Transformer

Sergey Ivanov (617)

qbrick@mail.ru

October 28, 2019

1 Recap

2 Attention

3 Multi-Head Attention

4 Attention is All You Need

5 BERT, GPT-2, . . .

Section 1

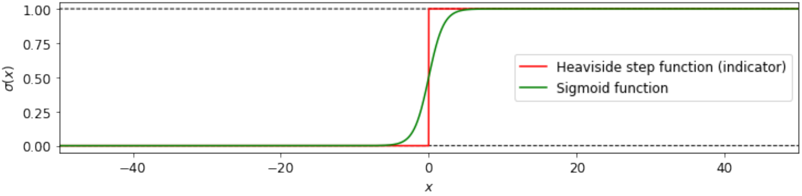
Recap

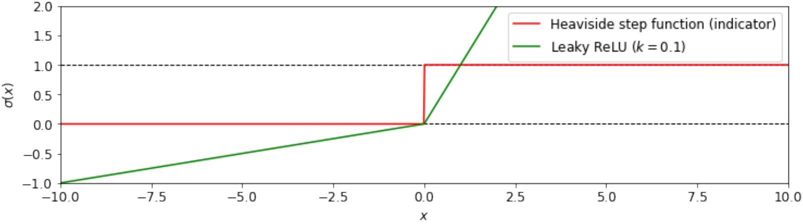
Neural Networks

$$y(x) = \sigma(Wx + b)$$

Neural Networks

$$y(x) = \sigma(Wx + b)$$





Softmax

Classification problem head:

$$softmax(x)_i \propto e^{x_i}$$

Softmax

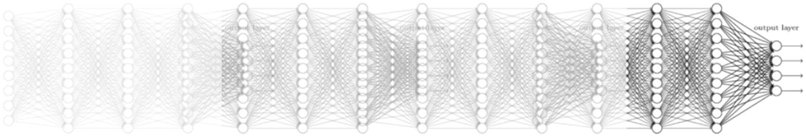
Classification problem head:

$$softmax(x)_i \propto e^{x_i}$$

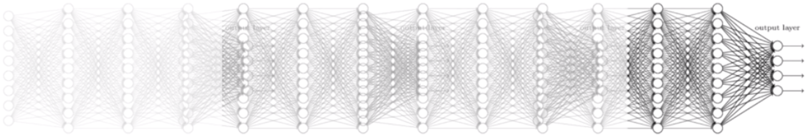
	x_1	x_2	x_3	x_4	x_5
x	**-0.3**	**0**	**0.01**	**1**	**-0.1**
$softmax(x)$	0.116	0.157	0.158	0.426	0.142
$softmax(3x)$	0.017	0.043	0.044	0.863	0.032
$softmax(10x)$	2e-06	5e-5	5e-5	0.99999	2e-05

! scaling factor often influences the smoothness of approximations!

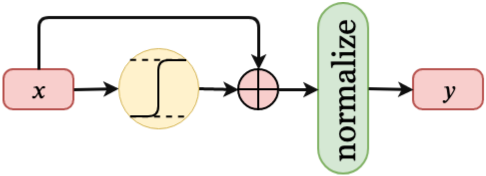
Vanishing gradients problem



Vanishing gradients problem

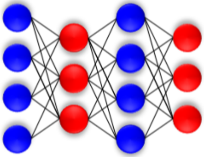


Trying to avoid the problem:

