بسم الله الرحمن الرحیم

# Transformer Language models

## Lecture 2 - Decoder-only models

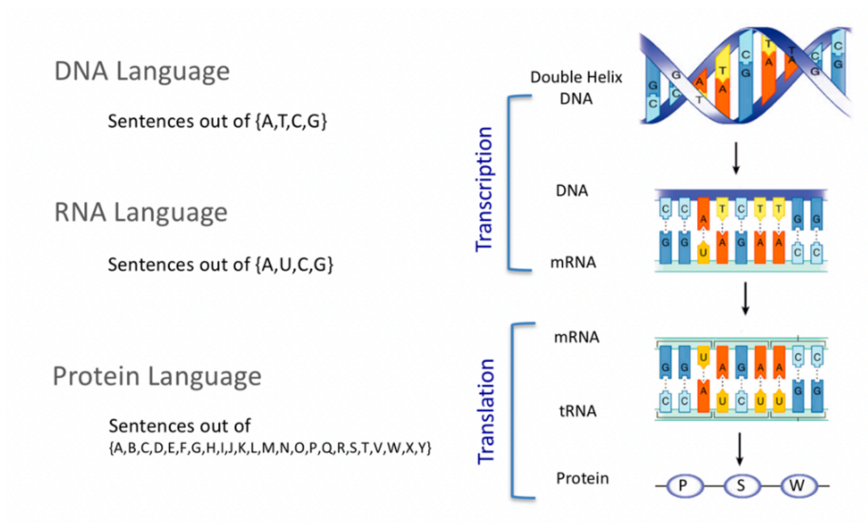
Oct. 1st 2023

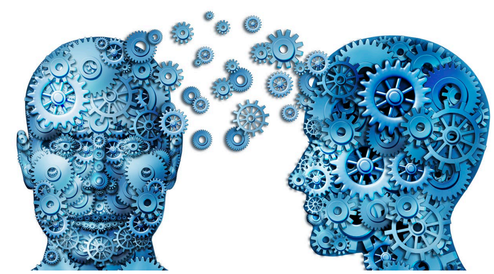
Artificial Intelligence Group
Computer Engineering Department, SUT

# –Language definition

# Language Definition

Chomsky (1959: 137) *"A language is a collection of* ***sentences*** *of* ***finite length*** *all constructed from a* ***finite alphabet*** *(or, where our concern is limited to syntax, a finite vocabulary) of symbols."*



# −Distributional Hypothesis

# Distributional hypothesis

J.R. Firth

Firth (1950) *"a word is characterized by the company it keeps"*

زیادتی مطلب، کار بر خود آسان کن

صراحی می لعل و بتی چو ماهت بس

A picture of a good friendship circle in Persian culture
Made with Bing Image Creator. Powered by DALL-E

REVIEW

## Language modeling

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop})$$

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

REVIEW

# N–gram Language modeling

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

## Bi-gram

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-1})$$

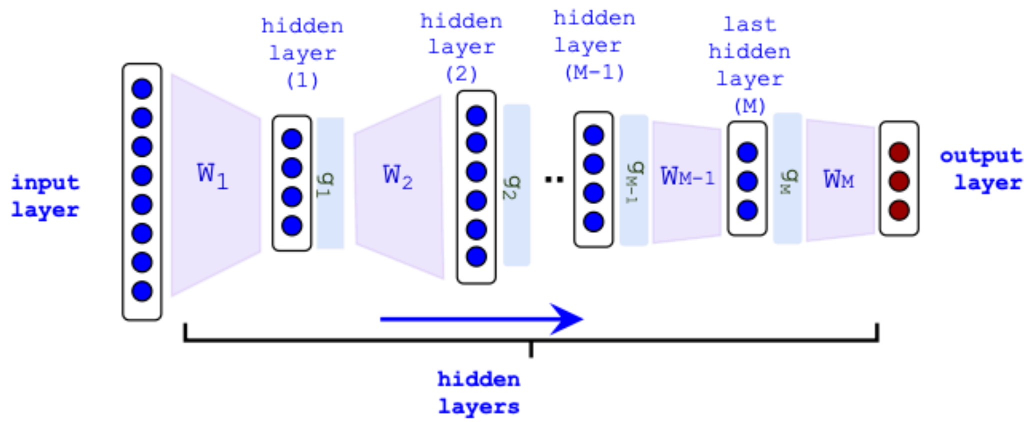
## Markov (m$^{\text{th}}$ order)

