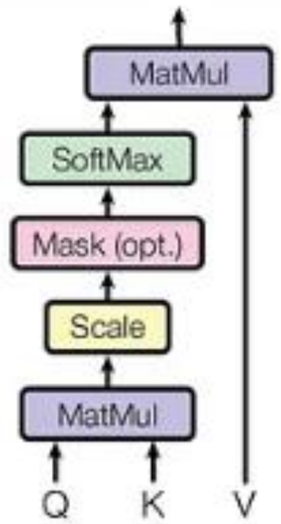
SoftMax

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

$$\text{softmax}(23.82) = \frac{e^{23.82}}{e^{23.82} + e^{12.77} + e^{20.3} + e^{8.06} + e^{11.51} + e^{17.62}}$$

=

0.9693	0	0.0287	0	0	0.002
0.86	0.0011	0.1152	0.0001	0.0006	0.023
0.9534	0.0001	0.0421	0	0	0.0044
0.6476	0.021	0.2177	0.005	0.0146	0.094
0.7803	0.0052	0.164	0.0006	0.0029	0.047
0.9174	0.0004	0.0716	0	0.0001	0.0105



Applying softmax on resultant matrix

Step 7. Multi Head Attention:
Calculating single-head attention

$$\text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right)$$

0.9693	0	0.0287	0	0	0.002
0.86	0.0011	0.1152	0.0001	0.0006	0.023
0.9534	0.0001	0.0421	0	0	0.0044
0.6476	0.021	0.2177	0.005	0.0146	0.094
0.7803	0.0052	0.164	0.0006	0.0029	0.047
0.9174	0.0004	0.0716	0	0.0001	0.0105

6 x 6

X

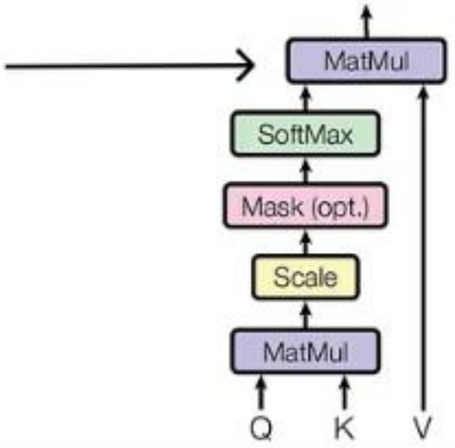
Value

3.88	3.8	4.08	3.42
2.55	1.86	2.77	1.78
3.39	3.6	3.49	2.72
1.02	1.18	1.24	1.3
1.9	1.56	1.88	1.53
3.04	2.9	2.73	2.22