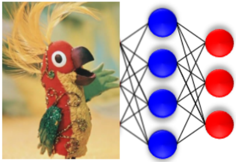


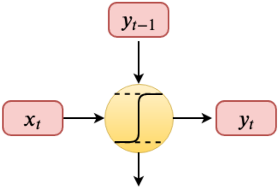
Architectures

FULLY-CONNECTED



CONVOLUTIONAL

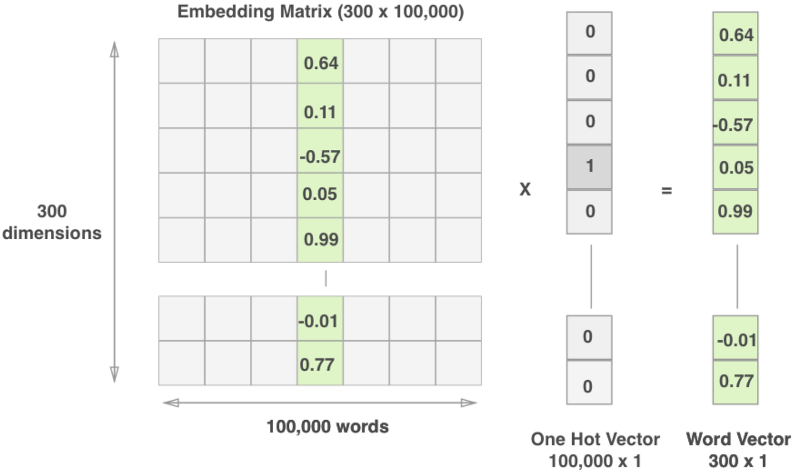




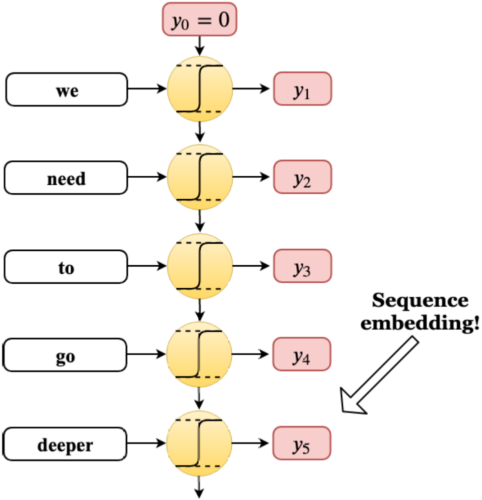
RECURRENT

TRANSFORMER

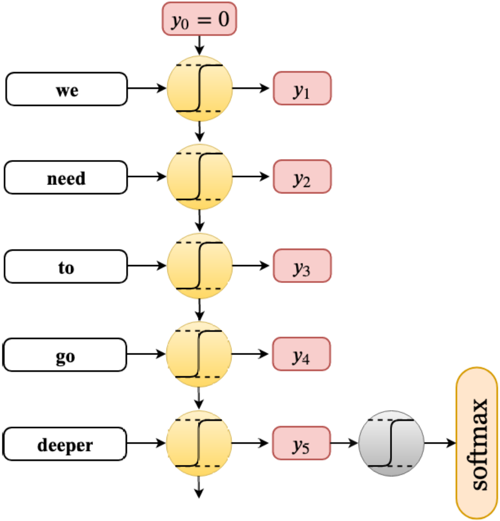
Word embeddings



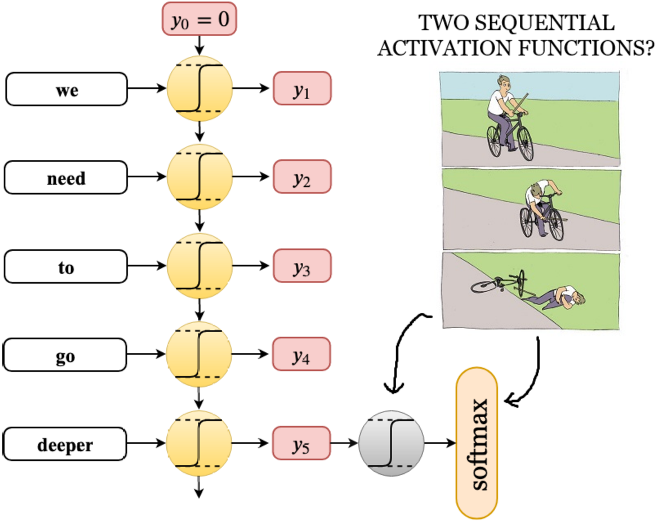
Sequence Embedding



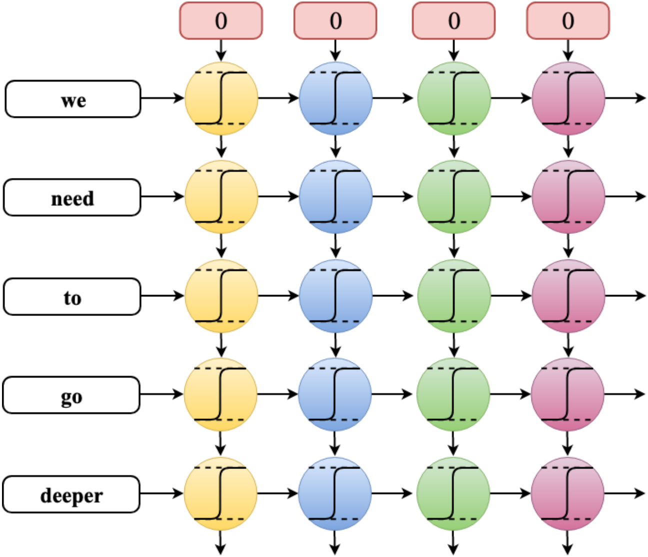
Sequence Embedding



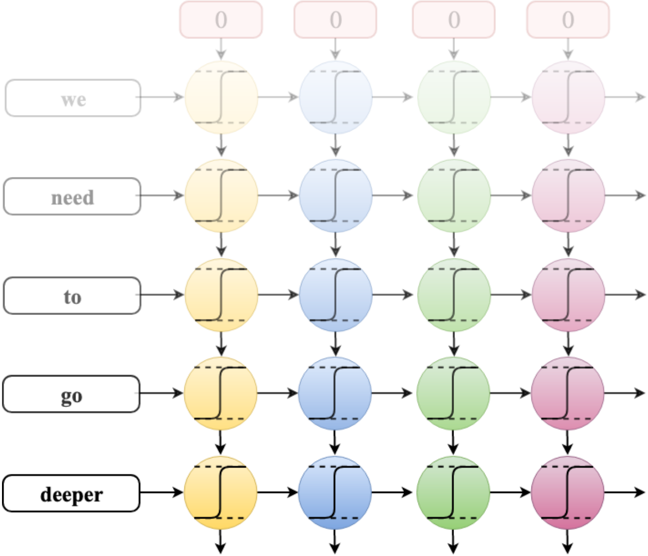
Sequence Embedding



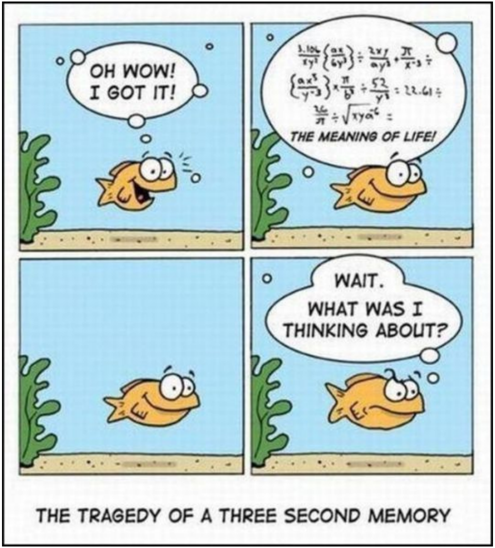
Sequence Embedding



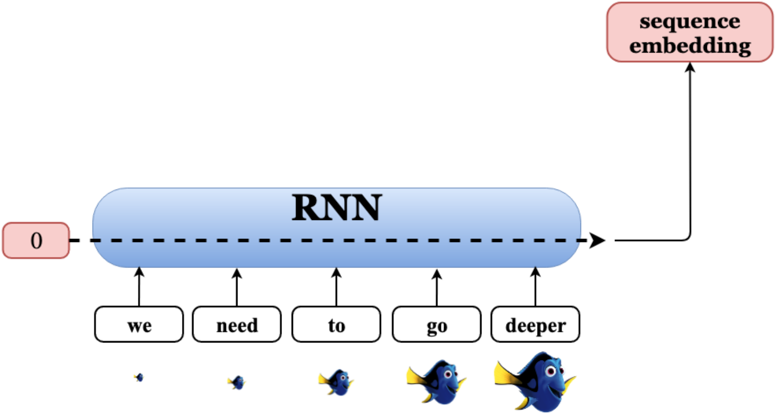
Sequence Embedding



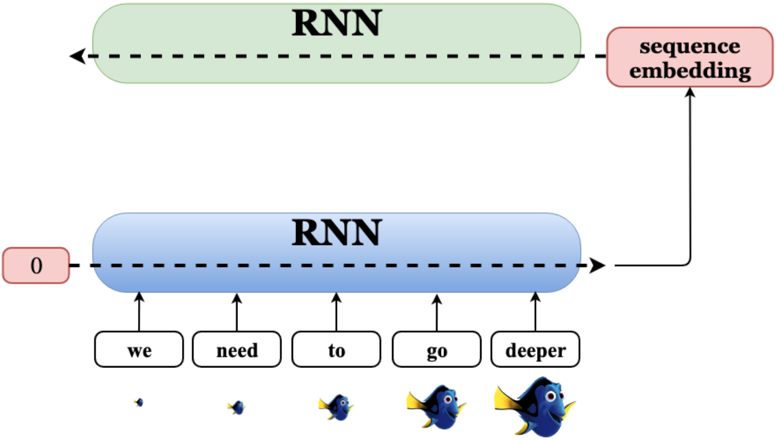
No memory?