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-m}, \ldots, w_{i-1})$$

REVIEW

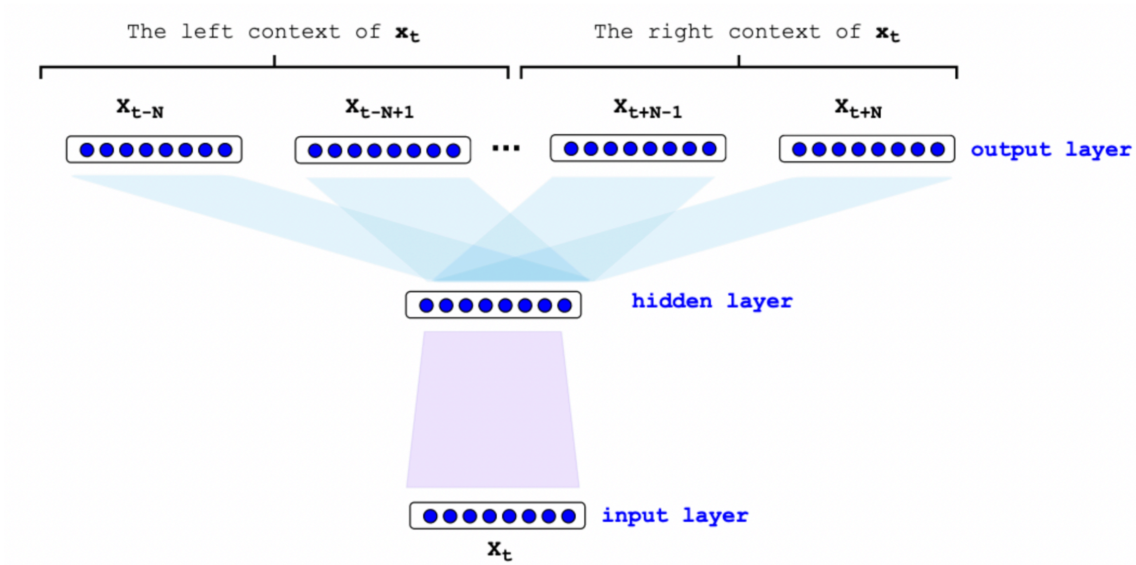
# Neural Language modeling

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop})$$



REVIEW

# Skip–gram – Similar to Language Models




$$\sum_{t=1}^{M} \sum_{c \in [t-N, t+N]} \log p(w_c \mid w_t)$$

$$p(w_c \mid w_t; \theta) = \frac{e^{v_c \cdot v_t}}{\sum_{c' \in \mathcal{C}} e^{v_{c'} \cdot v_t}}$$

$\longrightarrow$

Maximizing the following likelihood:

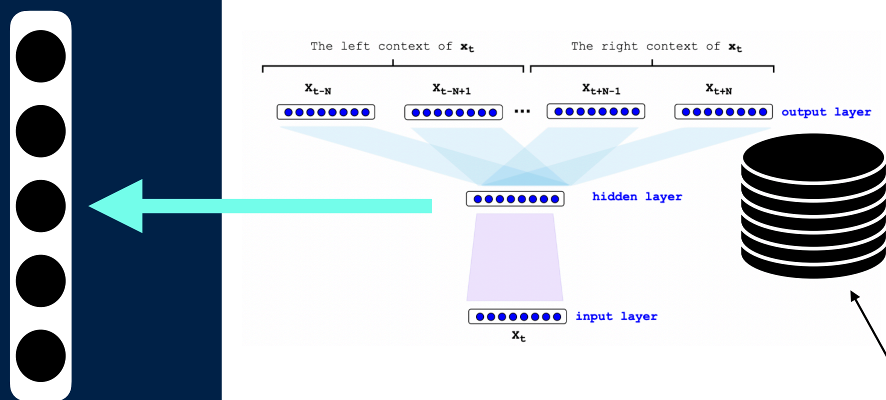
$$\sum_{t=1}^{T} \left[ \sum_{c \in [t-N, t+N]} \log\left(1 + e^{-s(w_t,\ w_c)}\right) + \sum_{w_r \in \mathcal{N}_{t,c}} \log\left(1 + e^{s(w_t,\ w_r)}\right) \right]$$

$$s(w_t,\ w_c) = {v_t}^{\top} \cdot v_c$$

REVIEW

مهر

# Fixed embeddings – Skip-gram




...که دگر نه عشق خورشید و نه مهر ماه دارم

فروزنده ماه و ناهید و مهر...

...فروشست از نگار و نقش ماه مهر و آبانش

## Fixed embeddings – Skip-gram

آبان
ماه
مهر
خورشید
ناهید

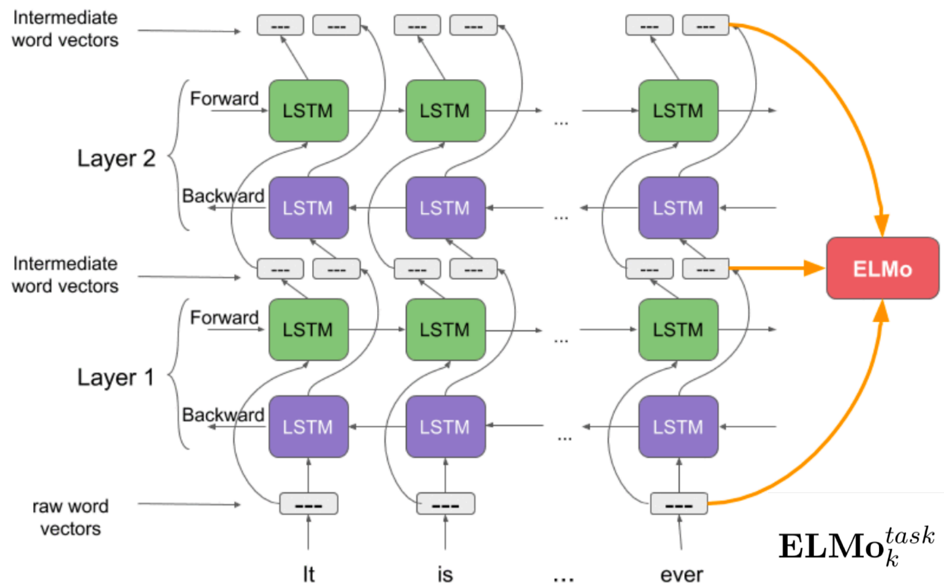
...که دگر نه عشق خورشید و نه مهر ماه دارم

فروزنده ماه و ناهید و مهر...

...فرو شست از نگار و نقش ماه مهر و آبانش

REVIEW

# ELMO: Deep contextualized word representations




$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

ELMo (Peters et al., 2018; NAACL 2018 best paper)

- Train two separate unidirectional LMs (left-to-right and right-to-left) based on LSTMs
- Feature-based approach: pre-trained representations used as input to task-specific models

# – Self–attention

آبان
ماه
مهر
خورشید
ناهید

## How to contextualize the fixed embeddings?

...که دگر نه عشق خورشید و نه مهر ماه دارم

فروزنده ماه و ناهید و مهر...

...فروشست از نگار و نقش ماه مهر و آبانش

Attention

## Self–Attention Idea

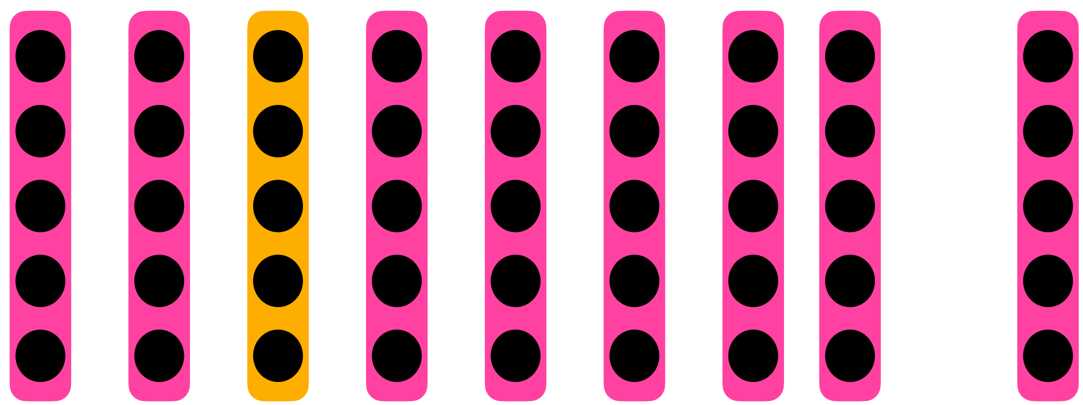
Input embeddings

$$x_1, x_2, \ldots, x_n$$

Output embeddings

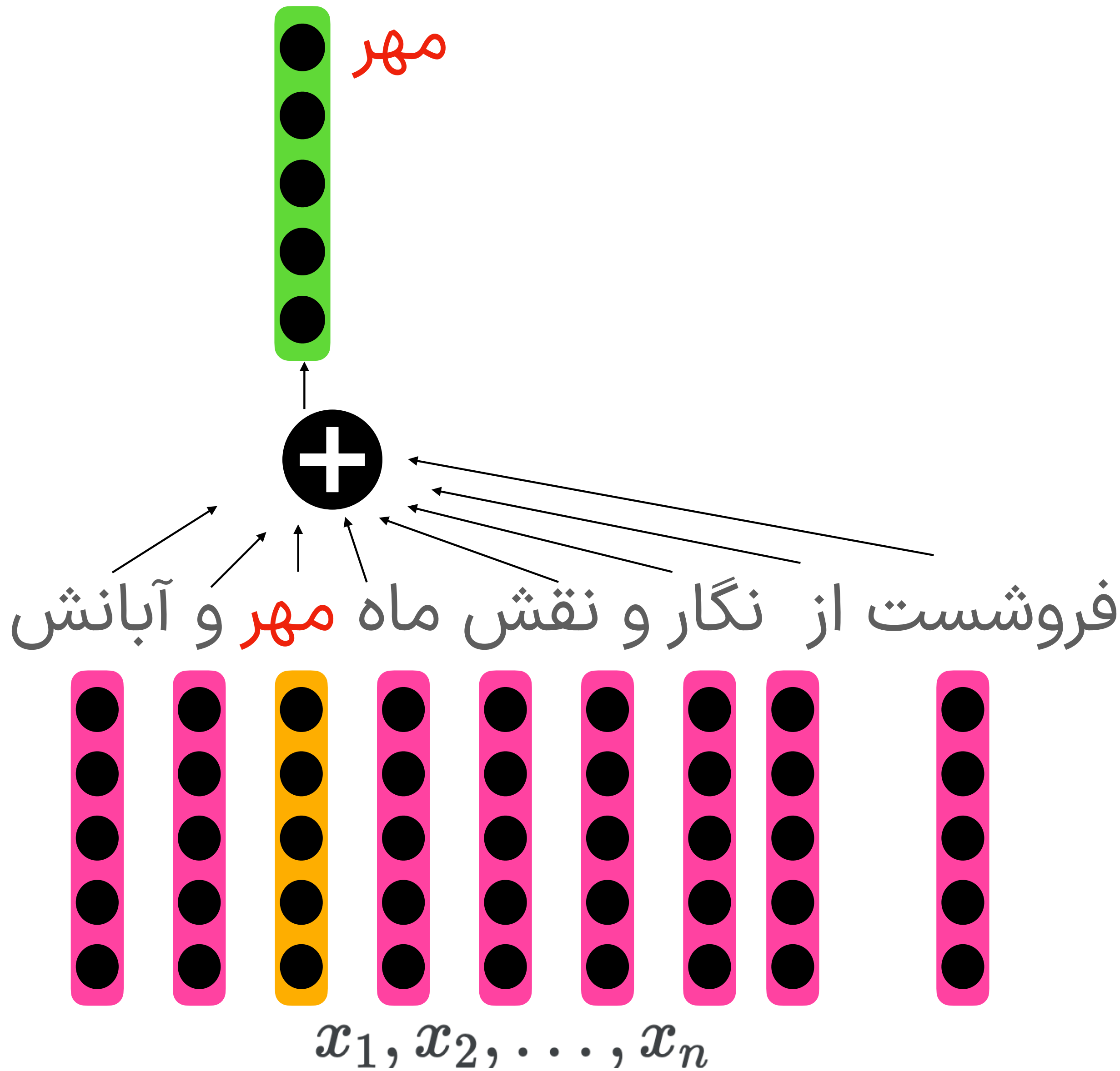
$$y_1, y_2, \ldots, y_n$$

فرو شست از نگار و نقش ماه مهر و آبانش



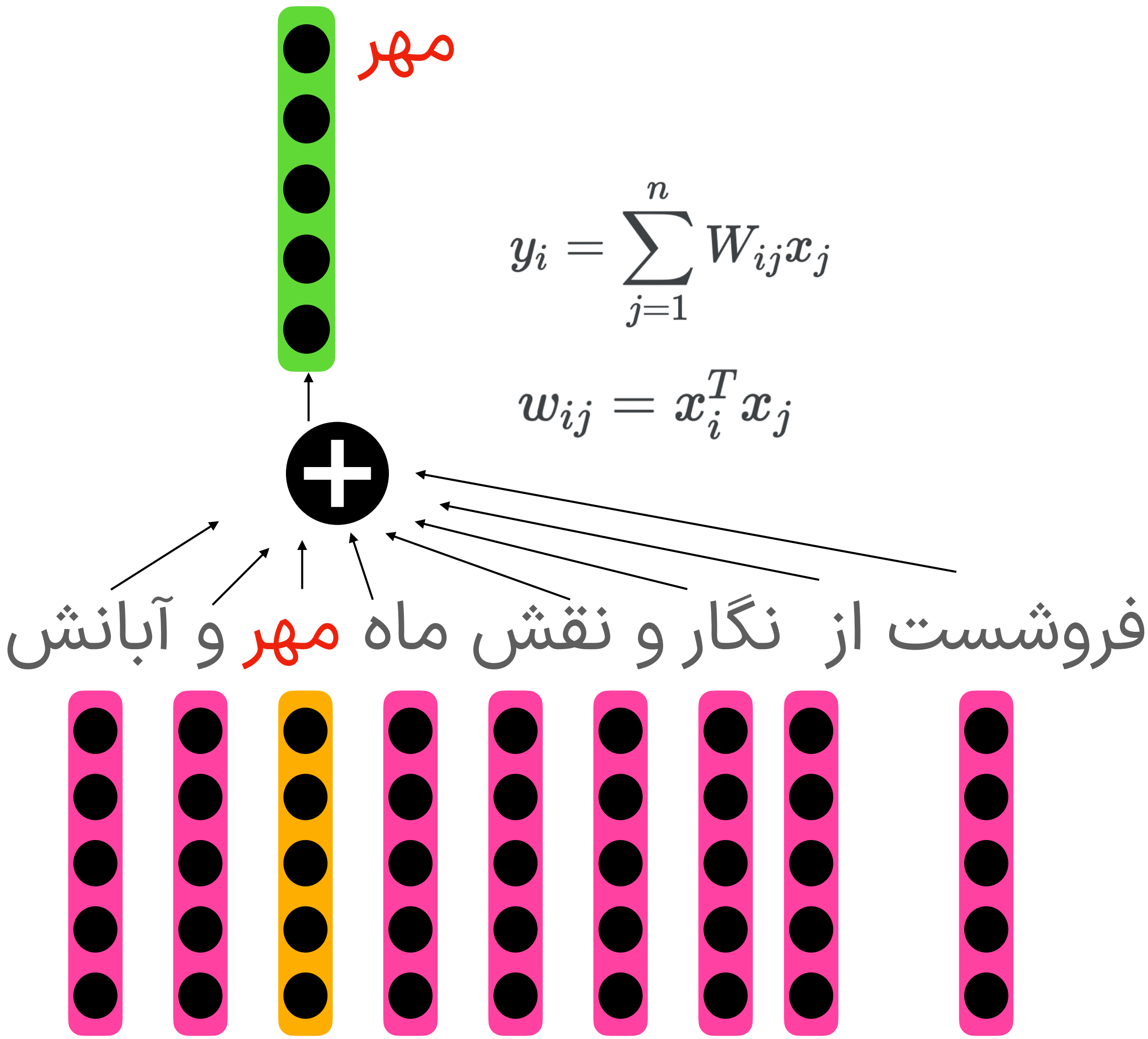
# Self-Attention Idea

Attention



## Self-Attention

# Attention




$$y_i = \sum_{j=1}^{n} W_{ij} x_j$$

$$w_{ij} = x_i^T x_j$$

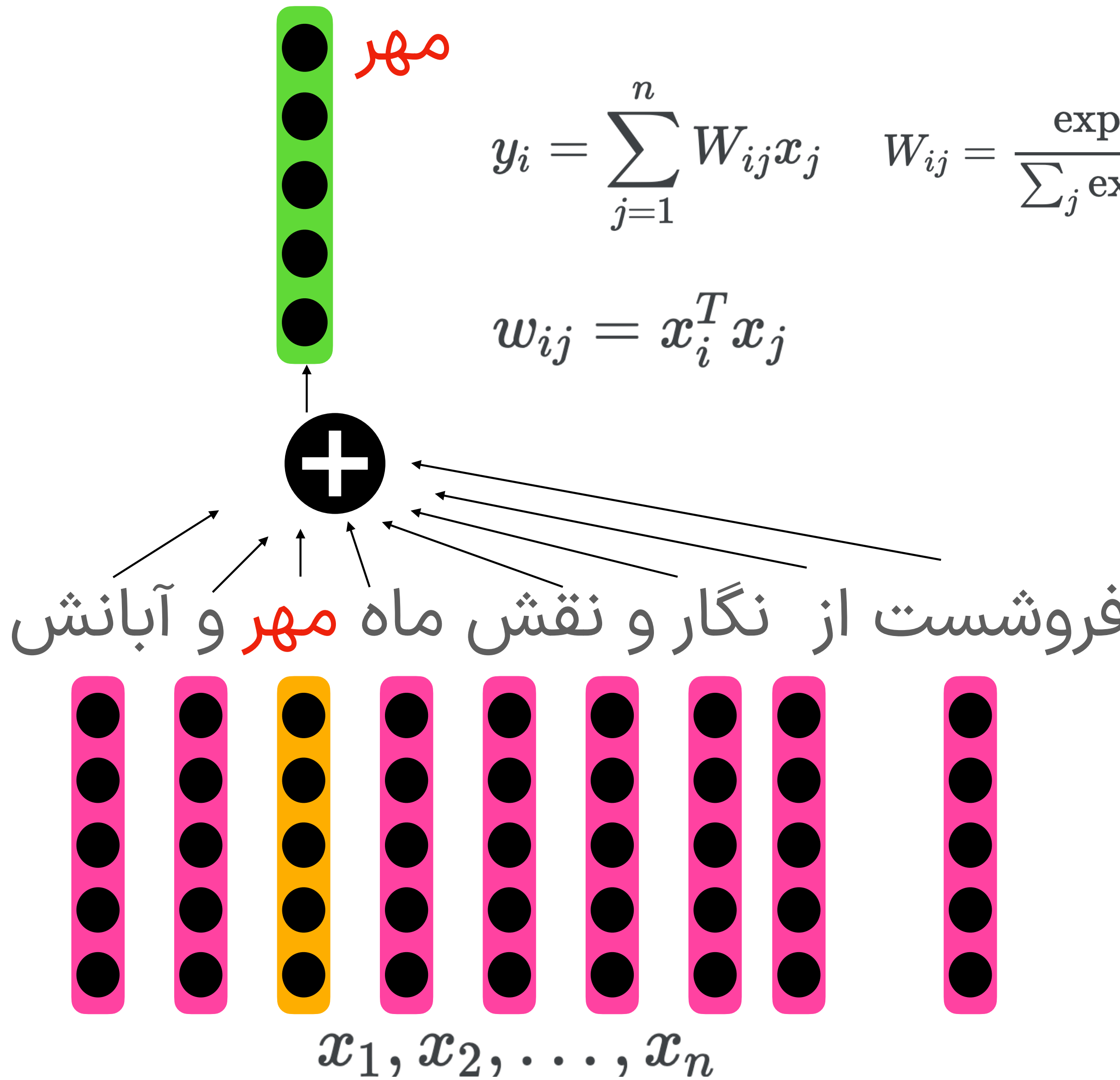
$$x_1, x_2, \ldots, x_n$$

## Self-Attention

# Attention

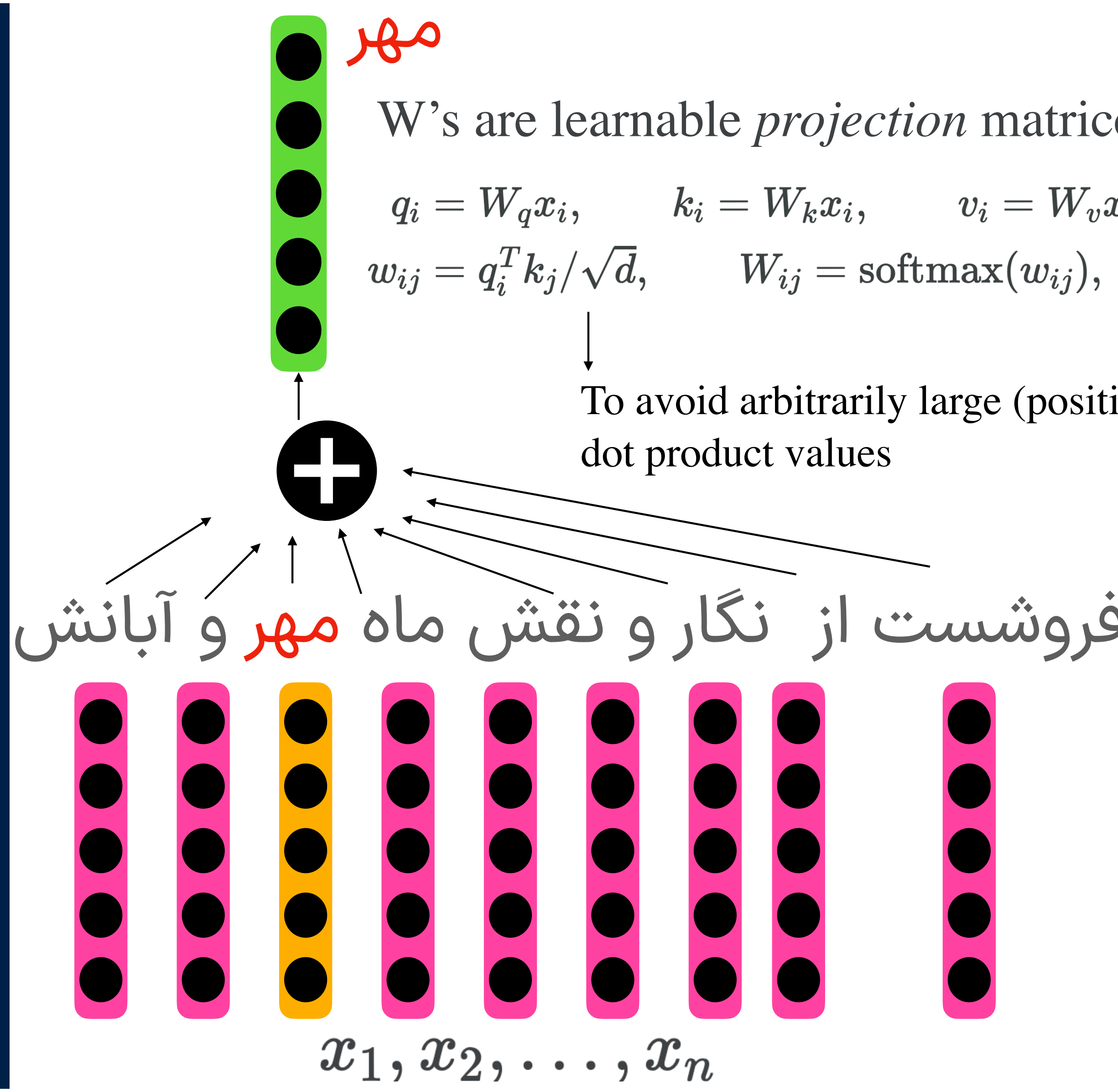
$$y_i = \sum_{j=1}^{n} W_{ij} x_j \qquad W_{ij} = \frac{\exp w_{ij}}{\sum_j \exp w_{ij}}$$

$$w_{ij} = x_i^T x_j$$



## Attention

Attention




W's are learnable *projection* matrices

$$q_i = W_q x_i, \qquad k_i = W_k x_i, \qquad v_i = W_v x_i$$

$$w_{ij} = q_i^T k_j / \sqrt{d}, \qquad W_{ij} = \mathrm{softmax}(w_{ij}), \qquad y_i = \sum_j W_{ij} v_j.$$

To avoid arbitrarily large (positive or negative) dot product values

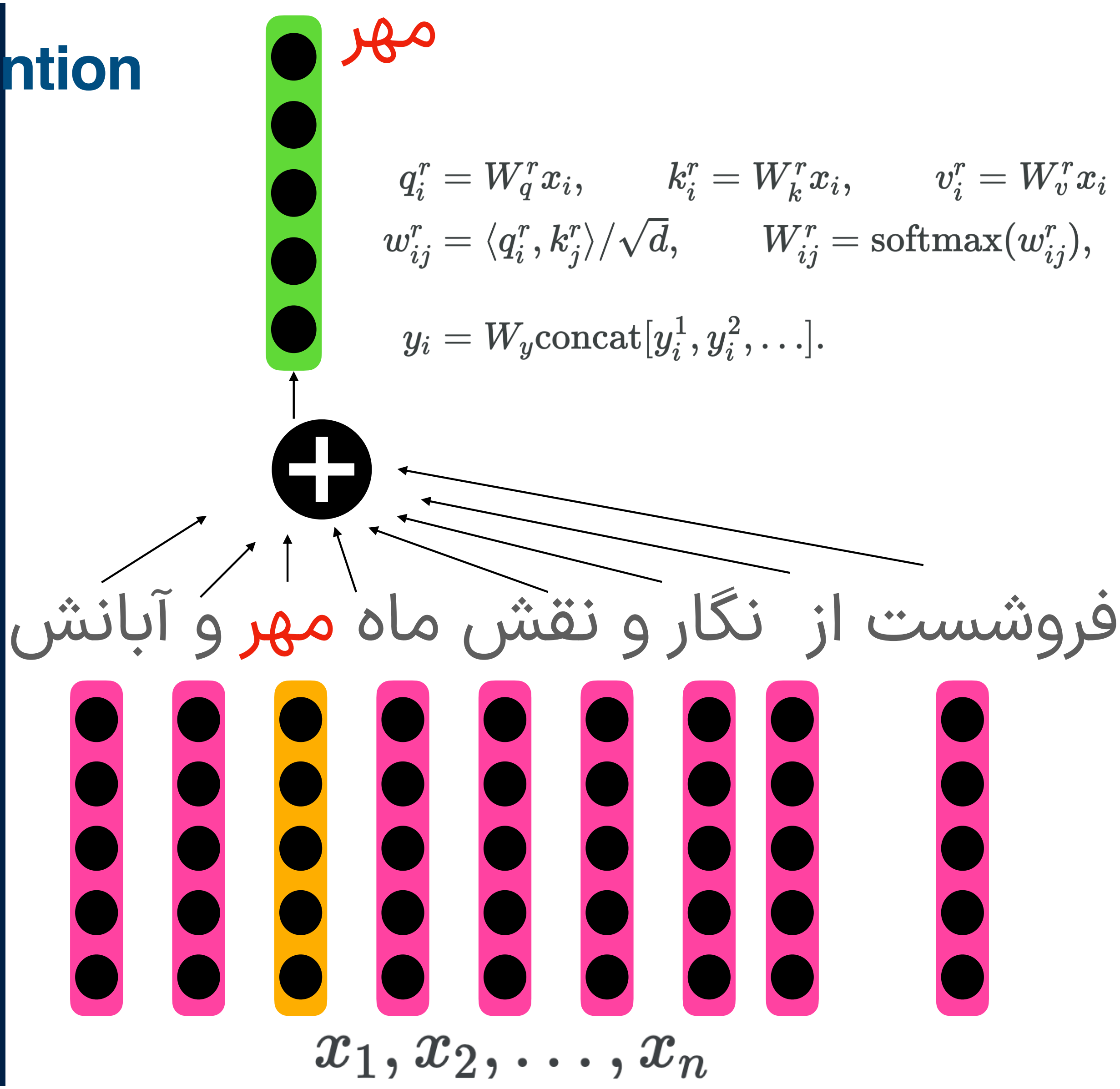
# Attention

Attention

- Inputs: a query q and a set of key-value (k-v) pairs to an output
  - All presented as vectors
- Output is weighted sum of values
- Weight of each value: inner product of query and corresponding key

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

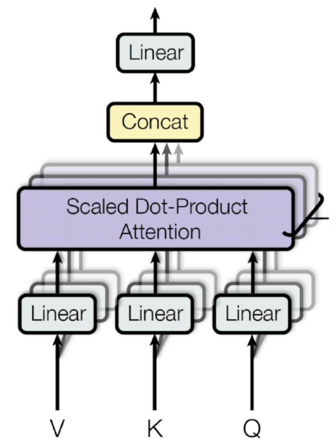
# Multihead Attention




$$q_i^r = W_q^r x_i, \qquad k_i^r = W_k^r x_i, \qquad v_i^r = W_v^r x_i$$

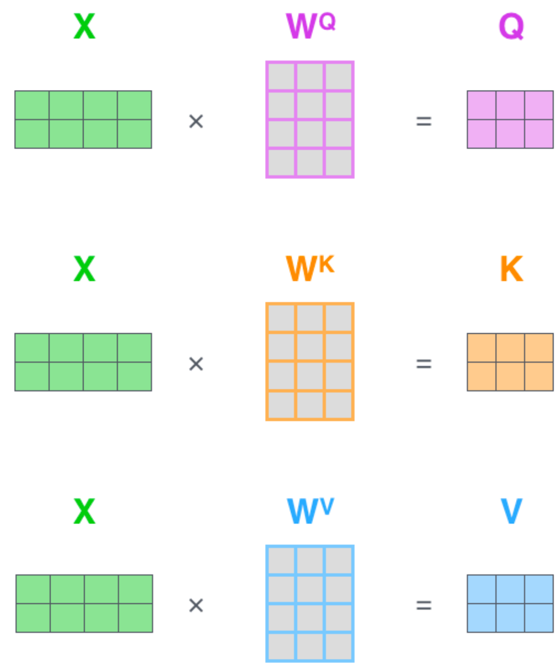
$$w_{ij}^r = \langle q_i^r, k_j^r \rangle / \sqrt{d}, \qquad W_{ij}^r = \mathrm{softmax}(w_{ij}^r), \qquad y_i^r = \sum_j W_{ij}^r v_j,$$

$$y_i = W_y \mathrm{concat}[y_i^1, y_i^2, \ldots].$$



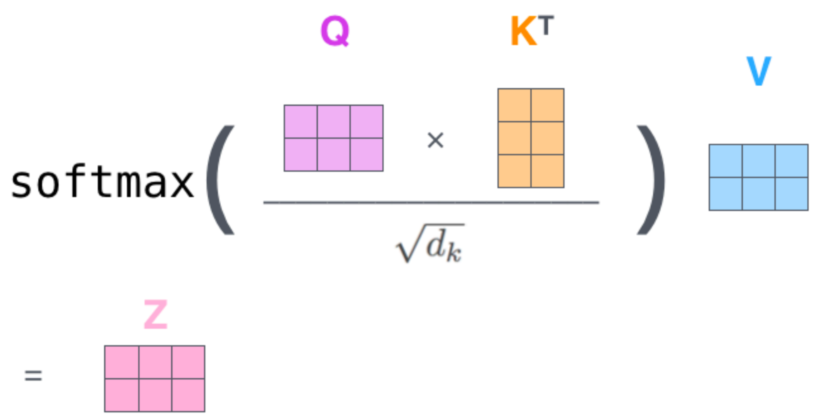
Attention

# Matrix Attention



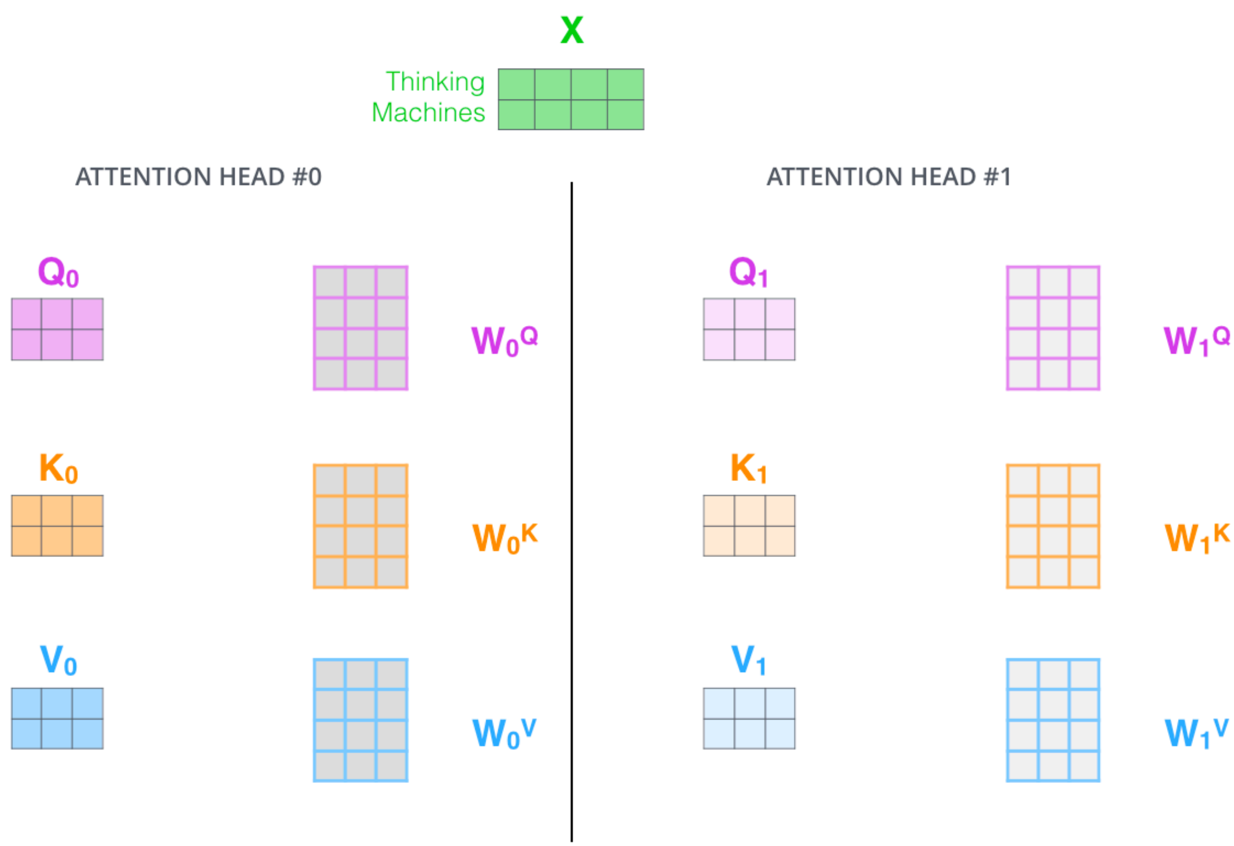
Attention

## Matrix Attention



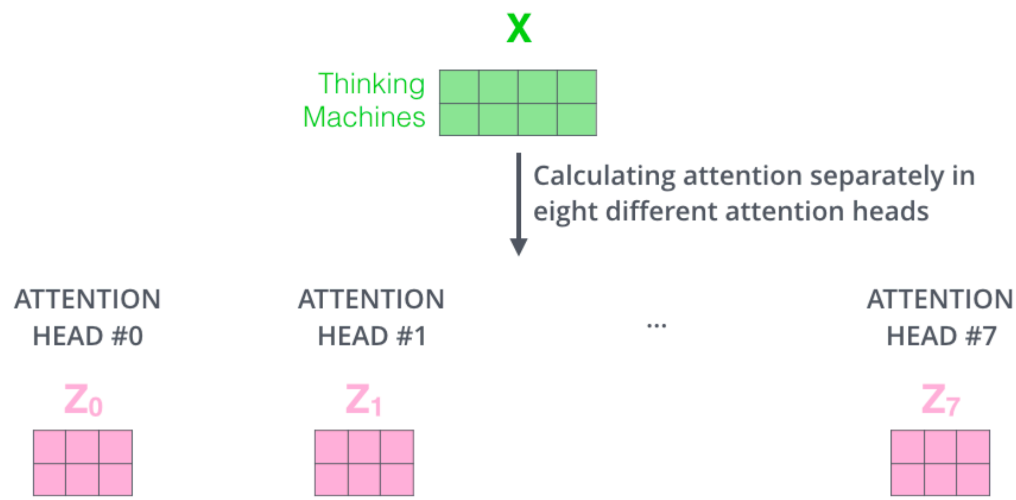
Attention

## Matrix Attention



Attention

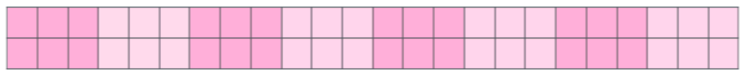
# Matrix Attention



# Attention

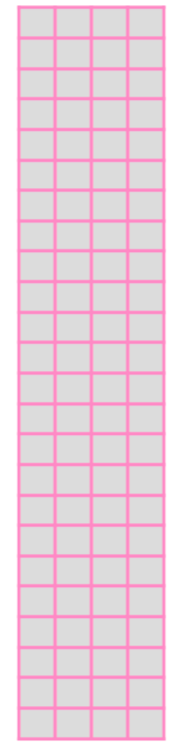
1) Concatenate all the attention heads

$Z_0$ $Z_1$ $Z_2$ $Z_3$ $Z_4$ $Z_5$ $Z_6$ $Z_7$



2) Multiply with a weight matrix $W^O$ that was trained jointly with the model

X

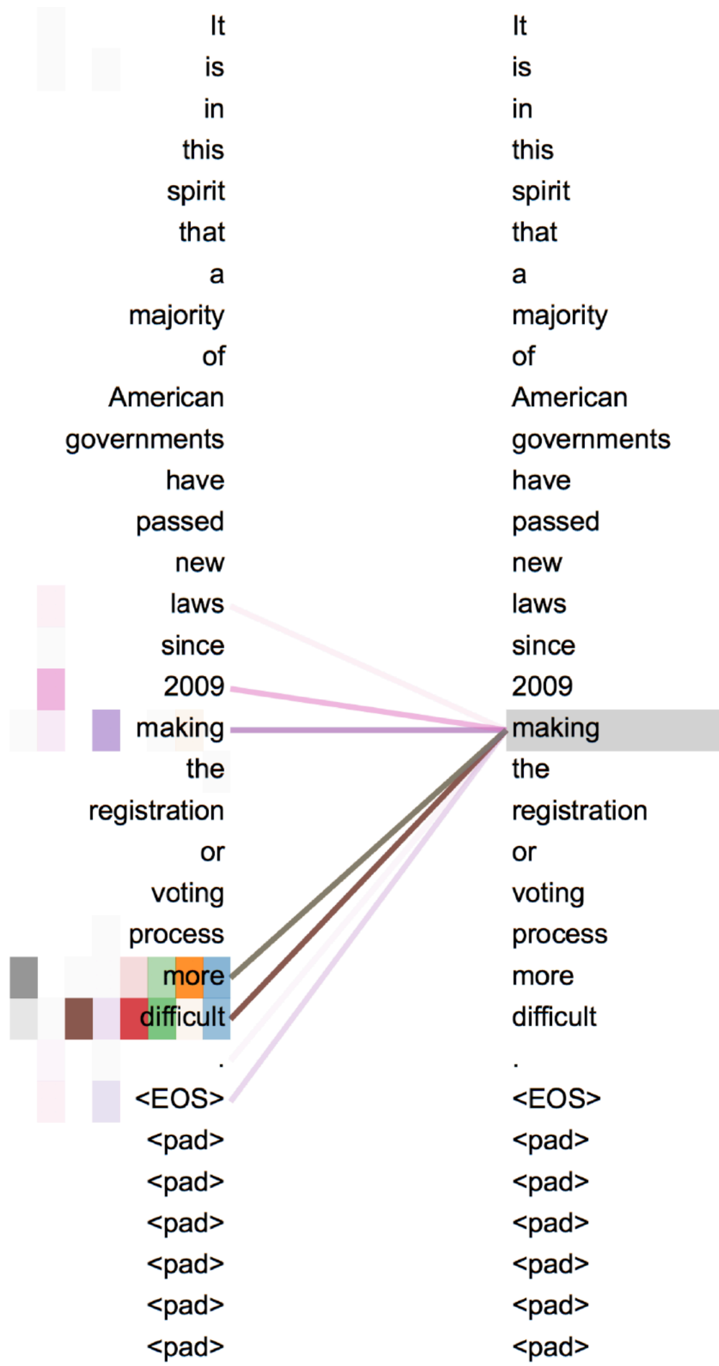


$W^O$

3) The result would be the $Z$ matrix that captures information from all the attention heads. We can send this forward to the FFNN

=

$Z$



It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad>
It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad>

# – Transformers

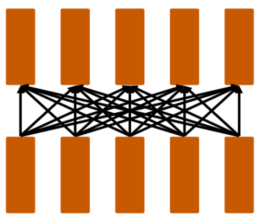
# Attention Is All You Need

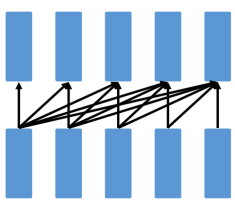
**Ashish Vaswani*** Google Brain avaswani@google.com
**Noam Shazeer*** Google Brain noam@google.com
**Niki Parmar*** Google Research nikip@google.com
**Jakob Uszkoreit*** Google Research usz@google.com
**Llion Jones*** Google Research llion@google.com
**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu
**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com
**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com

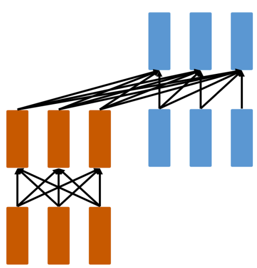
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

https://arxiv.org/abs/1706.03762

−Next lecture


Encoders


Decoders


Encoder-Decoders

# Transformer Architectures

- Encoder-only (e.g., BERT): bidirectional contextual embeddings
- Decoder-only (e.g., GPT-x): unidirectional contextual embeddings, generate one token at a time
- Encoder-decoder (e.g., T5): encode input, decode output