6 x 4

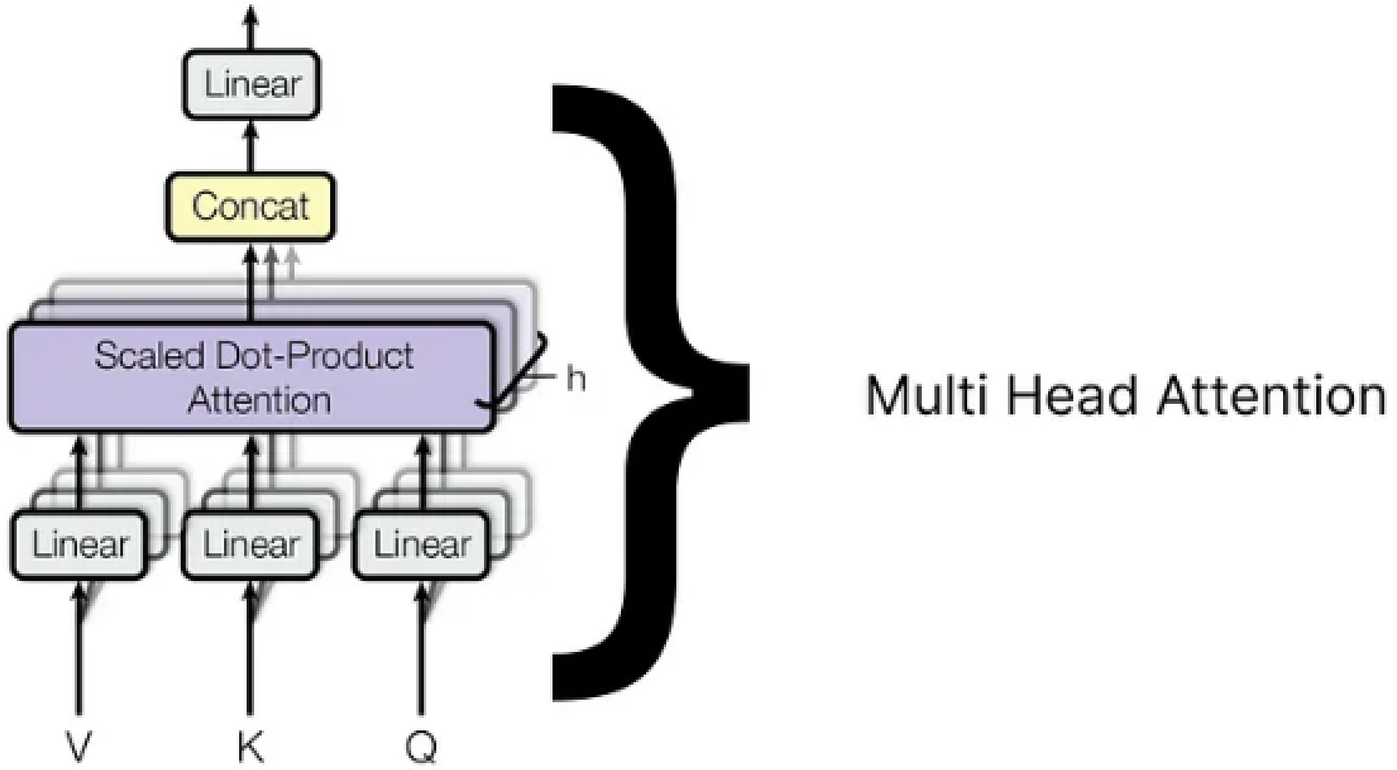
=

3.864257	3.79246	4.060367	3.39751
3.801295	3.75252	3.977937	3.30861
3.855542	3.787426	4.04909	3.385086
3.622841	3.584936	3.750419	3.081834
3.745786	3.706744	3.904894	3.233519
3.835366	3.77523	4.022837	3.356435



calculating the final matrix of single head attention

Step 7. Multi Head Attention:

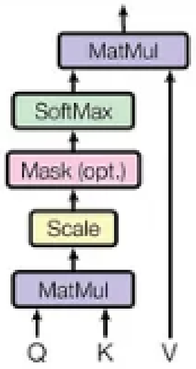


Multi Head attention in Transformer

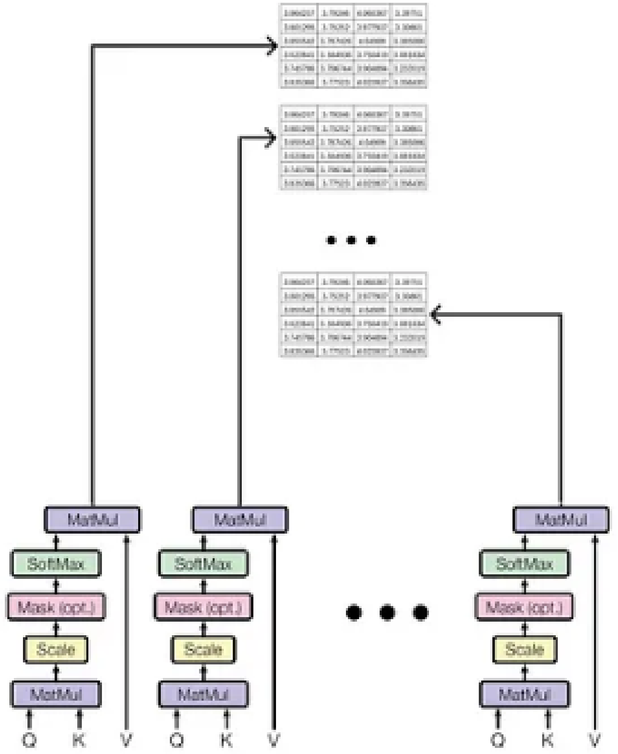
Step 7. Multi Head Attention:
Single-head vs Multi-head attention