OH WOW! I GOT IT!
THE MEANING OF LIFE!
WAIT. WHAT WAS I THINKING ABOUT?
THE TRAGEDY OF A THREE SECOND MEMORY

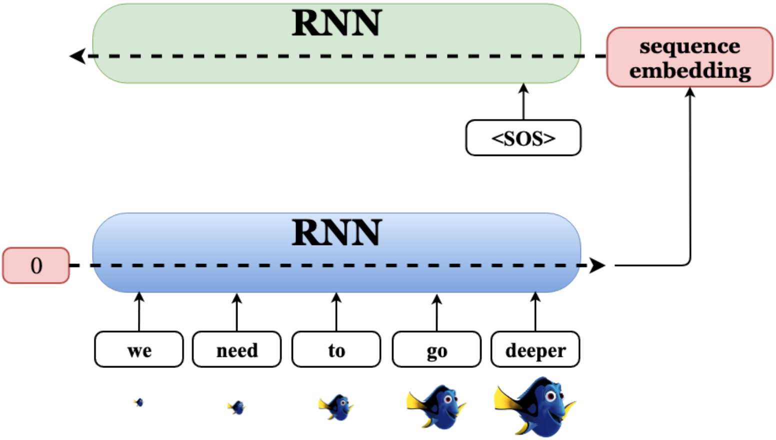
Machine Translation



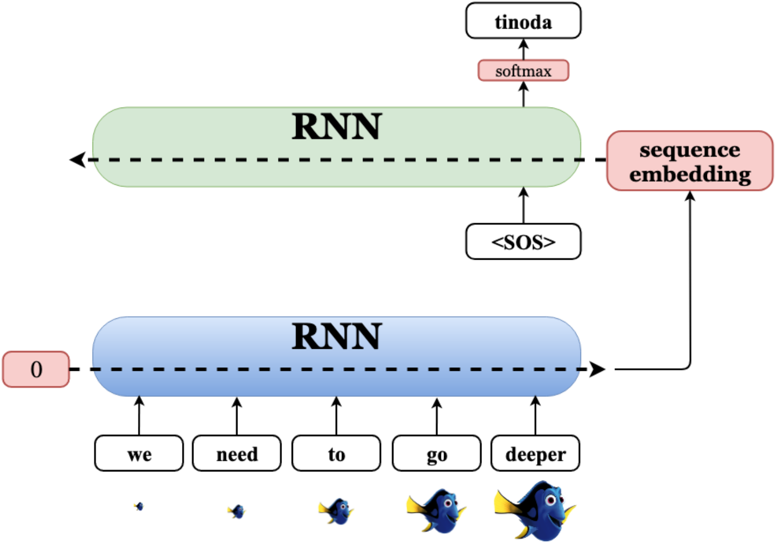
Machine Translation



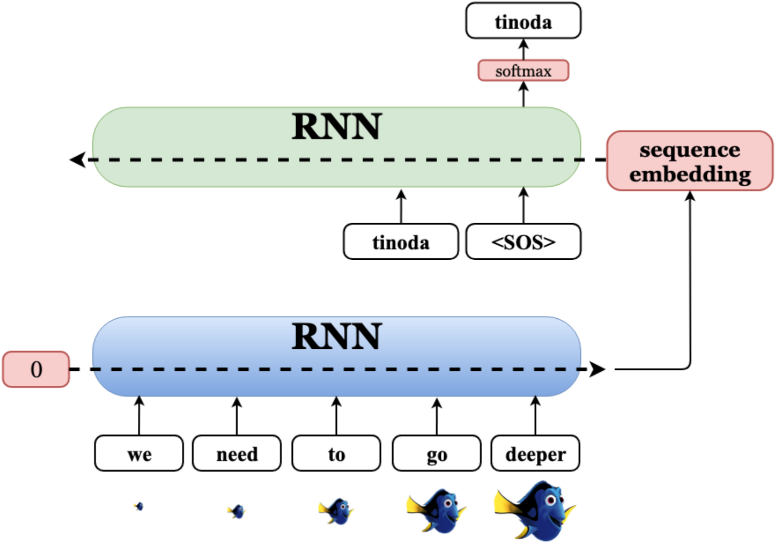
Machine Translation



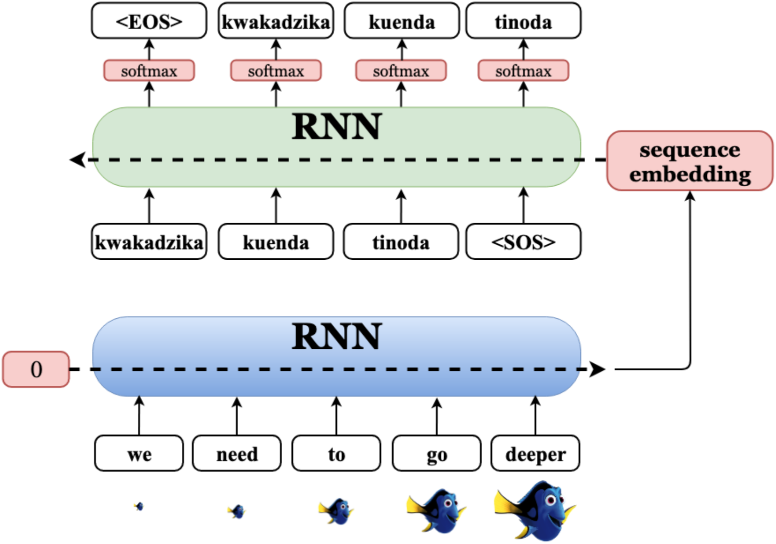
Machine Translation



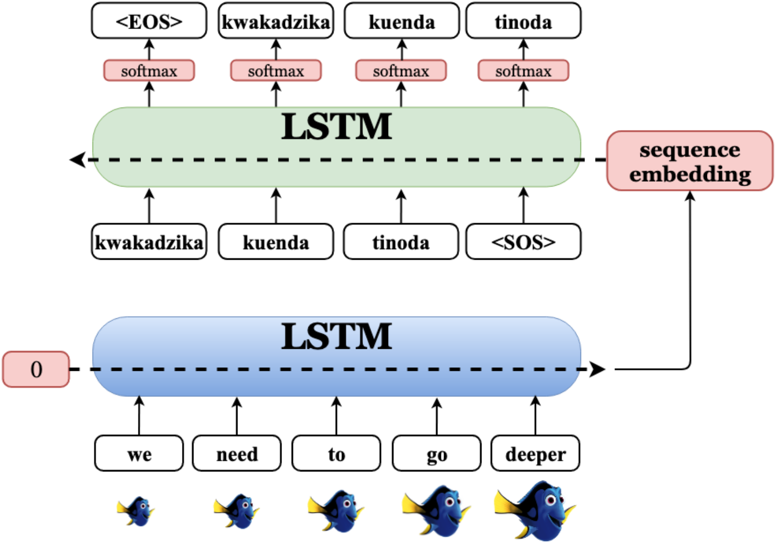
Machine Translation



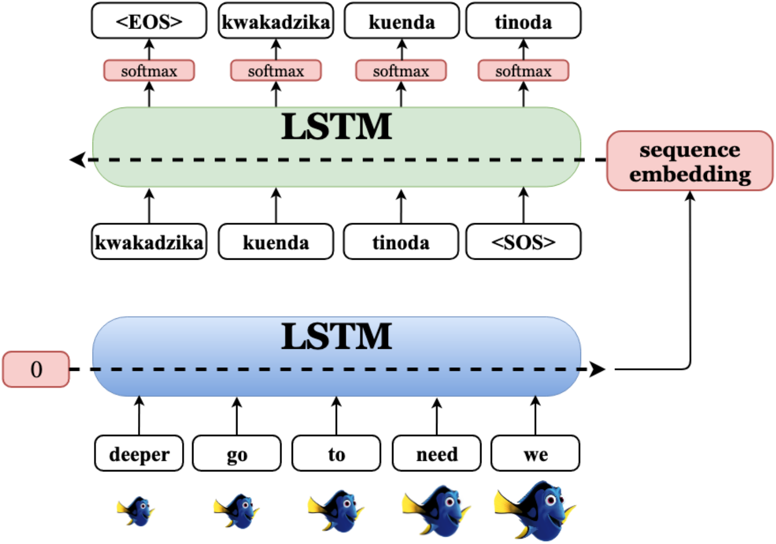
Machine Translation



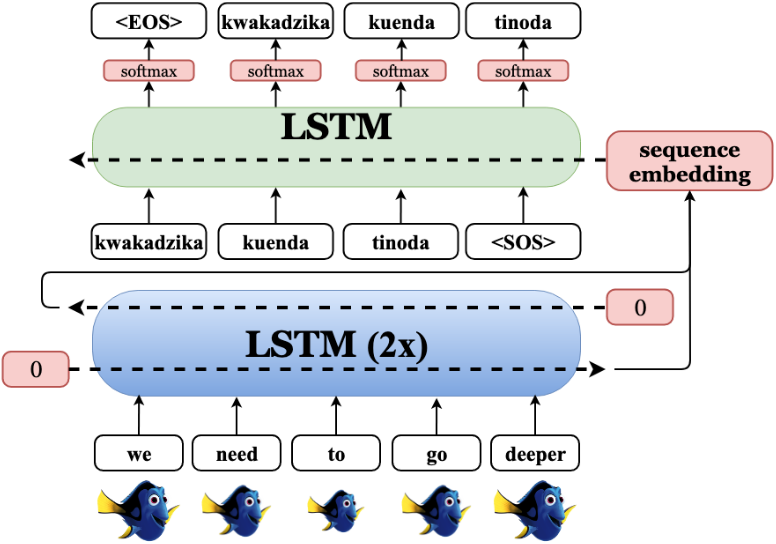
Machine Translation



Machine Translation



Machine Translation



Section 2

Attention

Remembering dict()

Suppose you have a Python dictionary:

```
d = dict(zip(K, V))
```

where

- $K \in \mathbb{R}^{n \times k}$ — n **keys**, $K_i \in \mathbb{R}^k$
- $V \in \mathbb{R}^{n \times v}$ — n **values**, $V_i \in \mathbb{R}^v$

Remembering dict()

Suppose you have a Python dictionary:

```
d = dict(zip(K, V))
```

where

- $K \in \mathbb{R}^{n \times k}$ — n **keys**, $K_i \in \mathbb{R}^k$
- $V \in \mathbb{R}^{n \times v}$ — n **values**, $V_i \in \mathbb{R}^v$

Suppose you have **query** $q \in \mathbb{R}^k$. d[q] — ?

Remembering dict()

Suppose you have a Python dictionary:

```
d = dict(zip(K, V))
```

where

- $K \in \mathbb{R}^{n \times k}$ — n **keys**, $K_i \in \mathbb{R}^k$
- $V \in \mathbb{R}^{n \times v}$ — n **values**, $V_i \in \mathbb{R}^v$

Suppose you have **query** $q \in \mathbb{R}^k$. d[q] — ?

Weeeell, find $i : K_i = q$, then the answer is V_i.

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^{k}$$

Solution

1. $w_i = \mathbb{I}[K_i = q]$
2. $\mathrm{d}[q] = \sum_i^n w_i V_i$

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^{k}$$

Solution

1. $w_i = \mathbb{I}[K_i = q]$ — discrete!
2. $\mathrm{d}[q] = \sum_i^n w_i V_i$

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^{k}$$

Solution

1. $w_i = \mathbb{I}[K_i = q]$ — discrete!
2. $\mathrm{d}[q] = \sum_i^n w_i V_i$

Let $\rho(K_i, q) \in \mathbb{R}$ be some measure of similarity (*compatibility function*) between K_i, q.

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^{k}$$

Solution

1. $w = \text{argmax}(a), \qquad a_i = \rho(K_i, q)$
2. $\text{d}[q] = \sum_i^n w_i V_i$

Let $\rho(K_i, q) \in \mathbb{R}$ be some measure of similarity (*compatibility function*) between K_i, q.

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^k$$

Solution

1. $w = \text{softmax}(a), \qquad a_i = \rho(K_i, q)$
2. $\text{d}[q] = \sum_i^n w_i V_i$

Let $\rho(K_i, q) \in \mathbb{R}$ be some measure of similarity (*compatibility function*) between K_i, q.

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^{k}$$

Solution

1. $w = \text{softmax}(a), \qquad a_i = \rho(K_i, q)$
2. $\text{d}[q] = \sum_i^n w_i V_i$

Let $\rho(K_i, q) \in \mathbb{R}$ be some measure of similarity (*compatibility function*) between K_i, q.

Common choice: $\rho(K_i, q) = \langle K_i, q \rangle$

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^{k}$$

Solution

1. $w = \text{softmax}(a), \qquad a = Kq$
2. $\text{d}[q] = \sum_i^n w_i V_i$

Let $\rho(K_i, q) \in \mathbb{R}$ be some measure of similarity (*compatibility function*) between K_i, q.

Common choice: $\rho(K_i, q) = \langle K_i, q \rangle$

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^k$$

Solution

1. $w = \text{softmax}(\text{K}q)$
2. $\text{d}[q] = \sum_i^n w_i V_i$

Let $\rho(K_i, q) \in \mathbb{R}$ be some measure of similarity (*compatibility function*) between K_i, q.

Common choice: $\rho(K_i, q) = \langle K_i, q \rangle$

Approximating dict()

$$K \in \mathbb{R}^{n \times k} \qquad V \in \mathbb{R}^{n \times v} \qquad q \in \mathbb{R}^{k}$$

Solution

1. $w = \text{softmax}(\text{Kq})$
2. $\text{d}[q] = w^T V$

Let $\rho(K_i, q) \in \mathbb{R}$ be some measure of similarity (*compatibility function*) between K_i, q.

Common choice: $\rho(K_i, q) = \langle K_i, q \rangle$

Scalar Product Normalization

Assume that all $q, K_i \sim \mathcal{N}(0, I_{k \times k})$.

Scalar Product Normalization

Assume that all $q, K_i \sim \mathcal{N}(0, I_{k \times k})$.
What will be distribution of $\langle q, K_i \rangle = \sum_j q_j K_{ij} \sim$?!?

Scalar Product Normalization

Assume that all $q, K_i \sim \mathcal{N}(0, I_{k \times k})$.
What will be distribution of $\langle q, K_i \rangle = \sum_j q_j K_{ij} \sim$?!?

$$\mathbb{E} \sum^{k} q_j K_{ij} = \sum^{k} \mathbb{E} q_j K_{ij} = \{\text{independence}\} = \sum^{k} \mathbb{E} q_j \mathbb{E} K_{ij} = \sum^{k} 0 = 0$$

Scalar Product Normalization

Assume that all $q, K_i \sim \mathcal{N}(0, I_{k\times k})$.
What will be distribution of $\langle q, K_i \rangle = \sum_j q_j K_{ij} \sim ?!?$

$$\mathbb{E} \sum^k q_j K_{ij} = \sum^k \mathbb{E} q_j K_{ij} = \{\text{independence}\} = \sum^k \mathbb{E} q_j \mathbb{E} K_{ij} = \sum^k 0 = 0$$

$$\mathbb{D} \sum^k q_j K_{ij} = \{\text{independence}\} = \sum^k \mathbb{D} q_j K_{ij} =$$

$$= \left\{ {\text{independence} \atop \text{zero expectation}} \right\} = \sum^k \mathbb{D} q_j \mathbb{D} K_{ij} = \sum^k 1 = k$$

Scalar Product Normalization

Assume that all $q, K_i \sim \mathcal{N}(0, I_{k \times k})$.
What will be distribution of $\langle q, K_i \rangle = \sum_j q_j K_{ij} \sim ?!?$

$$\mathbb{E} \sum^k q_j K_{ij} = \sum^k \mathbb{E} q_j K_{ij} = \{\text{independence}\} = \sum^k \mathbb{E} q_j \mathbb{E} K_{ij} = \sum^k 0 = 0$$

$$\mathbb{D} \sum^k q_j K_{ij} = \{\text{independence}\} = \sum^k \mathbb{D} q_j K_{ij} =$$

$$= \left\{ {\text{independence} \atop \text{zero expectation}} \right\} = \sum^k \mathbb{D} q_j \mathbb{D} K_{ij} = \sum^k 1 = k$$

Similarity metric normalisation:

$$\rho(K_i, q) = \frac{\langle K_i, q \rangle}{\sqrt{k}}$$

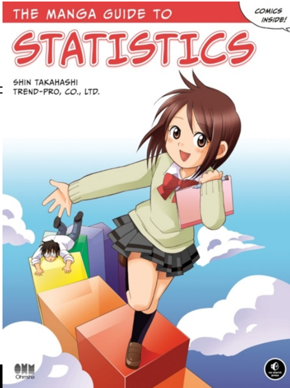
Scalar Product Normalization

Assume that all $q, K_i \sim \mathcal{N}(0, I_{k \times k})$.
What will be distribution of $\langle q, K_i \rangle = \sum_j q_j K_{ij} \sim$?!?

$$\mathbb{E} \sum^{k} q_j K_{ij} = \sum^{k} \mathbb{E} q_j K_{ij} = \{\text{independence}\} = \sum^{k} \mathbb{E} q_j \mathbb{E} K_{ij} = \sum^{k} 0 = 0$$

$$\mathbb{D} \sum^{k} q_j K_{ij} = \{\text{independence}\} = \sum^{k} \mathbb{D} q_j K_{ij} =$$

$$= \left\{ {\text{independence} \atop \text{zero expectation}} \right\} = \sum^{k} \mathbb{D} q_j \mathbb{D} K_{ij}$$



Similarity metric normalisation:

$$\rho(K_i, q) = \frac{\langle K_i, q \rangle}{\sqrt{k}}$$

Attention

input: $K \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{n \times v}, Q \in \mathbb{R}^{b \times k}$

Attention

input: $K \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{n \times v}, Q \in \mathbb{R}^{b \times k}$

Get attention weights matrix:

$$W = \text{softmax}\left(\frac{QK^T}{\sqrt{k}}, \text{dim=1}\right) \in \mathbb{R}^{b \times n}$$

Attention

input: $K \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{n \times v}, Q \in \mathbb{R}^{b \times k}$

Get attention weights matrix:

$$W = \text{softmax}\left(\frac{QK^T}{\sqrt{k}}, \text{dim=1}\right) \in \mathbb{R}^{b \times n}$$

output: $WV \in \mathbb{R}^{b \times v}$

Attention

input: $K \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{n \times v}, Q \in \mathbb{R}^{b \times k}$

Get attention weights matrix:

$$W = \text{softmax}\left(\frac{QK^T}{\sqrt{k}}, \text{dim=1}\right) \in \mathbb{R}^{b \times n}$$

output: $WV \in \mathbb{R}^{b \times v}$
parameters: :(

Attention

input: $K \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{n \times v}, Q \in \mathbb{R}^{b \times k}$

Get attention weights matrix:

$$W = \text{softmax}\left(\frac{QK^T}{\sqrt{k}}, \text{dim=1}\right) \in \mathbb{R}^{b \times n}$$

output: $WV \in \mathbb{R}^{b \times v}$
parameters: :(

Today...

$$K \equiv V$$

Attention

input: $K \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{n \times v}, Q \in \mathbb{R}^{b \times k}$

Get attention weights matrix:

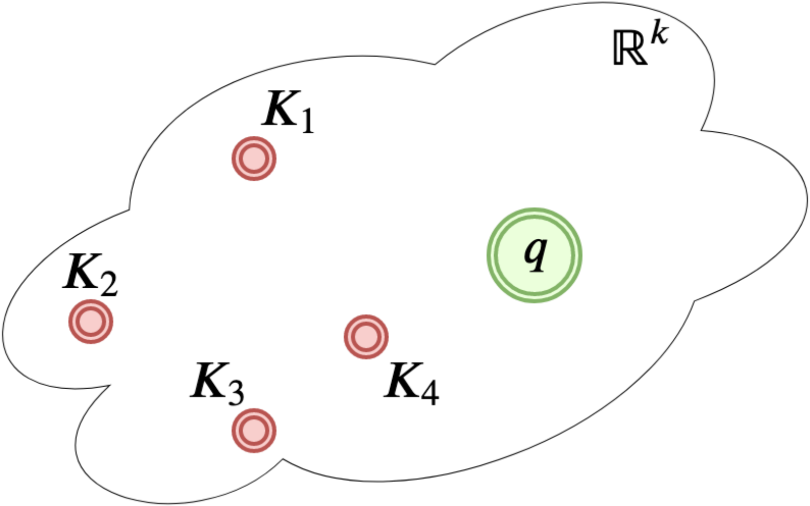
$$W = \text{softmax}\left(\frac{QK^T}{\sqrt{k}}, \text{dim=1}\right) \in \mathbb{R}^{b \times n}$$

output: $WV \in \mathbb{R}^{b \times v}$
parameters: :(

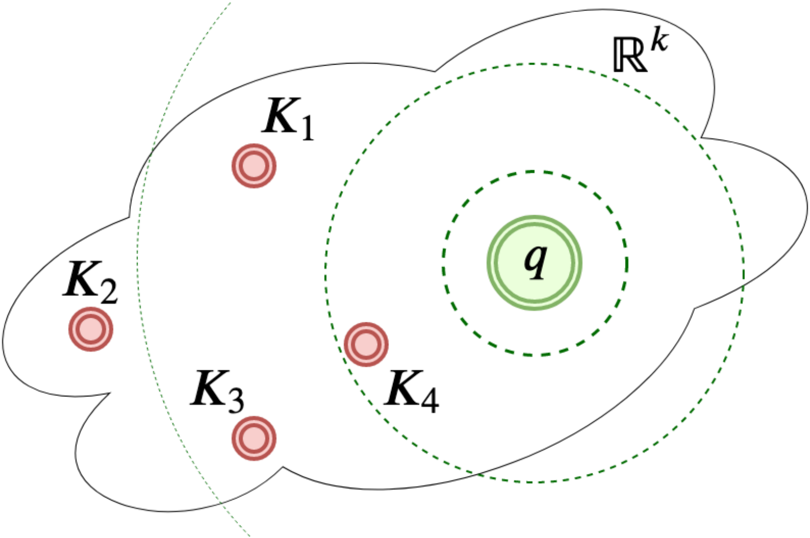
Today...

$$K \equiv V$$

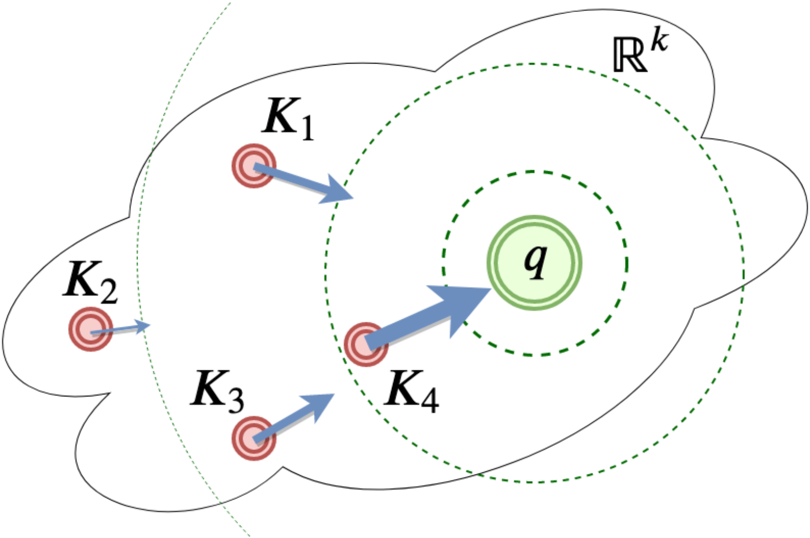
Attention intuition



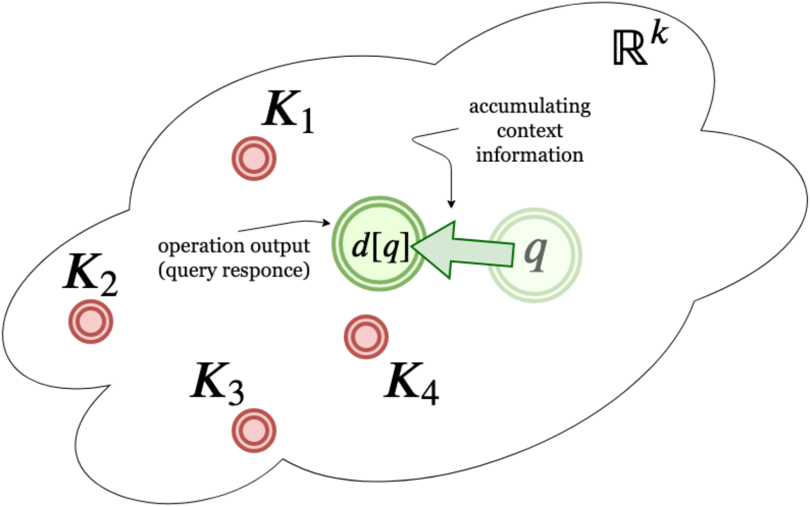
Attention intuition



Attention intuition



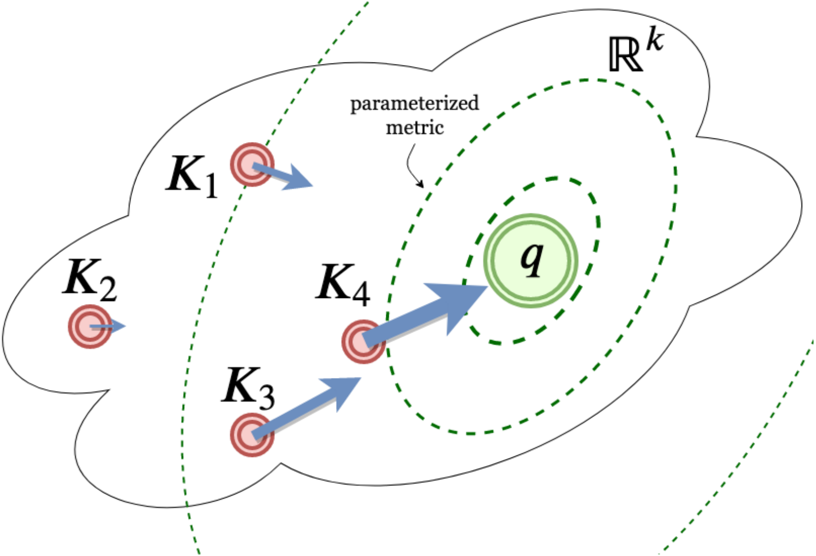
Attention intuition



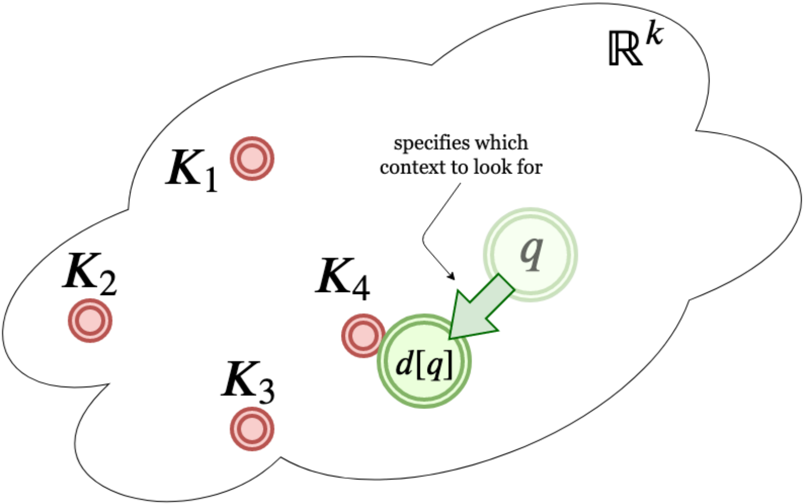
Section 3

Multi-Head Attention

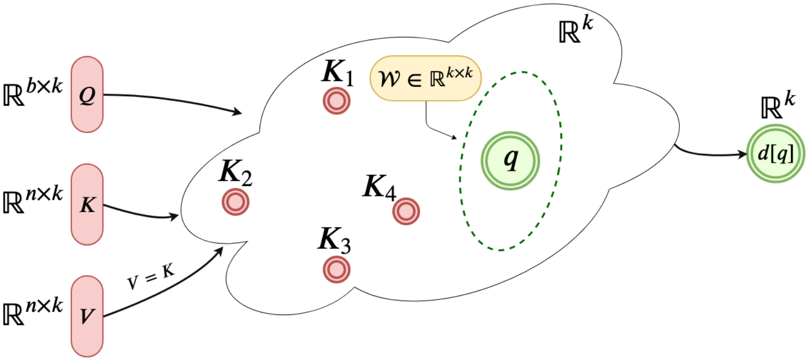
Metric parametrization



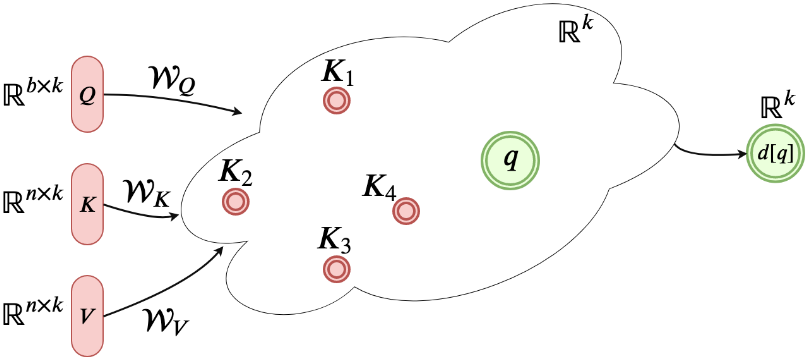
Metric parametrization



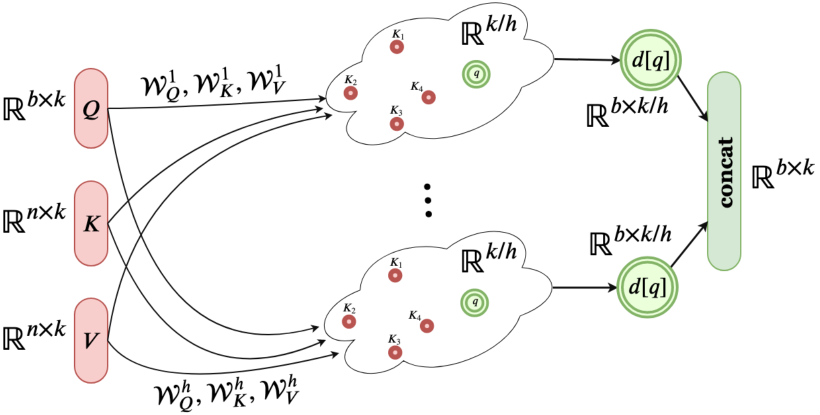
Multi-Head Attention



Multi-Head Attention



Multi-Head Attention



Self-attention

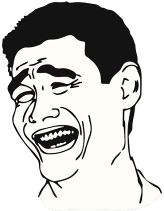
Self-attention

$$Q \equiv K \equiv V$$

Self-attention

Self-attention

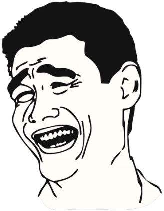
$$Q \equiv K \equiv V$$



Self-attention

Self-attention

$$Q \equiv K \equiv V$$

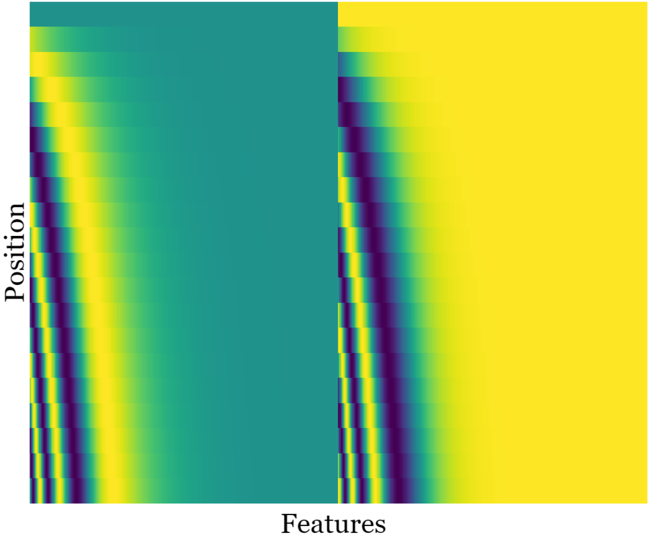


! not all multi-head attention blocks in Transformer are self-attention!

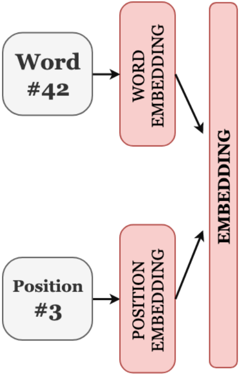
Section 4

Attention is All You Need

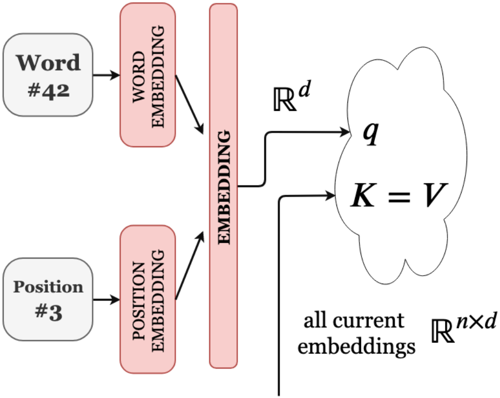
Positional Embeddings



Encoder

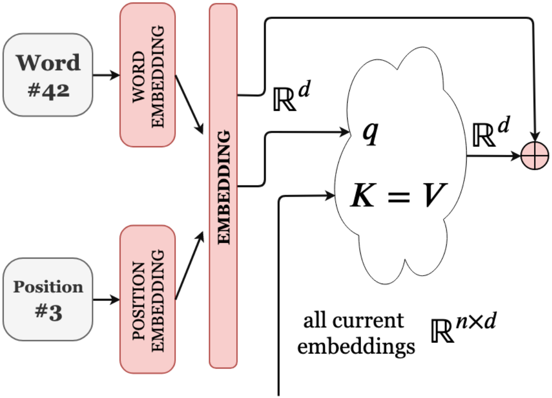

Word
#42
WORD
EMBEDDING
Position
#3
POSITION
EMBEDDING
EMBEDDING

Encoder

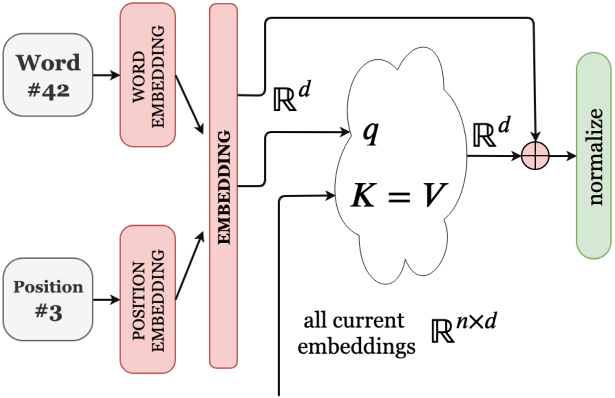


Word #42
WORD EMBEDDING
Position #3
POSITION EMBEDDING
EMBEDDING
$\mathbb{R}^d$
q
$K = V$
all current embeddings
$\mathbb{R}^{n \times d}$

Encoder

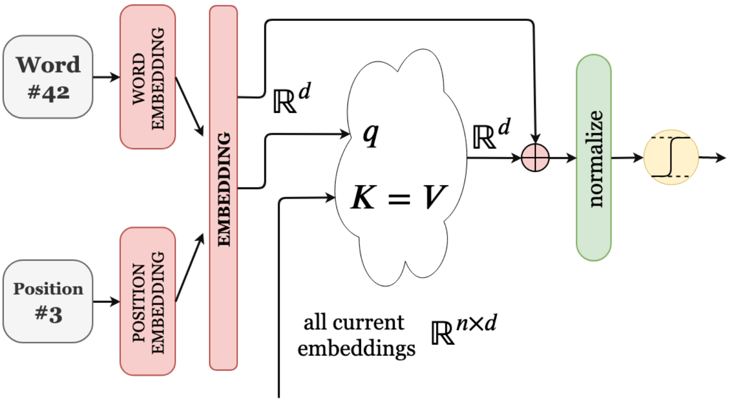

Word
#42
Position
#3
WORD
EMBEDDING
POSITION
EMBEDDING
EMBEDDING
$\mathbb{R}^d$
q
$K = V$
$\mathbb{R}^d$
all current
embeddings
$\mathbb{R}^{n \times d}$

Encoder



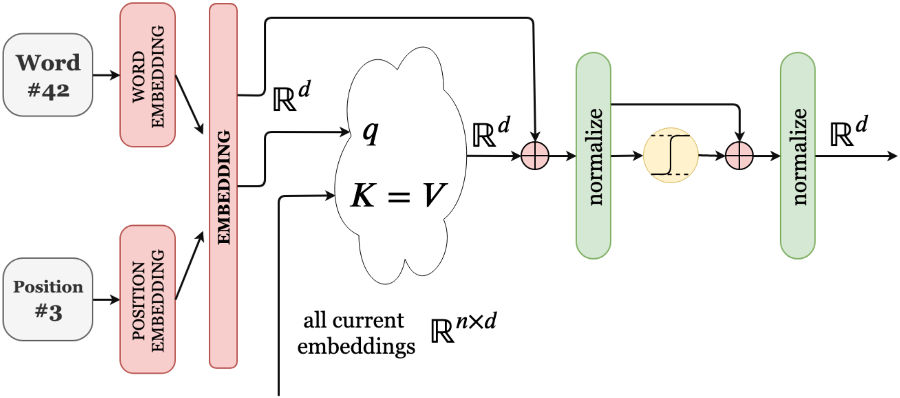
Word #42
WORD EMBEDDING
Position #3
POSITION EMBEDDING
EMBEDDING
$\mathbb{R}^d$
q
$K = V$
$\mathbb{R}^d$
normalize
all current embeddings
$\mathbb{R}^{n \times d}$

Encoder

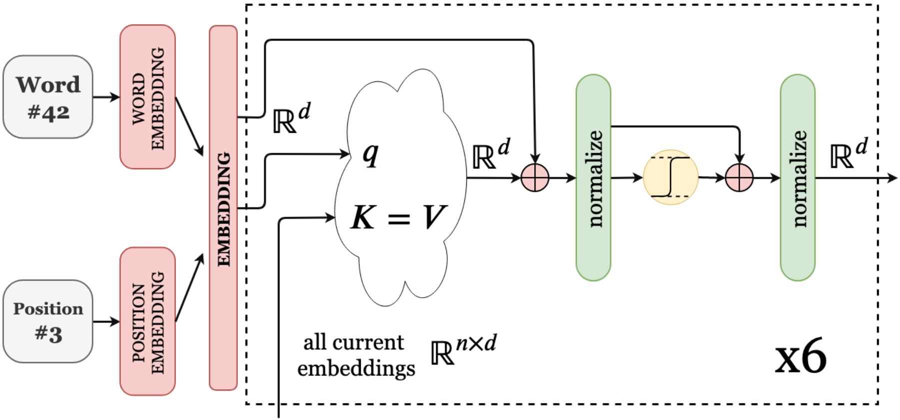


Word #42
WORD EMBEDDING
Position #3
POSITION EMBEDDING
EMBEDDING
$\mathbb{R}^d$
q
$K = V$
$\mathbb{R}^d$
normalize
all current embeddings
$\mathbb{R}^{n \times d}$

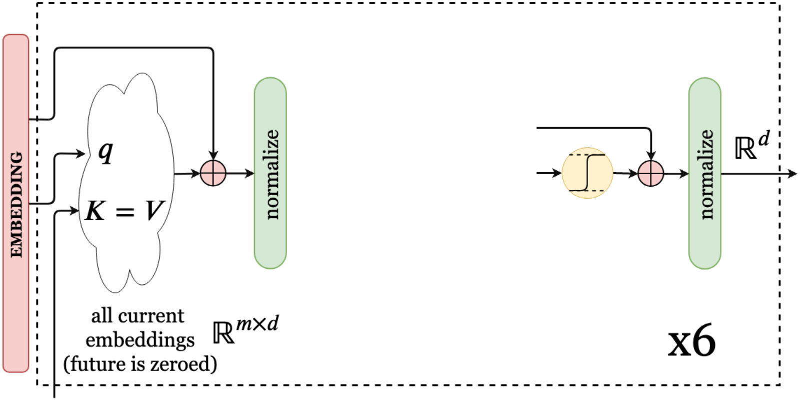
Encoder


Word
#42
WORD
EMBEDDING
Position
#3
POSITION
EMBEDDING
EMBEDDING
$\mathbb{R}^d$
q
$K = V$
$\mathbb{R}^d$
all current
embeddings
$\mathbb{R}^{n \times d}$
normalize
normalize
$\mathbb{R}^d$

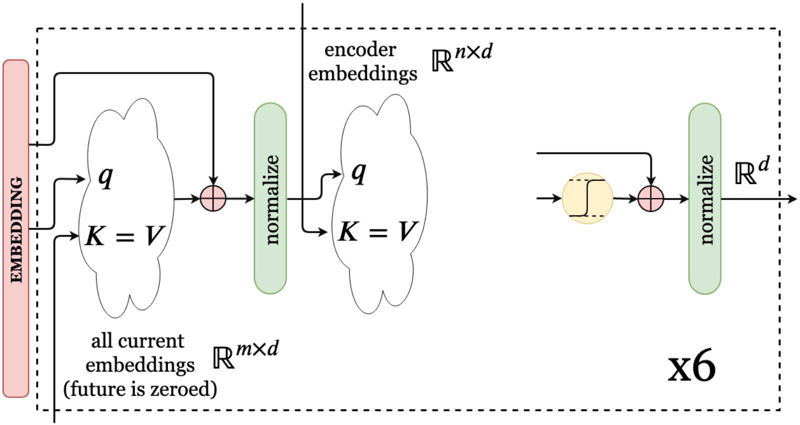
Encoder


Word
#42
WORD
EMBEDDING
Position
#3
POSITION
EMBEDDING
EMBEDDING
$\mathbb{R}^d$
q
$K = V$
$\mathbb{R}^d$
normalize
normalize
$\mathbb{R}^d$
all current
embeddings
$\mathbb{R}^{n \times d}$
x6

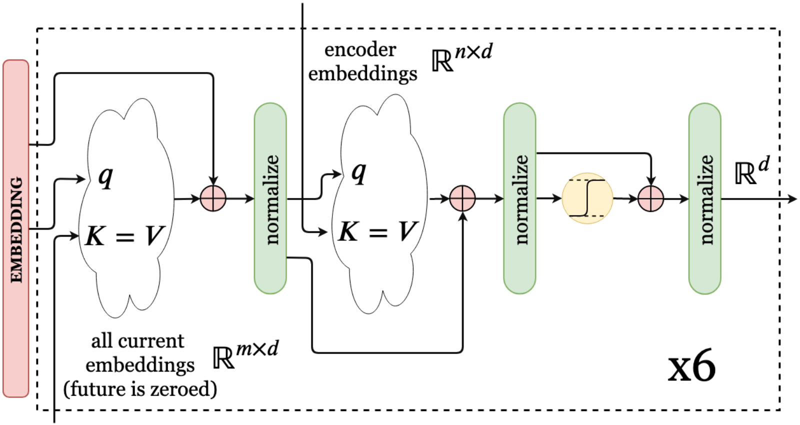
Decoder


EMBEDDING
q
$K = V$
normalize
all current embeddings (future is zeroed)
$\mathbb{R}^{m \times d}$
normalize
$\mathbb{R}^d$
x6

Decoder

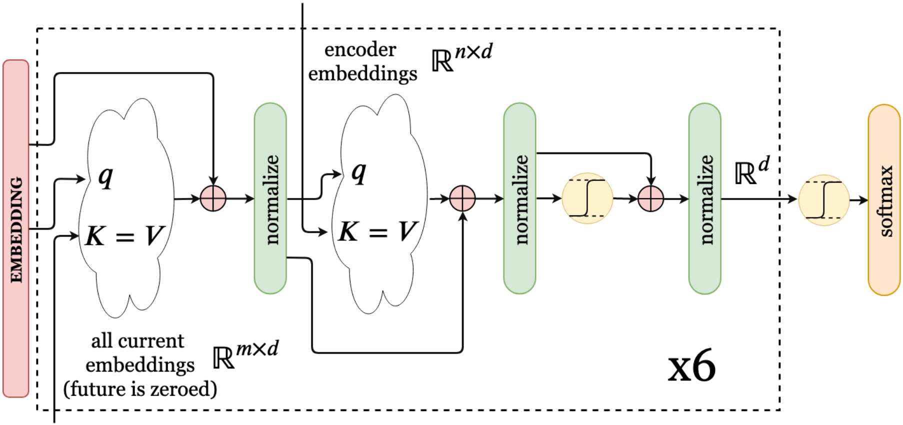

EMBEDDING
q
K = V
all current
embeddings
(future is zeroed)
$\mathbb{R}^{m\times d}$
normalize
encoder
embeddings
$\mathbb{R}^{n\times d}$
q
K = V
normalize
$\mathbb{R}^d$
x6

Decoder



EMBEDDING
q
K = V
all current embeddings (future is zeroed)
$\mathbb{R}^{m\times d}$
normalize
encoder embeddings
$\mathbb{R}^{n\times d}$
q
K = V
normalize
normalize
$\mathbb{R}^{d}$
x6

Decoder



EMBEDDING
q
K = V
all current
embeddings
(future is zeroed)
$\mathbb{R}^{m\times d}$
normalize
encoder
embeddings
$\mathbb{R}^{n\times d}$
q
K = V
normalize
normalize
$\mathbb{R}^{d}$
softmax
x6

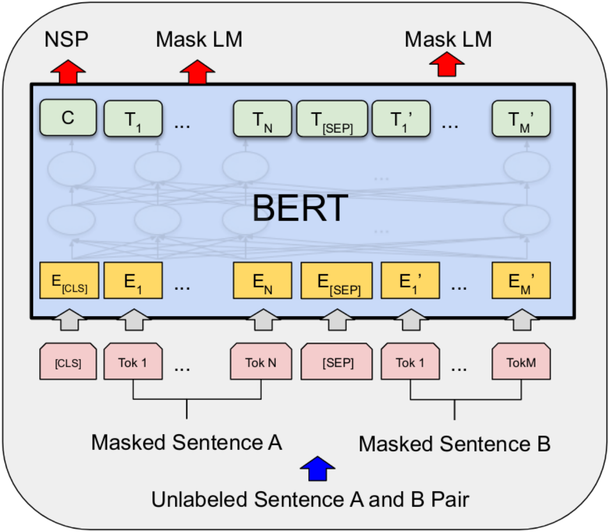
More sources about Transformer

- Animation of intuition: Link
- Illustrated Transformer
  https://jalammar.github.io/illustrated-transformer/
- OpenAI Blog
  https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
- MIPT Lecture (RUS)
  https://www.youtube.com/watch?v=Bg8Y5q1OiP0

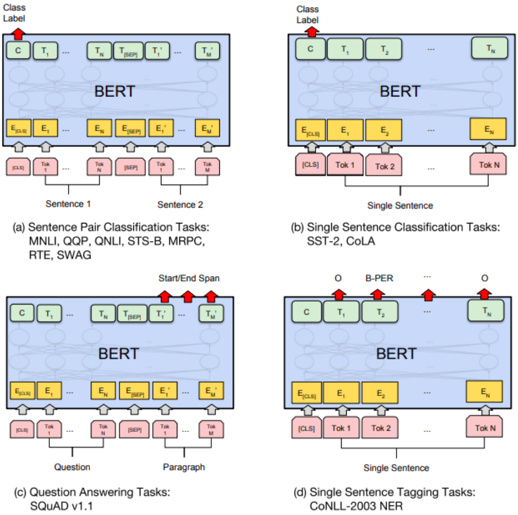
Section 5

BERT, GPT-2, ...

BERT: pre-training



BERT: fine-tuning



GPT-2 (Generative Pre-Training)

- language model based on masked multi-head self-attention
- with 1.5 billions of parameters (!)
- (rumors) 2048 TPU days to train
- which is able to generate pretty realistic texts

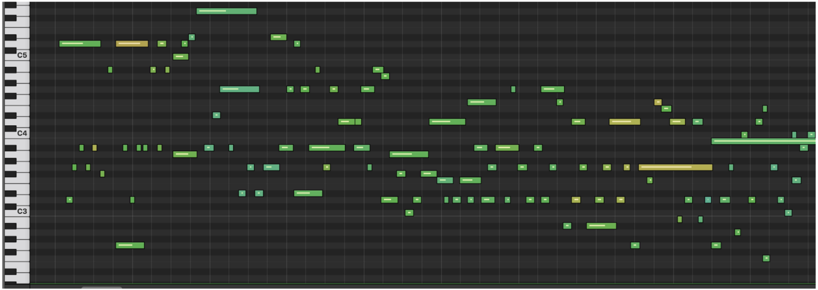
GPT-2 (Generative Pre-Training)

- language model based on masked multi-head self-attention
- with 1.5 billions of parameters (!)
- (rumors) 2048 TPU days to train
- which is able to generate pretty realistic texts

How to apply this to music?

How to apply this to music?

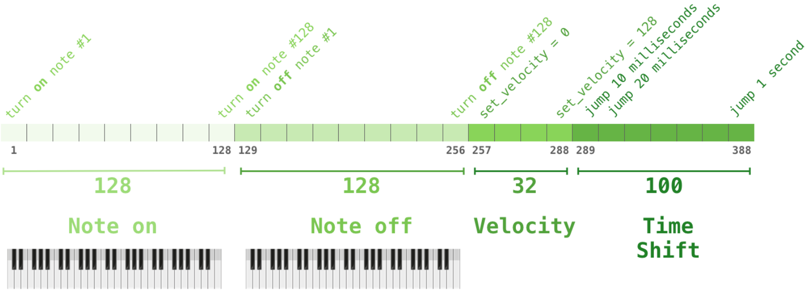
MIDI Music Representation



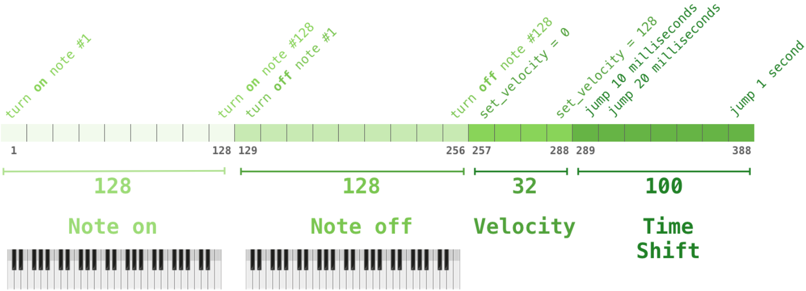
Each note has:

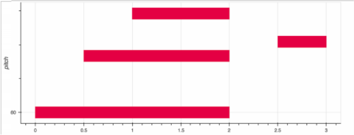
- beginning (milliseconds)
- end (milliseconds)
- pitch (key): 128 possible options
- velocity (128 possible options; but we can take smaller grid)

Notes is Language



Notes is Language





SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>,
NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>

Looking for more details...

- Generalized Language Modeling (BERT section)
  https://lilianweng.github.io/lil-log/2019/01/31/generalized-language-models.html#bert
- Illustrated GPT
  http://jalammar.github.io/illustrated-gpt2/
- Transformer-XL
  https://arxiv.org/abs/1901.02860
- Music Transformer
  https://magenta.tensorflow.org/music-transformer
- Generated music visualisation: Link