Single Head Attention
Our Case

3.864257	3.79246	4.060367	3.39751
3.801295	3.75252	3.977937	3.30861
3.855542	3.787426	4.04909	3.385086
3.622841	3.584936	3.750419	3.081834
3.745786	3.706744	3.904894	3.233519
3.835366	3.77523	4.022837	3.356435



Multi Head Attention (N Heads)
Real world Case Concatenation



Single Head attention vs Multi Head attention

Step 7. Multi Head Attention:
Normalizing single-head attention

$\text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right)$ X Value

3.86	3.79	4.06	3.4
3.8	3.75	3.98	3.31
3.86	3.79	4.05	3.39
3.62	3.58	3.75	3.08
3.75	3.71	3.9	3.23
3.84	3.78	4.02	3.36

6 x 4

X

Linear weights
columns length must be
(embedding+positional) matrix columns length

0.8	0.34	0.45	0.54	0.07	0.53
0.85	0.74	0.78	0.5	0.75	0.55
0.53	0.81	0.55	0.59	0.49	0.14
0.7	0.6	0.12	0.42	0.29	0.87

4 x 6

=

10.84	9.45	7.33	7.8	6.09	7.66
10.65	9.28	7.22	7.67	5.99	7.51
10.83	9.43	7.33	7.79	6.08	7.65
10.08	8.77	6.85	7.25	5.67	7.09
10.48	9.12	7.11	7.54	5.89	7.38
10.77	9.38	7.29	7.75	6.05	7.6

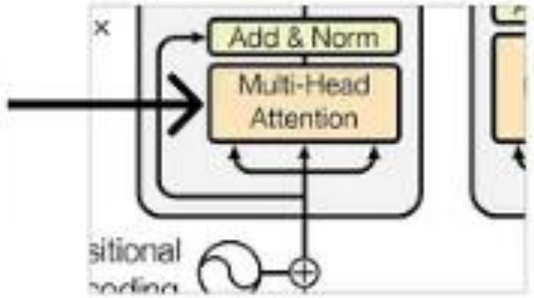
normalizing single head attention matrix

Step 7. Multi Head Attention:

Output of Multi Head attention

10.84	9.45	7.33	7.8	6.09	7.66
10.65	9.28	7.22	7.67	5.99	7.51
10.83	9.43	7.33	7.79	6.08	7.65
10.08	8.77	6.85	7.25	5.67	7.09
10.48	9.12	7.11	7.54	5.89	7.38
10.77	9.38	7.29	7.75	6.05	7.6

6 x 6



Output matrix of multi head attention

Step 8. Adding and Normalizing:

Word Embedding + Positional Embedding

When	0.79	1.6	0.96	1.64	0.97	1.2
you	1.22	0.17	0.06	0.79	0.9	0.74
play	0.92	1.51	0.27	1.31	0.56	1.59
game	0.26	0.74	0.66	0.22	0.07	0.37
of	0.12	0.59	0.8	0.62	0.5	0.7
thrones	-0.36	1.3	0.76	1.48	0.94	1.21

6 x 6

+

Output of Multi Head attention

10.84	9.45	7.33	7.8	6.09	7.66
10.65	9.28	7.22	7.67	5.99	7.51
10.83	9.43	7.33	7.79	6.08	7.65
10.08	8.77	6.85	7.25	5.67	7.09
10.48	9.12	7.11	7.54	5.89	7.38
10.77	9.38	7.29	7.75	6.05	7.6

6 x 6

=

11.63	11.05	8.29	9.44	7.06	8.86
11.87	9.45	7.28	8.46	6.89	8.25
11.75	10.94	7.6	9.1	6.64	9.24
10.34	9.51	7.51	7.47	5.74	7.46
10.6	9.71	7.91	8.16	6.39	8.08
10.41	10.68	8.05	9.23	6.99	8.81

Adding matrices to perform add and norm step

Step 8. Adding and Normalizing:

$$mean = \frac{\sum_{i=1}^{N} X_i}{N}$$

$$standard\ dev. = \sqrt{\frac{\sum_{i=1}^{N} (X_i - \mu)^2}{N}}$$

Row Wise Implementation

						Mean	Standard Deviation
11.63	11.05	8.29	9.44	7.06	8.86	9.26	1.57
11.87	9.45	7.28	8.46	6.89	8.25	8.56	1.64
11.75	10.94	7.6	9.1	6.64	9.24	9.04	1.76
10.34	9.51	7.51	7.47	5.74	7.46	7.86	1.51
10.6	9.71	7.91	8.16	6.39	8.08	8.37	1.35
10.41	10.68	8.05	9.23	6.99	8.81	8.93	1.28

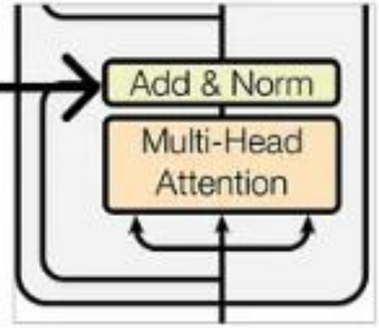
calculating meand and std.

Step 8. Adding and Normalizing:

11.63	11.05	8.29	9.44	7.06	8.86
11.87	9.45	7.28	8.46	6.89	8.25
11.75	10.94	7.6	9.1	6.64	9.24
10.34	9.51	7.51	7.47	5.74	7.46
10.6	9.71	7.91	8.16	6.39	8.08
10.41	10.68	8.05	9.23	6.99	8.81

Mean	Std
9.26	1.57
8.56	1.64
9.04	1.76
7.86	1.51
8.37	1.35
8.93	1.28

$$\frac{\text{value} - \text{mean}}{\text{std} + \text{error}} = \frac{11.63 - 9.26}{1.57 + 0.0001}$$



=

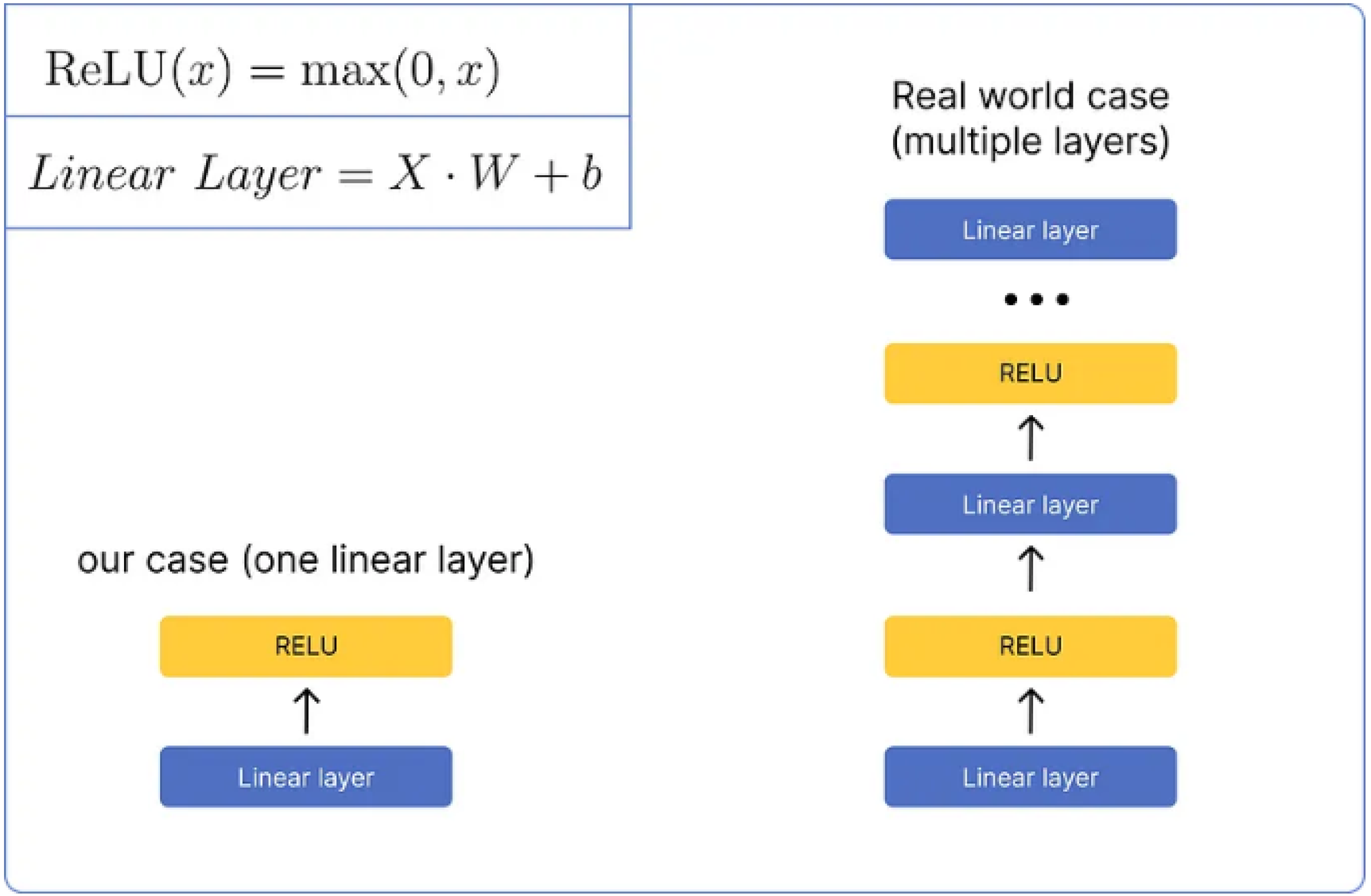
1.51	1.14	-0.62	0.11	-1.4	-0.25
2.02	0.54	-0.78	-0.06	-1.02	-0.19
1.54	1.08	-0.82	0.03	-1.36	0.11
1.64	1.09	-0.23	-0.26	-1.4	-0.26
1.65	0.99	-0.34	-0.16	-1.47	-0.21
1.16	1.37	-0.69	0.23	-1.52	-0.09

normalizing the resultant matrix

Step 9. Feed Forward Network:

$$\mathrm{ReLU}(x) = \max(0, x)$$

$$Linear\ Layer = X \cdot W + b$$



Feed Forward network comparison

Step 9. Feed Forward Network:

Matrix afrer add and norm step

1.51	1.14	-0.62	0.11	-1.4	-0.25
2.02	0.54	-0.78	-0.06	-1.02	-0.19
1.54	1.08	-0.82	0.03	-1.36	0.11
1.64	1.09	-0.23	-0.26	-1.4	-0.26
1.65	0.99	-0.34	-0.16	-1.47	-0.21
1.16	1.37	-0.69	0.23	-1.52	-0.09

6 x 6

X

W

0.5	0.05	0.97	0.22	0.56	0.02
0.17	0.52	0.63	0.48	0.06	0.6
0.53	0.87	0.47	0.1	0.31	0.79
0.83	0.58	0.38	0.09	0.64	0.25
0.81	0.85	0.74	0.35	0.31	0.53
0.25	0.31	0.22	0.77	0.57	0.85

6 x 6

$X \cdot W$

0.49	1.07	0.84	0.14	0.22	0.7
0.24	1.26	1.11	0.12	0.46	0.97
0.53	1.18	-0.82	0.39	0.33	0.59
0.53	0.97	0.98	0.15	0.16	0.52
0.56	1.11	-0.87	0.11	0.2	0.64
0.62	1.02	0.61	0.26	0.14	0.52

6 x 6

+

Bias

b1	b2	b3	b4	b5	b6
0.42	0.18	0.25	0.42	0.35	0.45

=

0.91	1.25	1.09	0.56	0.57	1.15
0.66	1.44	1.36	0.54	0.81	1.42
0.95	1.36	-0.57	0.81	0.68	1.04
0.95	1.15	1.23	0.57	0.51	0.97
0.98	1.29	-0.62	0.53	0.55	1.09
1.04	1.2	0.86	0.68	0.49	0.97

6 x 6

Calculating Linear Layer

Step 9. Feed Forward Network:

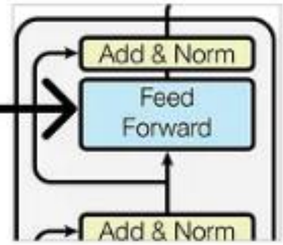
$$\mathrm{ReLU}(x) = \max(0, x)$$

0.91	1.25	1.09	0.56	0.57	1.15
0.66	1.44	1.36	0.54	0.81	1.42
0.95	1.36	-0.57	0.81	0.68	1.04
0.95	1.15	1.23	0.57	0.51	0.97
0.98	1.29	-0.62	0.53	0.55	1.09
1.04	1.2	0.86	0.68	0.49	0.97

6 x 6

$\max(0, 0.91)$

0.91	1.25	1.09	0.56	0.57	1.15
0.66	1.44	1.36	0.54	0.81	1.42
0.95	1.36	0	0.81	0.68	1.04
0.95	1.15	1.23	0.57	0.51	0.97
0.98	1.29	0	0.53	0.55	1.09
1.04	1.2	0.86	0.68	0.49	0.97



Calculating ReLU Layer

Step 10. Adding and Normalizing Again:

Matrix from Feed Forward Network

0.91	1.25	1.09	0.56	0.57	1.15
0.66	1.44	1.36	0.54	0.81	1.42
0.95	1.36	0	0.81	0.68	1.04
0.95	1.15	1.23	0.57	0.51	0.97
0.98	1.29	0	0.53	0.55	1.09
1.04	1.2	0.86	0.68	0.49	0.97

+

Matrix from Previous Add and Norm Step

1.51	1.14	-0.62	0.11	-1.4	-0.25
2.02	0.54	-0.78	-0.06	-1.02	-0.19
1.54	1.08	-0.82	0.03	-1.36	0.11
1.64	1.09	-0.23	-0.26	-1.4	-0.26
1.65	0.99	-0.34	-0.16	-1.47	-0.21
1.16	1.37	-0.69	0.23	-1.52	-0.09

=

2.42	2.39	0.47	0.67	-0.83	0.9
2.68	1.98	0.58	0.48	-0.21	1.23
2.49	2.44	-0.82	0.84	-0.68	1.15
2.59	2.24	1	0.31	-0.89	0.71
2.63	2.28	-0.34	0.37	-0.92	0.88
2.2	2.57	0.17	0.91	-1.03	0.88

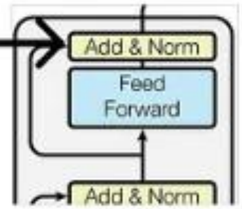
→

Mean	Std
1.0033	1.103534
1.1233	1.214349
0.9033	1.301837
0.9933	1.289055
0.8167	1.306016
0.95	1.320773

=

1.28	1.26	-0.48	-0.3	-1.66	-0.09
1.28	0.71	-0.45	-0.53	-1.1	0.09
1.22	1.18	-1.32	-0.05	-1.22	0.19
1.24	0.97	0.01	-0.53	-1.46	-0.22
1.39	1.12	-0.89	-0.34	-1.33	0.05
0.95	1.23	-0.59	-0.03	-1.5	-0.05

→



Add and Norm after Feed Forward Network

Step 11. Decoder Part: