

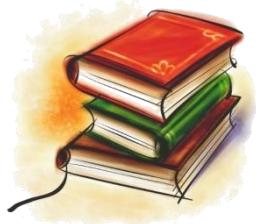
# شبکه‌های عصبی مصنوعی

## ساز و کار توجه و مبدل‌ها

هادی ویسی

[h.veisi@ut.ac.ir](mailto:h.veisi@ut.ac.ir)

دانشگاه تهران - دانشکده علوم و فنون نوین



## فهرست

- مرور شبکه‌های بازگشتی: معماری Seq2Seq
- شبکه‌های بازگشتی: ساز و کار توجه
- مراحل محاسبه توجه
- انواع توجه

(Self-Attention)  
(Multi-Head Attention)

- مبدل (Transformer)
- ساختار رمزگذار
- ساختار رمزگشا
- ساختار آخرین لایه خطی و SoftMax
- کاربردهای مبدل‌ها

○ مدل زبانی و تعییه کلمات BERT

○ پروژه GPT-3

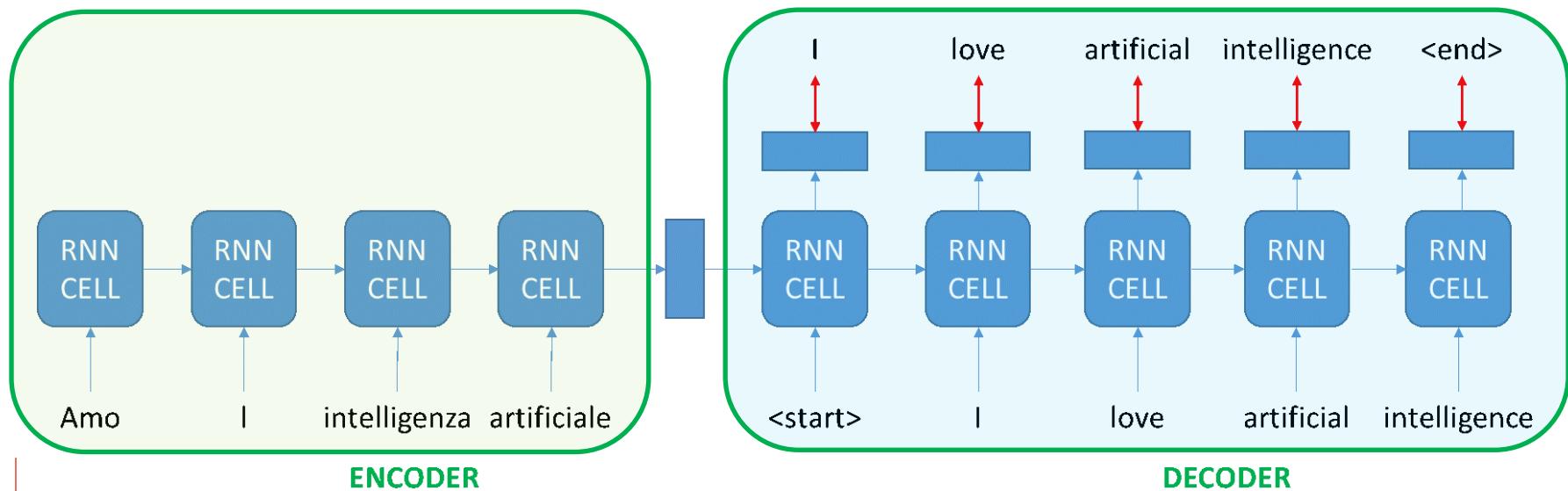
○ مبدل‌های بینایی و صوتی

# مرور شبکه‌های بازگشتی: معماری ...Seq2Seq

## نحوه کار کردن

- رمزگذار (Encoder): کد کردن دنباله ورودی در یک بردار بافت (Context)
- رمزگشای (Decoder): پیش‌بینی کلمه بعدی با دریافت بردار بافت و کلمه قبلی

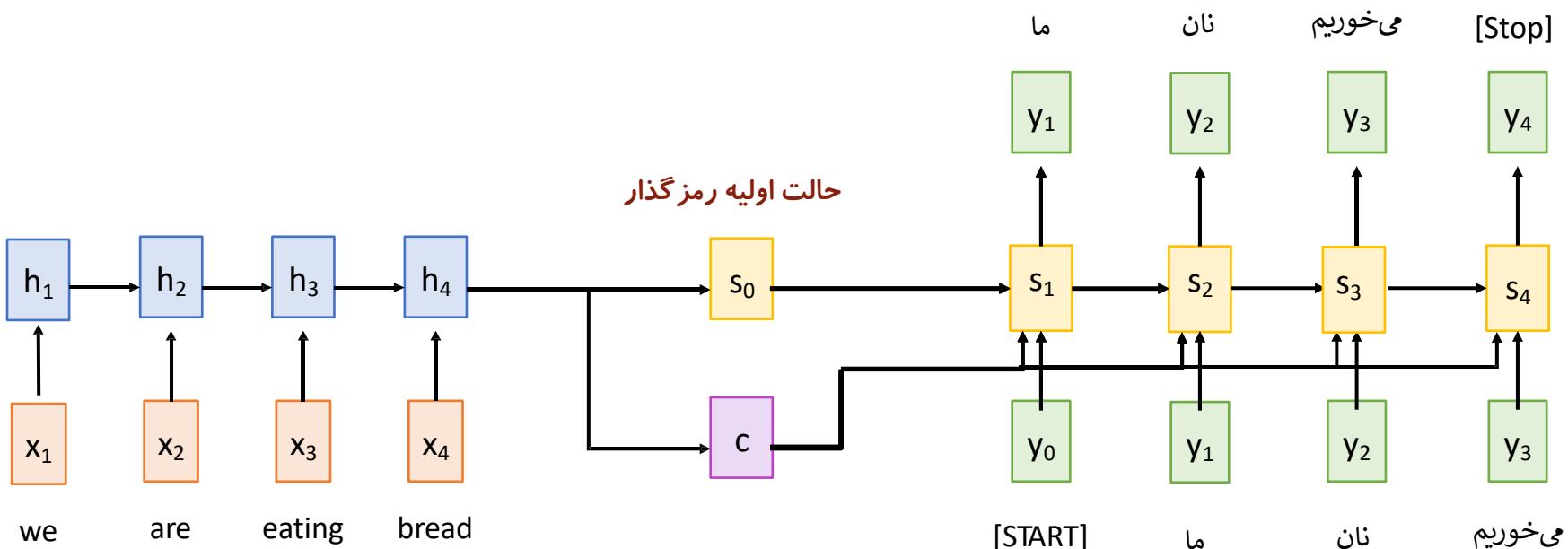
- مثال: ترجمه ماشینی



# شبکه‌های بازگشتی: ... Seq2Seq

## نگاشت کردن دو دنباله به هم‌دیگر

- ورودی: دنباله  $x_1, x_2, \dots$
- خروجی: دنباله  $y_1, y_2, \dots$



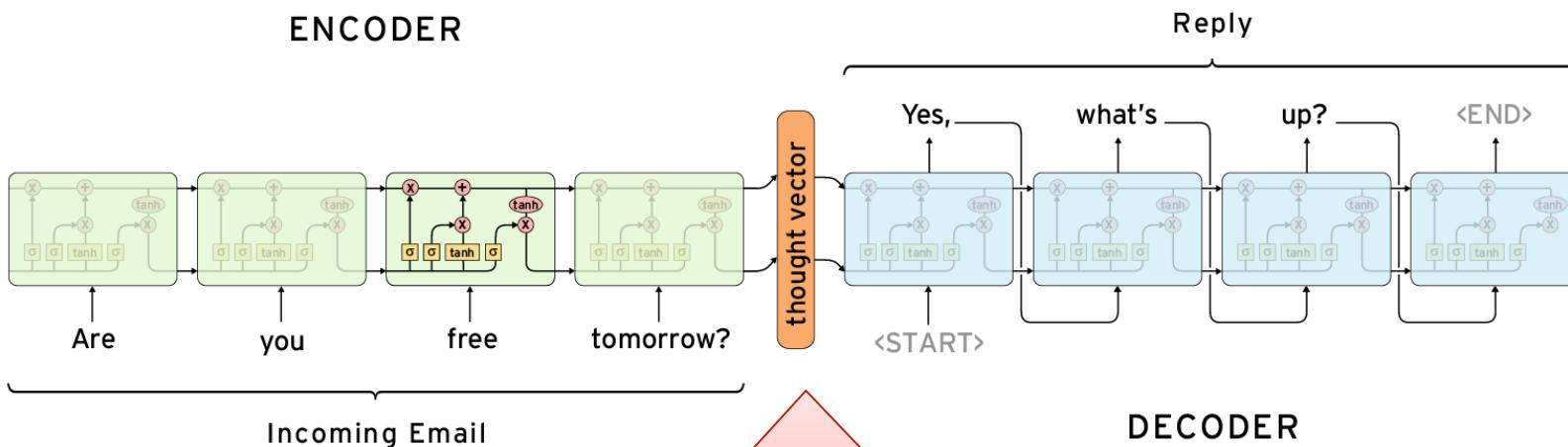
$$h_t = f_w(x_t, h_{t-1}) \text{ : رمزگذار}$$

$$s_t = g_u(y_{t-1}, s_{t-1}, C) \text{ : رمزگشا}$$

# مرور شبکه‌های بازگشتی: معماری ...Seq2Seq

## ○ دستیار هوشمند (چت بات)

- ورودی: دنباله کلمات ورودی (تبديل هر کلمه به بردار با روش‌های Word Embedding)
- خروجی: دنباله کلمات پاسخ



نیاز به بهبود بردار بافت برای رفع مشکل  
فراموشی در دنباله‌های طولانی

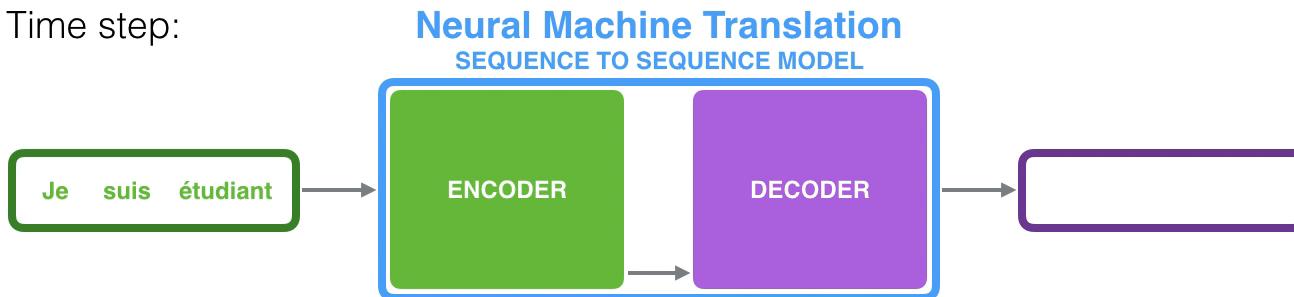
# مرور شبکه‌های بازگشتی: ...Seq2Seq

○ یک بردار بافت به ازای کل دنباله ورودی

- بردار بافت = خروجی بلوک‌های آخرین لایه مخفی رمزگذار = ورودی رمزگشا

- ابعاد رایج: ۱۰۲۴، ۵۱۲ و ۲۵۶

Time step:





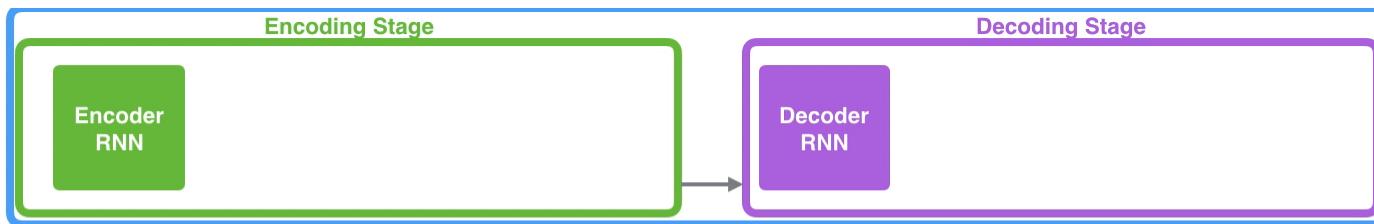
# مرور شبکه‌های بازگشتی: معماری ...Seq2Seq

○ یک بردار بافت به ازای کل دنباله ورودی

- بردار بافت = خروجی بلوک‌های آخرین لایه مخفی رمزگذار = ورودی رمزگشا

- ابعاد رایج: ۱۰۲۴، ۵۱۲ و ۲۵۶

- مثال: ترجمه ماشینی



Je suis étudiant

# شبکه‌های بازگشتی: ساز و کار توجه ...

## توجه (Attention) ...

- توجه بیشتر انسان به اطلاعات مهم در پردازش داده (تصویر، متن و ...)

### مثال: عنوان گذاری تصویر



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.

دیدگاه کاربران

پرسش و پاسخ

کاربر دیجی‌کالا ۱۴۰۰ | اردیبهشت ۱۳۹۸

بعد از سه روز استفاده باید بگم از هر جهت گوشی کامل و بی نقصیه

آیا این دیدگاه برایتان مفید بود؟

### مثال: تحلیل احساس

# شبکه‌های بازگشتی: ساز و کار توجه ...

## توجه (Attention)

- نواحی توجه در مساله عنوان گذاری تصویر

*A woman is throwing a frisbee in a park*



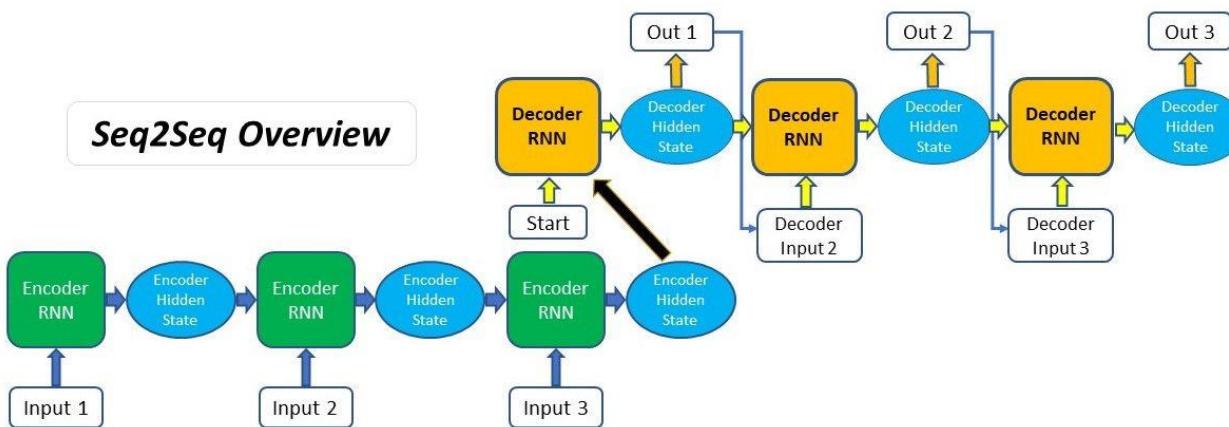
# شبکه‌های بازگشتی: ساز و کار توجه ...

## مشکل

- عدم امکان مدلسازی دقیق دنباله طولانی با یک بردار بافت به عنوان ورودی رمزگشا

## راه حل

- در نظر گرفته خروجی همه حالت‌های میانی رمزگذار در فرایند رمزگشایی
- برای تولید هر خروجی توسط رمزگشا، از همه حالت‌های میانی رمزگذار استفاده می‌شود
- وزن دادن (توجه) بیشتر به حالت‌های میانی مرتبط با خروجی جاری





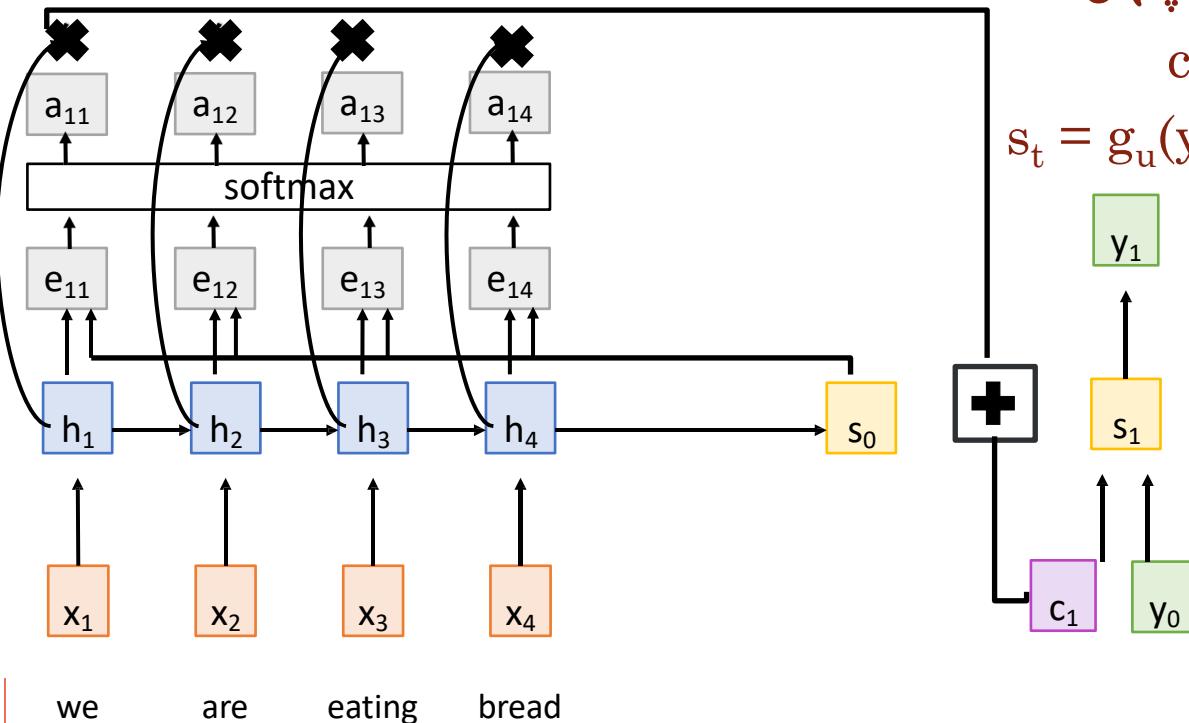
## شبکه‌های بازگشتی: ساز و کار توجه ...

### راه حل

- محاسبه‌ی وزن‌های توجه (Softmax)
- نرمال‌سازی وزن‌های توجه
- جمع وزن‌دار حالت‌های پنهان

$$s_t = g_u(y_{t-1}, s_{t-1}, c_t)$$

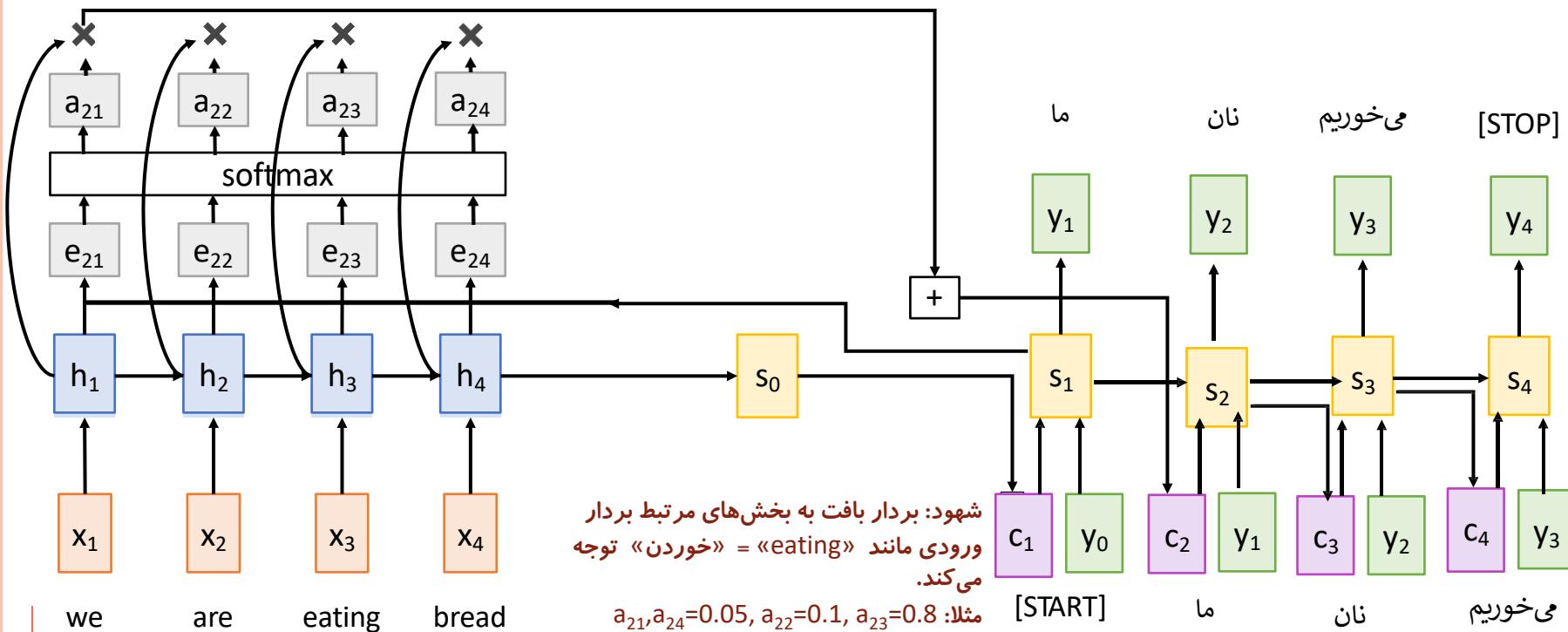
شهود: بردار بافت  $C$  به بخش‌های مشابه ورودی توجه می‌کند



# شبکه‌های بازگشتی: ساز و کار توجه ...

## راه حل

- تکرار مراحل: استفاده از  $S_1$  برای محاسبه بردار بافت  $C_2$  (وزن جدید)
- تکرار مراحل: استفاده از  $S_2$  برای محاسبه بردار بافت  $C_3$  (وزن جدید)

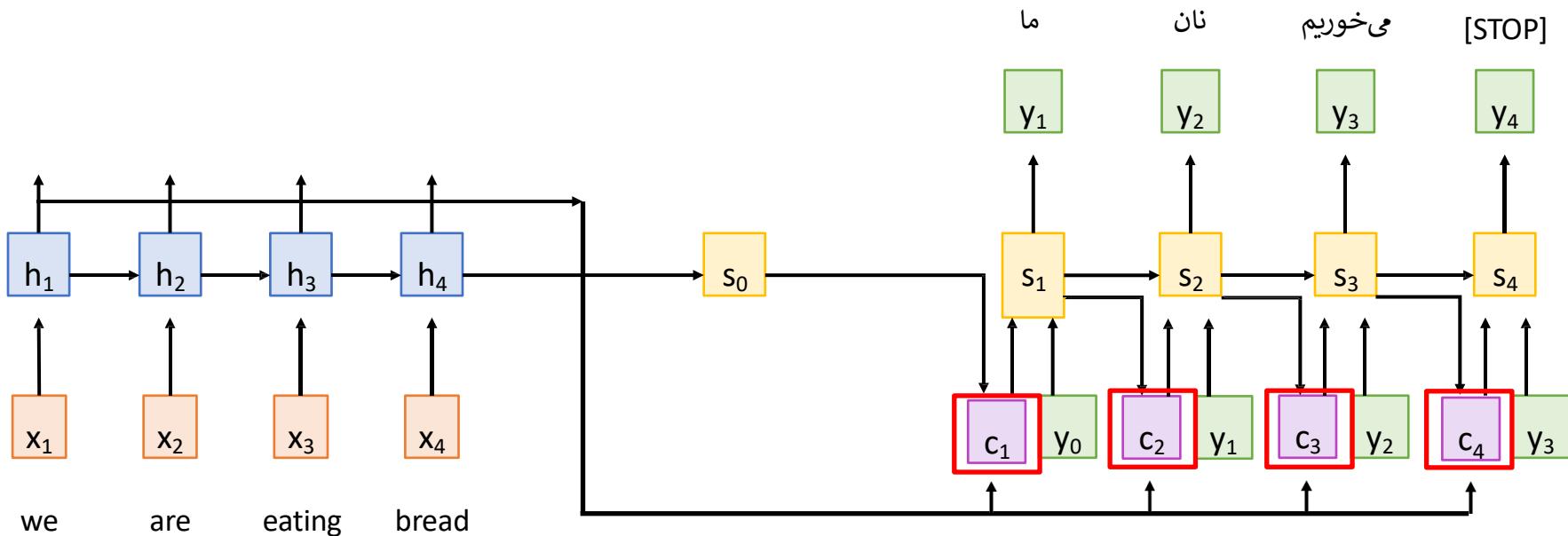




## شبکه‌های بازگشتی: سازوکار توجه ...

### ○ سازوکار توجه ...

- در هر مرحله زمانی رمزگشا از یک بردار بافت متفاوت استفاده می‌کند
- دنباله ورودی محدود به یک بردار گلوگاه (bottlenecked) به عنوان بردار بافت نمی‌شود
- در هر مرحله زمانی، بردار بافت به بخش‌های مختلف دنباله ورودی نگاه می‌کند





## شبکه‌های بازگشتی: سازوکار توجه ...

### ○ سازوکار توجه ...

- فرض: ترجمه متن  $n$  کلمه‌ای  $\mathbf{x}$  در زبان مبدا به متن  $m$  کلمه‌ای  $\mathbf{y}$  در زبان مقصد

$$\mathbf{x} = [x_1, x_2, \dots, x_n]$$

$$\mathbf{y} = [y_1, y_2, \dots, y_m]$$

- رمزگذار: یک BiLSTM با حالت‌های مخفی جلو رو  $\overrightarrow{h}_i$  و عقب رو  $\overleftarrow{h}_i$

$$\mathbf{h}_i = [\overrightarrow{h}_i^\top; \overleftarrow{h}_i^\top]^\top, i = 1, \dots, n$$

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t)$$

- حالت مخفی رمزگشا برای کلمه خروجی  $t=1, \dots, m$

• تابعی از حالت قبلی رمزگشا، کلمه خروجی قبلی و بردار بافت  $\mathbf{c}_t$

• بردار بافت: جمع وزن دار حالت‌های رمزگذار برای همه کلمات ورودی

$$\alpha_{t,i} = \text{align}(y_t, x_i)$$

وزن‌های قابل آموختش با فرایند یادگیری

$$= \frac{\exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_i))}{\sum_{i'=1}^n \exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_{i'}))}$$

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{s}_t; \mathbf{h}_i])$$

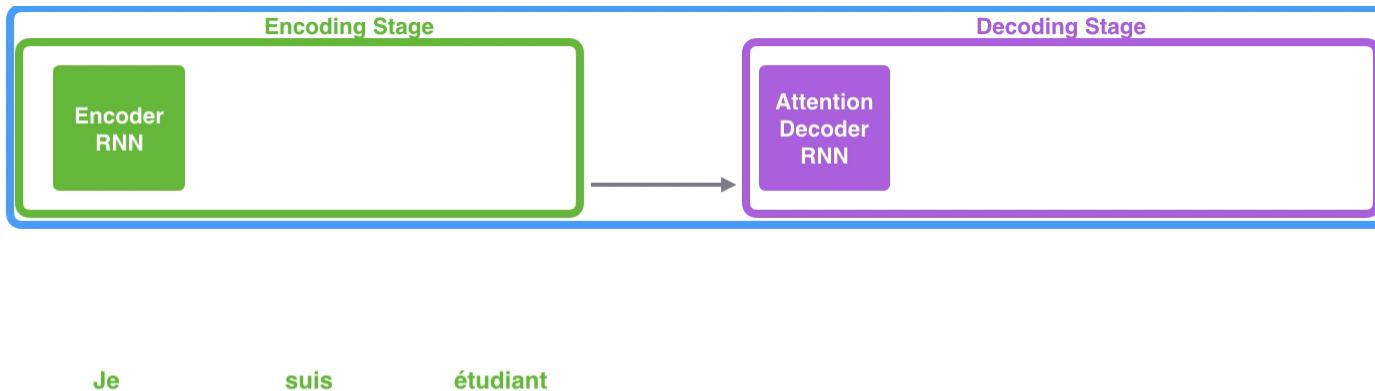
- وزن‌های  $\alpha_{t,i}$  = میزان توجه (تراز بودن) کلمه ورودی  $i$  با کلمه خروجی  $t$



# شبکه‌های بازگشتی: سازوکار توجه ...

## ○ سازوکار توجه ...

- استفاده از بردارهای بافت همه واحدهای ورودی (و نه فقط آخرین بردار)
- مثال ترجمه: ارسال بردارهای بافت همه دنباله ورودی به رمزگشا





## شبکه‌های بازگشتی: ساز و کار توجه ...

### ○ ساز و کار توجه ...

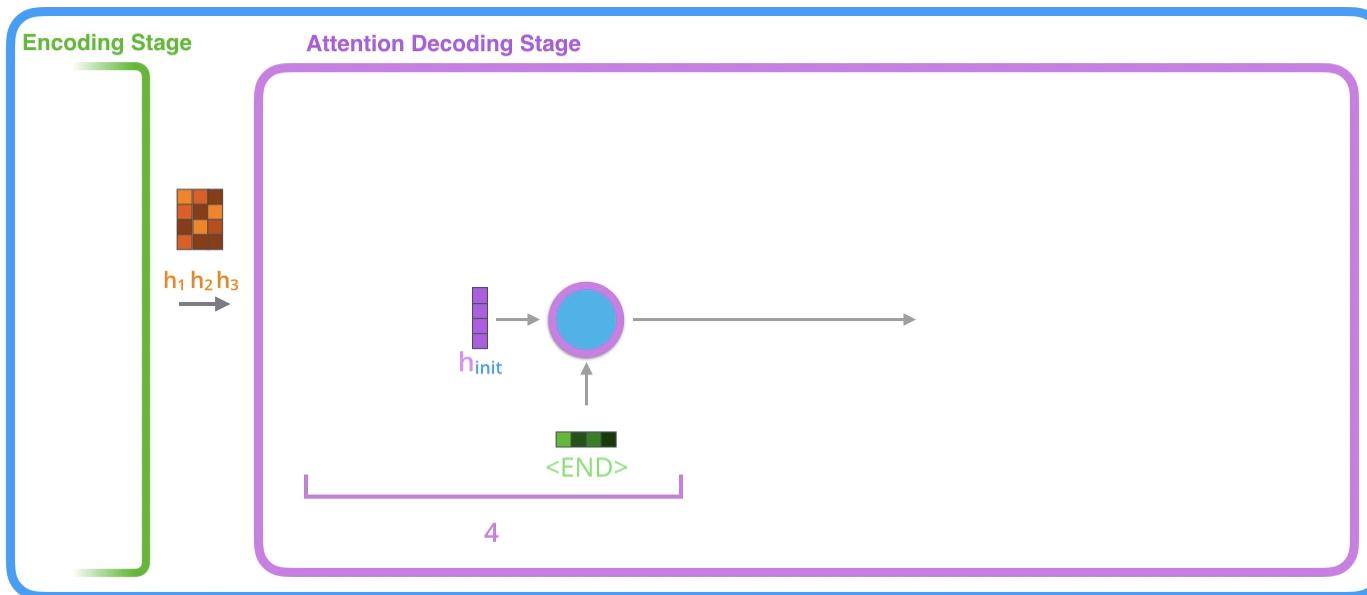
- محاسبه وزن بردارهای بافت رمزگذار برای واحد خروجی بعدی
- محاسبه میانگین وزن دار همه بردارهای بافت رمزگذار و استفاده از آن به عنوان بردار بافت



# شبکه‌های بازگشتی: ساز و کار توجه ...

## ○ ساز و کار توجه ...

- متصل کردن بردار بافت رمزگشا به بردار بافت حاصل از ساز و کار توجه برای محاسبه خروجی

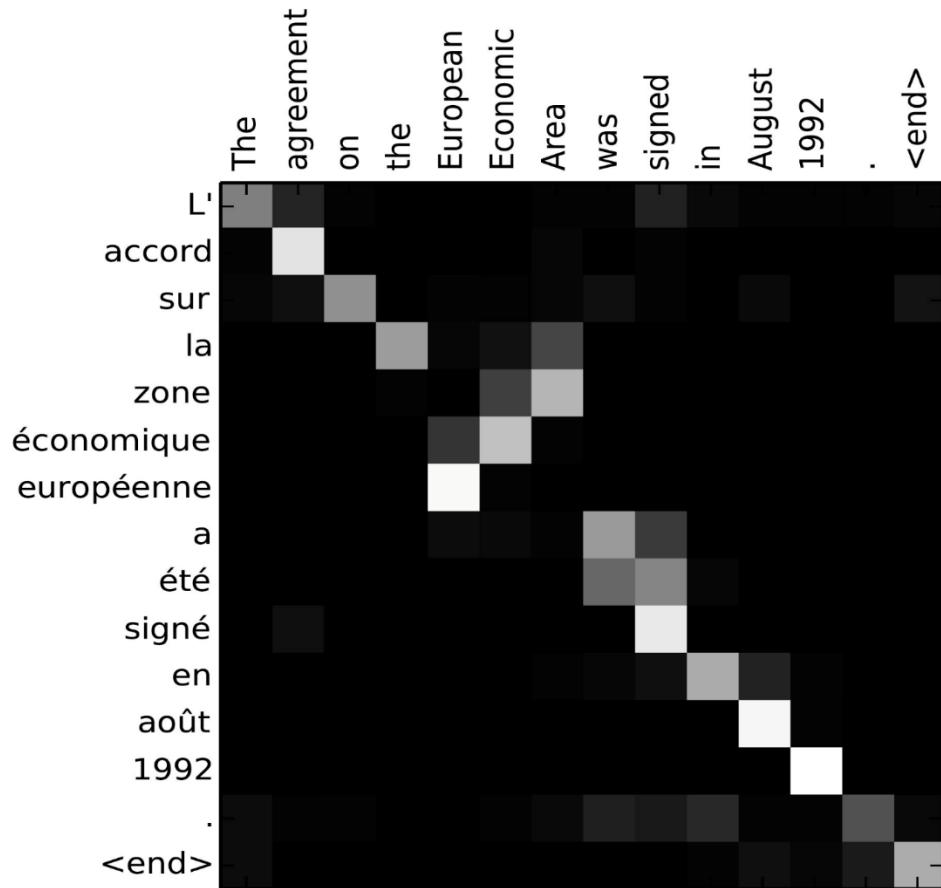




# شبکه‌های بازگشتی: ساز و کار توجه ...

- ساز و کار توجه ...

- مقادیر وزن‌های  $\alpha_{t,i}$





# شبکه‌های بازگشتی: سازوکار توجه ...

## ○ سازوکار توجه ...

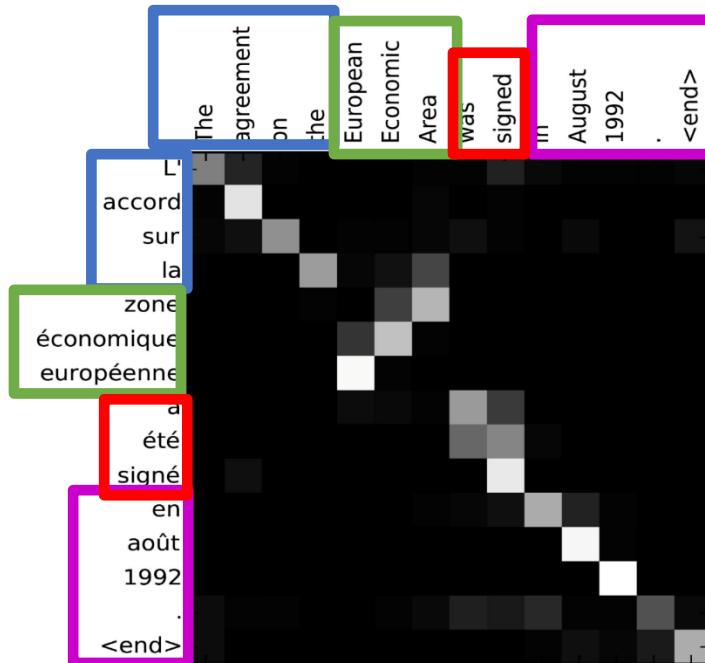
- مثال: ترجمه انگلیسی به فرانسوی

توجه مورب (قطري) به معنai  
مطابقت کلمات به ترتیب است

توجه، ترتیب کلمات  
مختلف را مشخص می‌کند

صرف فعل

توجه مورب (قطري) به معنai  
مطابقت کلمات به ترتیب است



وروادي:

**"The agreement on the European Economic Area was signed in August 1992."**

خروجی:

**"L'accord sur la zone économique européenne a été signé en août 1992."**

# شبکه‌های بازگشتی: ساز و کار توجه ...

C4 مثال: محاسبه

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

The attention mechanism

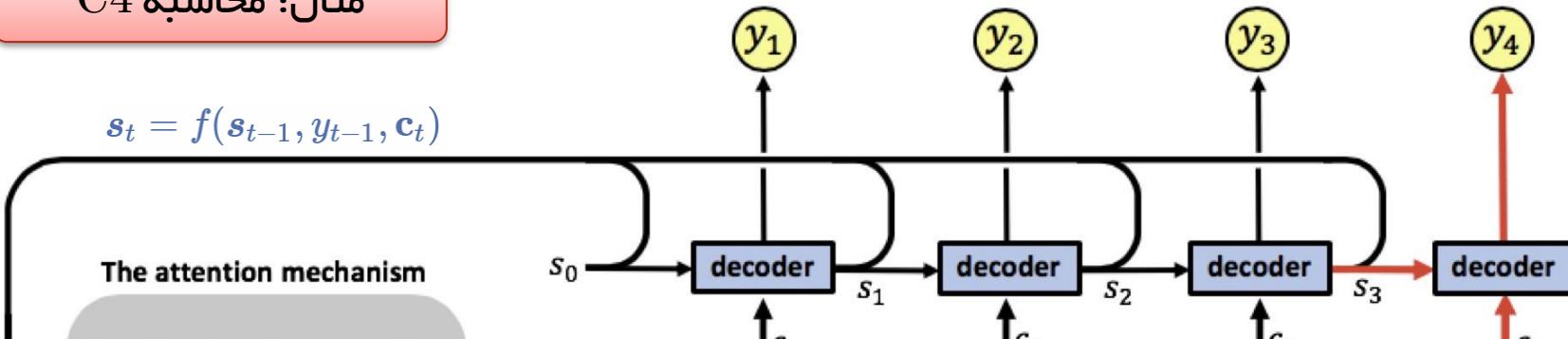
$$\alpha_{41}, \alpha_{42}, \alpha_{43}, \alpha_{44}$$



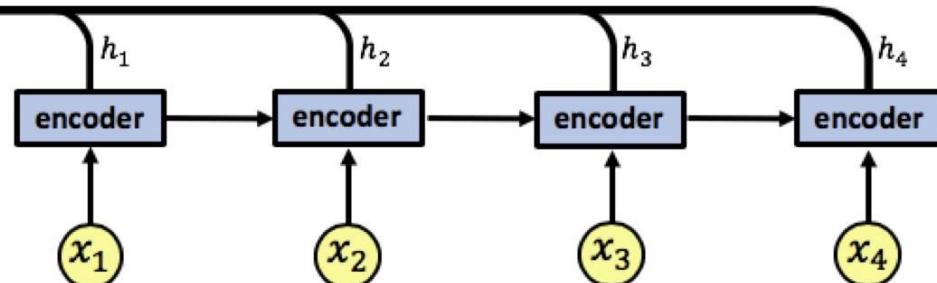
$$c_t = \sum_{i=1}^n \alpha_{t,i} h_i$$

$$\begin{aligned} \alpha_{t,i} &= \text{align}(y_t, x_i) \\ &= \frac{\exp(\text{score}(s_{t-1}, h_i))}{\sum_{i'=1}^n \exp(\text{score}(s_{t-1}, h_{i'}))} \end{aligned}$$

$$\text{score}(s_t, h_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[s_t; h_i])$$



$$s_3 \quad h_1, h_2, h_3, h_4$$



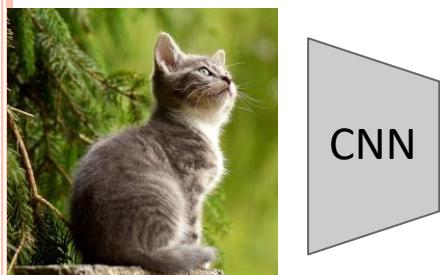


## شبکه‌های بازگشتی: سازوکار توجه ...

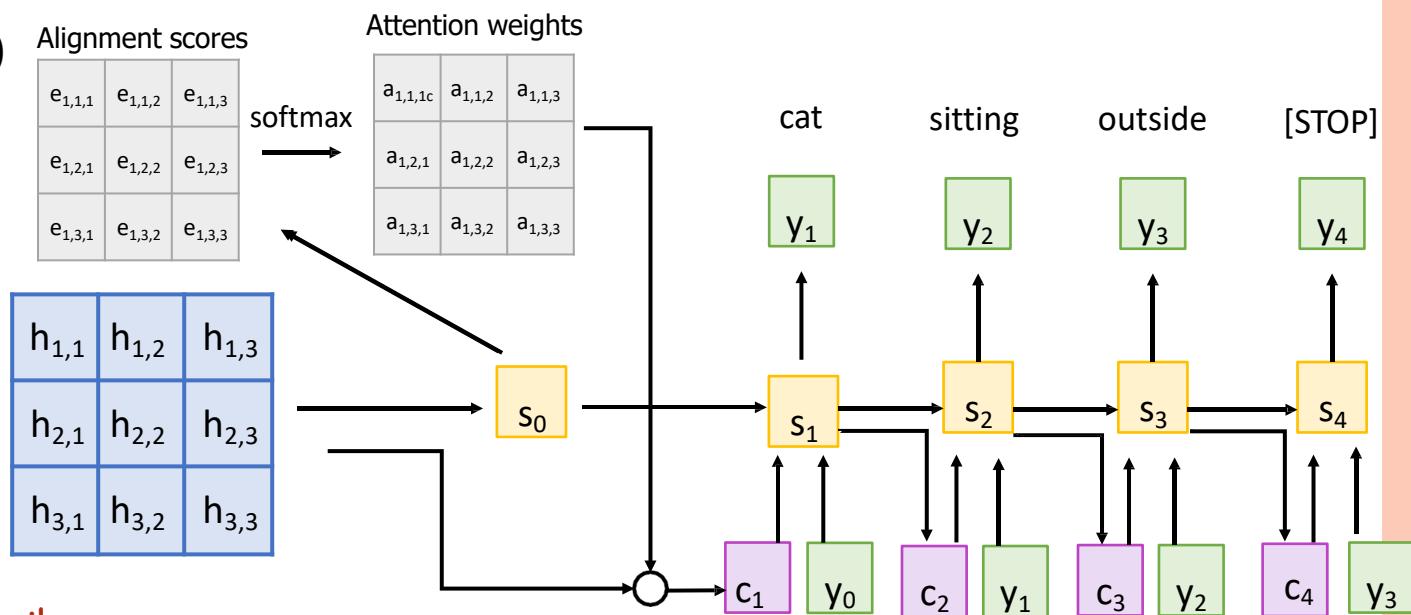
$$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$$

$$a_{t,:} = \text{softmax}(e_{t,:})$$

$$c_t = \sum a_{t,i,j} h_{i,j}$$



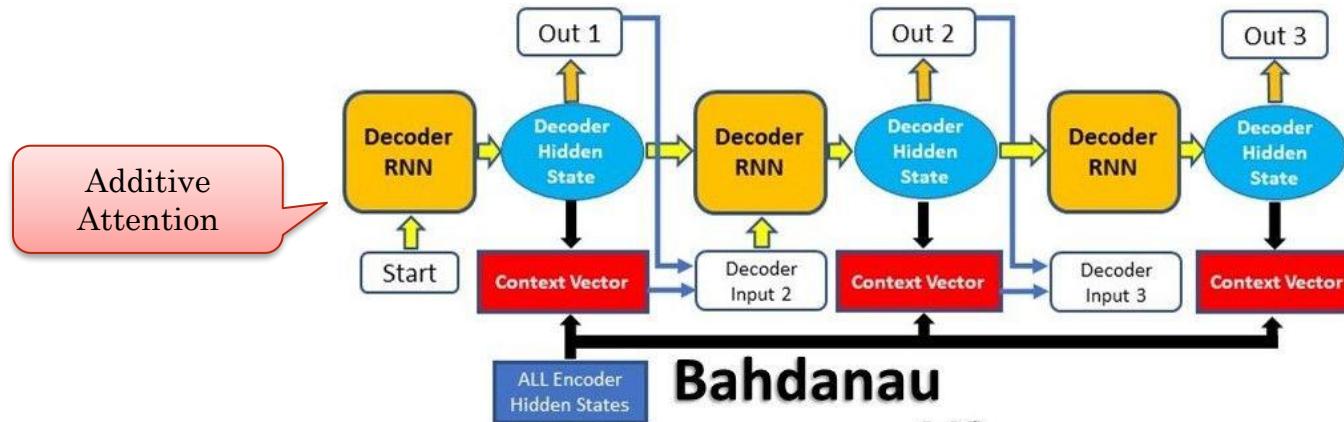
از شبکه عصبی پیچشی برای استخراج ویژگی‌های تصویر استفاده کنید



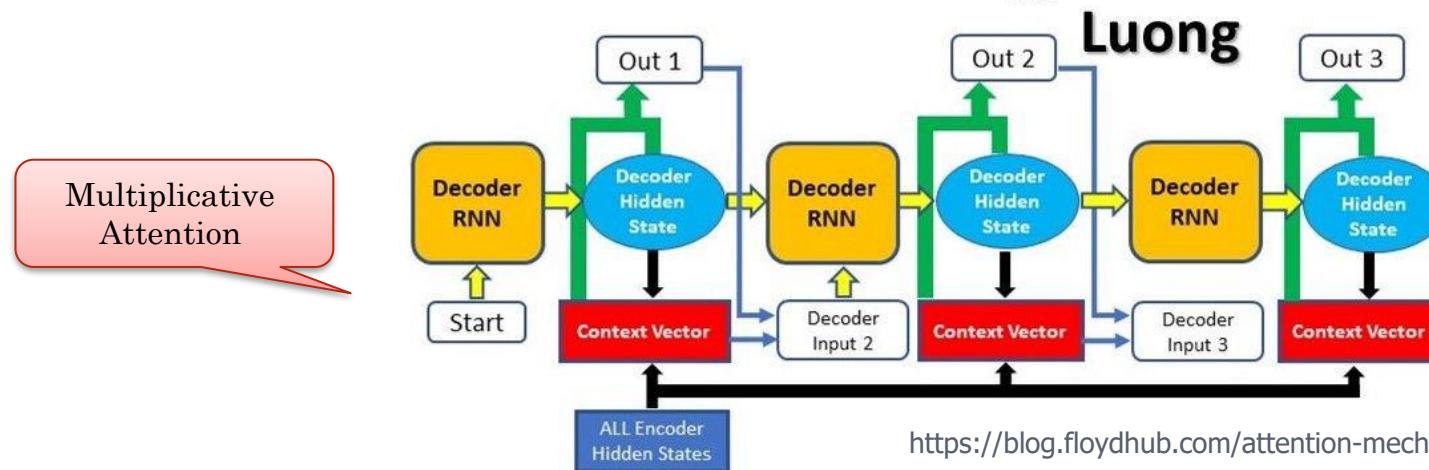
در هر مرحله زمانی رمزگشا از بردار بافت متفاوتی استفاده می‌کند که به بخش‌های مختلف تصویر ورودی نگاه می‌کند

# شبکه‌های بازگشتی: ساز و کار توجه ...

○ انواع توجه ...



VS



<https://blog.floydhub.com/attention-mechanism>



# شبکه‌های بازگشتی: ساز و کار توجه ...

## روش‌های مختلف محاسبه امتیاز توجه

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \text{cosine}[\mathbf{s}_t, \mathbf{h}_i]$	Graves2014
Additive(*)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{s}_t; \mathbf{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \mathbf{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^\top \mathbf{W}_a \mathbf{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^\top \mathbf{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \frac{\mathbf{s}_t^\top \mathbf{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

## شبکه‌های بازگشتی: ساز و کار توجه ...

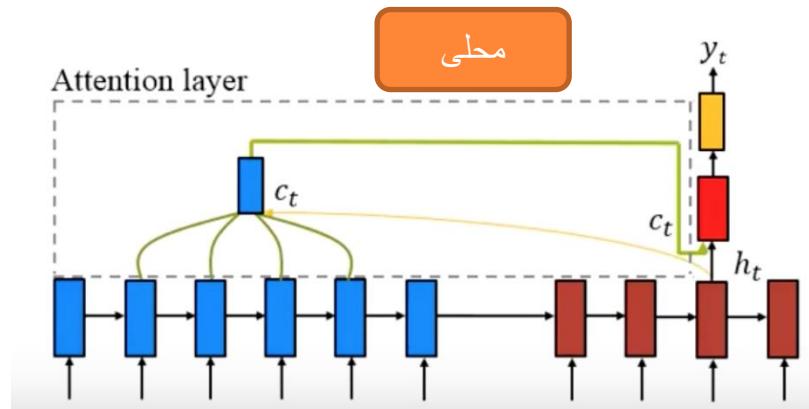
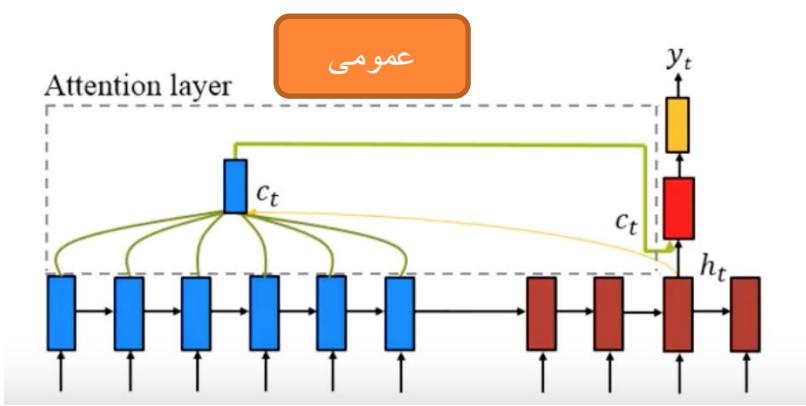
### ○ انواع توجه ...

- نرم (Soft) و سخت (Hard)

- نرم: یادگیری وزن‌ها و محاسبه آنها با در نظر گرفتن کل داده (کل تصویر یا کل جمله)
- سخت: در نظر گرفتن تنها بخشی از داده در هر مرحله

- محلی (Local) و عمومی (Global)

- عمومی = توجه نرم = در نظر گرفتن کل داده در محاسبات وزن‌ها
- محلی: پیش‌بینی فقط یک خروجی برای تراز کردن با ورودی جاری و سپس در نظر گرفتن یک پنجره اطراف این ورودی برای محاسبه بردار بافت



# شبکه‌های بازگشتی: سازوکار توجه ...

## ○ انواع توجه: آشنایی با بردار Key, Query و Value

کلید تا چه اندازه به پرسش شباهت دارد؟

۱- محاسبه  
attention mask  
تا چه اندازه هر کلید  
به پرسش مورد نظر  
شباخت دارد؟

**Google** Query (Q) کتاب هادی ویسی هوش

**صباید** https://sabavid.ir › Video آقای هوش مصنوعی [تعریف هوش مصنوعی دکتر هادی ویسی - صباید

 آقای هوش مصنوعی [تعریف هوش مصنوعی دکتر هادی ویسی - امروزش، پردازش تصویر ایران مجله تحول دیجیتال و هوش مصنوعی ir /https://imageprocessing.ir میلم ...

1:37 April 23, 1402 AP · صباید

**فروشگاه کتاب شاب** http://ketabshop.ir ... ۱۴۹۶ هادی ویسی - خرد آنلاین و اینترنتی کتاب

درآمدی بر دولت محلی مؤلف: دکتر هادی ویسی اداره امور کشور نیازمند سازوکارهای ویژه ای است که یکی از آنها دولت محلی است. دولت محلی واقعیت اکنوناپذیر و مارادی ...

Missing: هوش | Search with: هوش

**آپارات** https://www.aparat.com ... آقای هوش مصنوعی [تعریف هوش مصنوعی دکتر هادی ویسی - آپارات

پردازش تصویر ایران مجله تحول دیجیتال و هوش مصنوعی.ir میلم ... تقدیم زاده بیو-همسگر هوش مصنوعی دانشگاه تهران مجری و برنامه ...

 1:37 04/23/1402 AP · آپارات

Missing: کتاب | Search with: کتاب

**دانشگاه شریف** http://library.sharif.ir › search نتیجه جستجو - شبکه‌های عصبی کامپیوتر

۶,۶ تاب بردها / نویسنده لوران فاست؛ ترجمه هادی ویسی، کبری مخادرخی، سعید بهاری‌شورکی، فلست، لوران، نصی...،

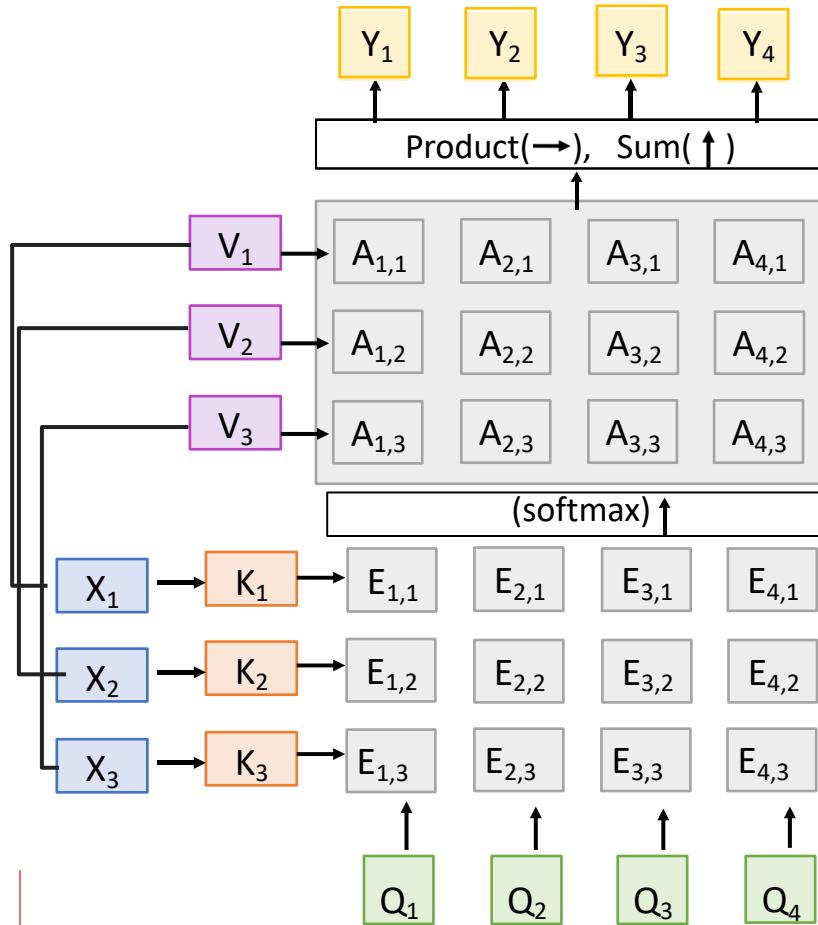
**فیدیبو** https://fidiбо.com ... مترجم > فیدیبو (FidiBo) آقای هوش مصنوعی دکتر هادی ویسی

معرفی کتاب‌های هادی ویسی. مترجم، نویسنده، از هادی ویسی بخوانید. مشاهده مشترک. پردازش گفتوگو زبانگ هوانگ. ۹۵,۰۰۰ نظر. مناطق، مخفات شده، تقدیم، نسخه، ...

## ۲- استخراج مقادیر بر مبنای توجه:

# شبکه‌های بازگشتی: ساز و کار توجه ...

## ● لایه توجه



### Computation:

**Key vectors:**  $K = \mathbf{X}W_K$  (Shape:  $N_x \times D_Q$ )

( $x \times D_v N$  : epahS)  $\mathbf{X}W_V = \mathbf{V}$  : Value Vectors

**Similarities:**  $E = \mathbf{Q}K^T / \sqrt{D_Q}$  (Shape:  $N_Q \times N_x$ ),

$$E_{i,j} = (\mathbf{Q}_i \cdot \mathbf{K}_j) / \sqrt{D_Q}$$

**Attention weights:**  $A = \text{softmax}(E, \text{dim}=1)$  (Shape:  $N_Q \times N_x$ )

**Output vectors:**  $Y = A\mathbf{V}$  (Shape:  $N_Q \times D_V$ )  $Y_i = \sum_j A_{i,j} \mathbf{V}_j$

### Inputs:

**Query vectors:**  $\mathbf{Q}$  (Shape:  $N_Q \times D_Q$ )

**Input vectors:**  $\mathbf{X}$  (Shape:  $N_x \times D_x$ )

**Key matrix:**  $\mathbf{W}_K$  (Shape:  $D_x \times D_Q$ )

**Value matrix:**  $\mathbf{W}_V$  (Shape:  $D_x \times D_V$ )

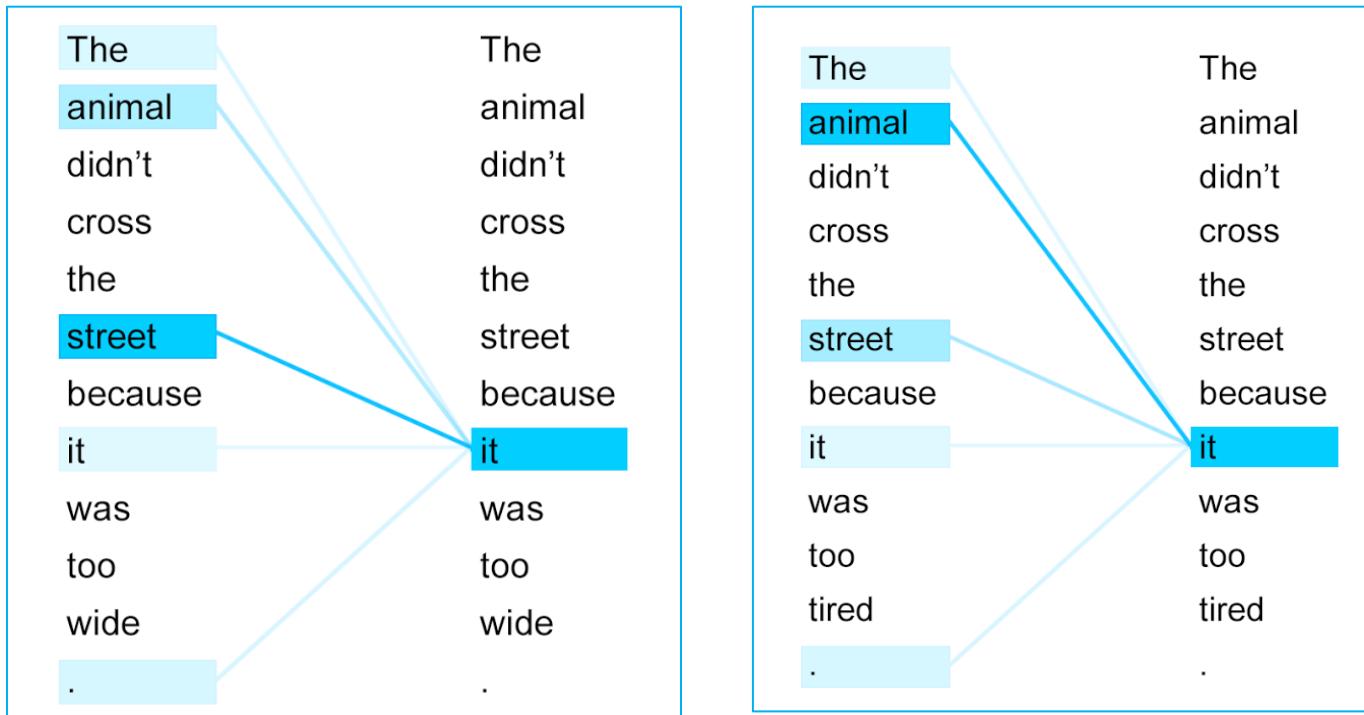


# شبکه‌های بازگشتی: ساز و کار توجه ...

## ○ انواع توجه ...

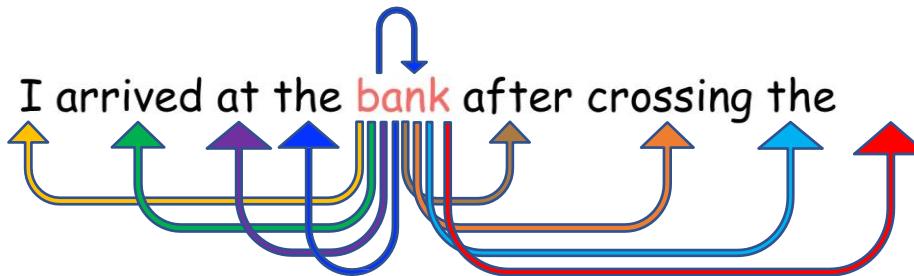
### • توجه به خود (Self-Attention)

- بیانگر میزان توجه مابین واحدهای مختلف یک دنباله (مانند کلمات یک جمله)
- یاد گرفتن اینکه it معادل کدام کلمه است (مرجع ضمیر)

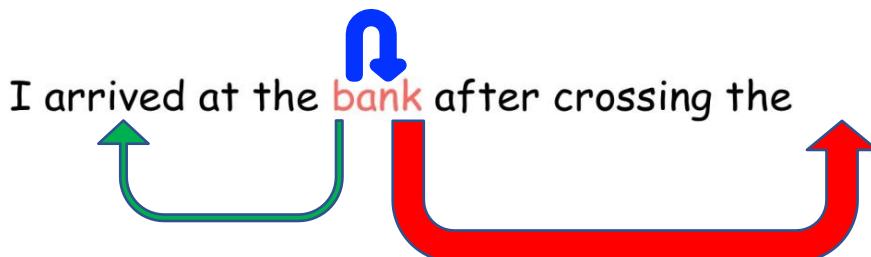


## شبکه‌های بازگشتی: ساز و کار توجه ...

- انواع توجه: توجه به خود (Self-Attention)
- نمایش جدید هر کلمه در دنباله نشان‌دهنده‌ی رابطه‌ی آن کلمه با همه‌ی کلمات می‌باشد:



- ضخامت پیکان نشان‌دهنده وزن توجه است:

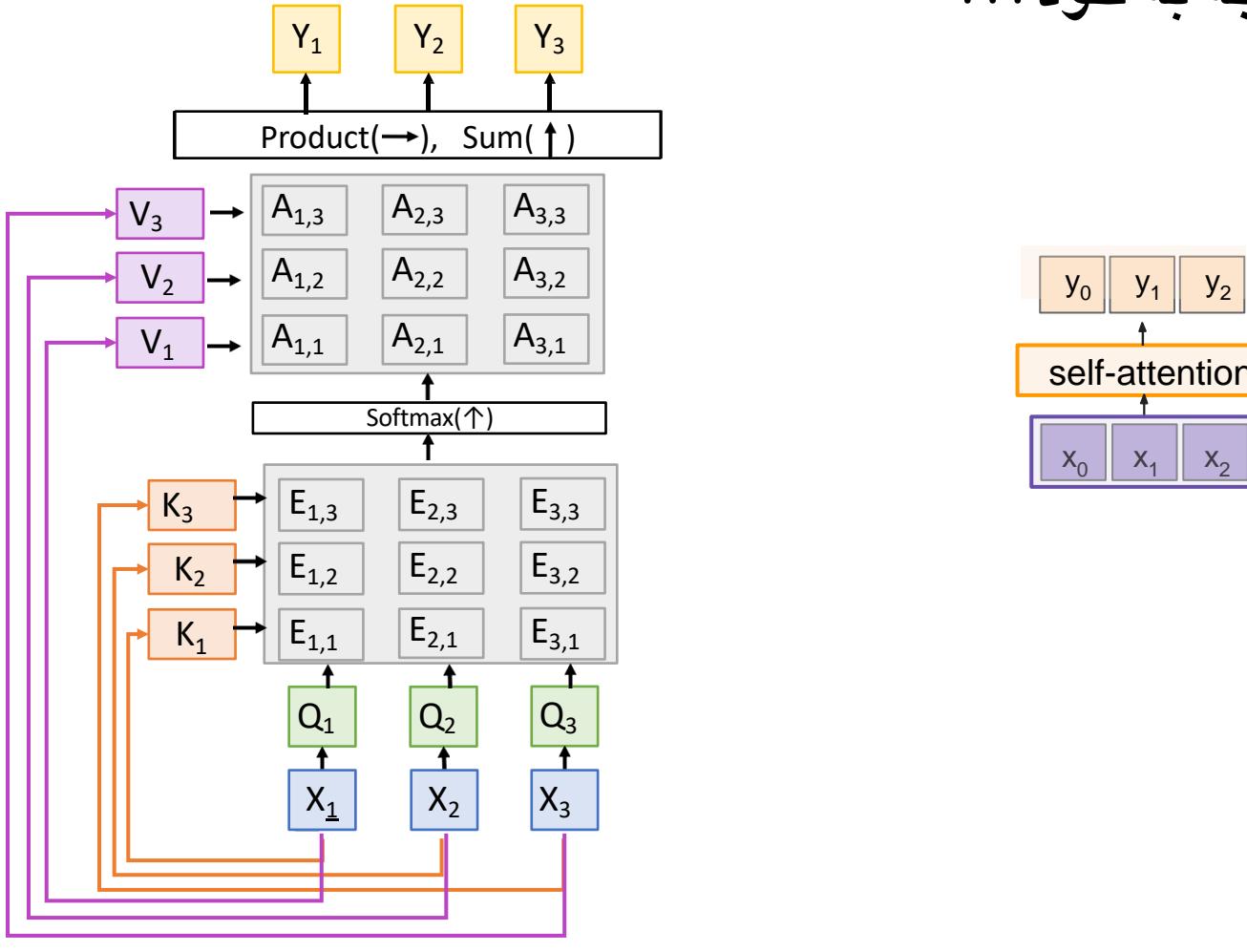


وزن توجه زیاد، به این معنی است که کلمه دیگر، به شدت نشان‌دهنده نمایش جدیدی از کلمه است



## شبکه‌های بازگشتی: سازوکار توجه ...

○ توجه به خود ...





## شبکه‌های بازگشتی: ساز و کار توجه ...

### توجه به خود: مثال ...

- برای عبارت Action gets results
- محاسبه برای هر کلمه نسبت به سایر کلمات
- برای کلمه اول (Action)
- بردار کلمه = کلمه Query (در ترجمه Seq2Seq، خروجی قبلی رمزگشایی (st))
- بردارهای Keys: همه کلمات (در ترجمه Seq2Seq، بردارهای حالت مخفی رمزگذار، h<sub>i</sub>)
- بردارهای Value: همه کلمات (در ترجمه Seq2Seq، بردارهای حالت مخفی رمزگذار، v<sub>i</sub>)

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}}\right)\mathbf{V}$$

n= 64 میزان توجه

Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum	
Action	q <sub>1</sub>	k <sub>1</sub>	v <sub>1</sub>	q <sub>1</sub> · k <sub>1</sub>	q <sub>1</sub> · k <sub>1</sub> / 8	x <sub>11</sub>	x <sub>11</sub> * v <sub>1</sub> → z <sub>1</sub>		خروجی Self-Attention
gets		k <sub>2</sub>	v <sub>2</sub>	q <sub>1</sub> · k <sub>2</sub>	q <sub>1</sub> · k <sub>2</sub> / 8	x <sub>12</sub>	x <sub>12</sub> * v <sub>2</sub>		
results		k <sub>3</sub>	v <sub>3</sub>	q <sub>1</sub> · k <sub>3</sub>	q <sub>1</sub> · k <sub>3</sub> / 8	x <sub>13</sub>	x <sub>13</sub> * v <sub>3</sub>		

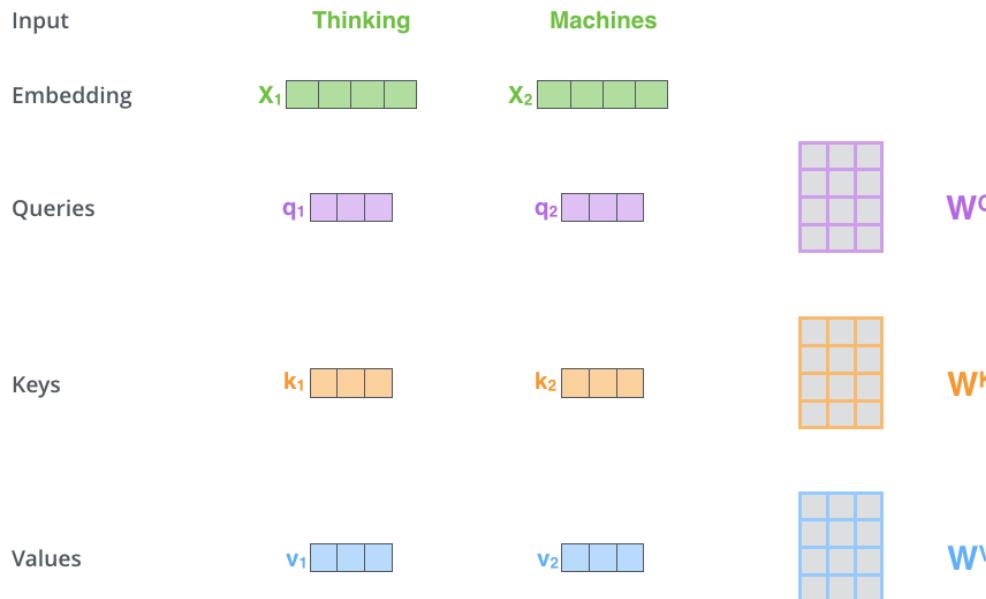
محاسبه به صورت مشابه برای بقیه کلمات



# شبکه‌های بازگشتی: ساز و کار توجه ...

## ○ توجه به خود: ماتریس‌های تبدیل ...

- محاسبه  $Value$ ,  $Query$  و  $Key$  برای هر کلمه با ضرب بردارهای تعییه کلمات در ماتریس‌های متناظر  $Query$ ,  $Value$  و  $Key$
- محاسبه این ماتریس‌ها در فرایند آموزش (لایه خطی)
- معمولاً منجر به کاهش بعد می‌شوند (مثلاً از ۵۱۲ به ۶۴)

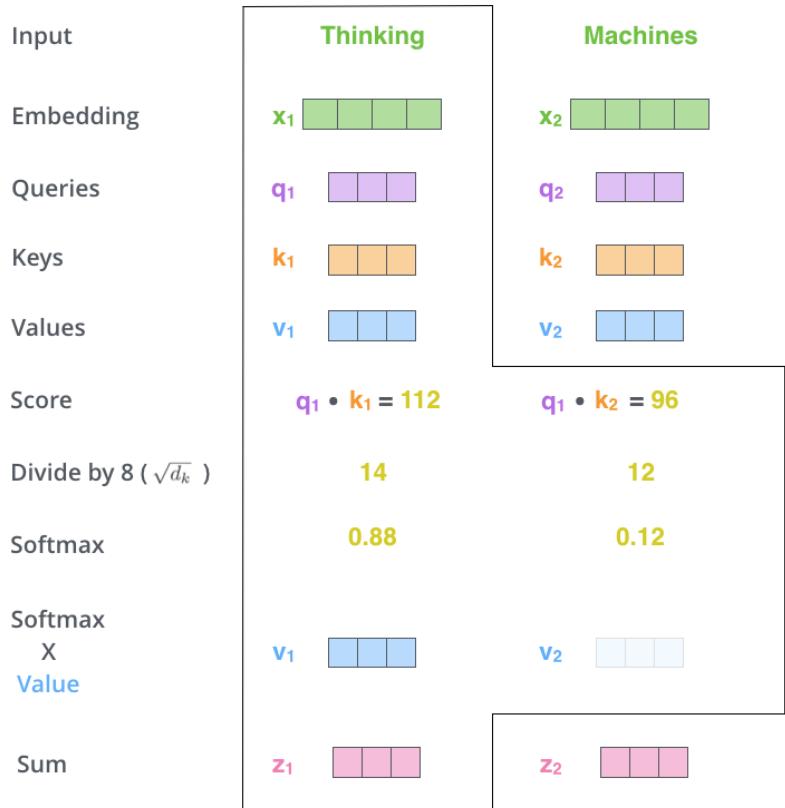




# شبکه‌های بازگشتی: ساز و کار توجه ...

## توجه به خود: مرافق ...

- محاسبه Key, Query و Value برای هر کلمه
- ضرب داخلی بردارهای Key و Query
- نرمال کردن مقدار و تبدیل به احتمال (وزن)
- ضرب وزن‌ها در بردارهای Value
- جمع کردن بردارهای وزن‌دار Value
- اتصال بردار همه هسته‌ها و عبور از لایه خطی

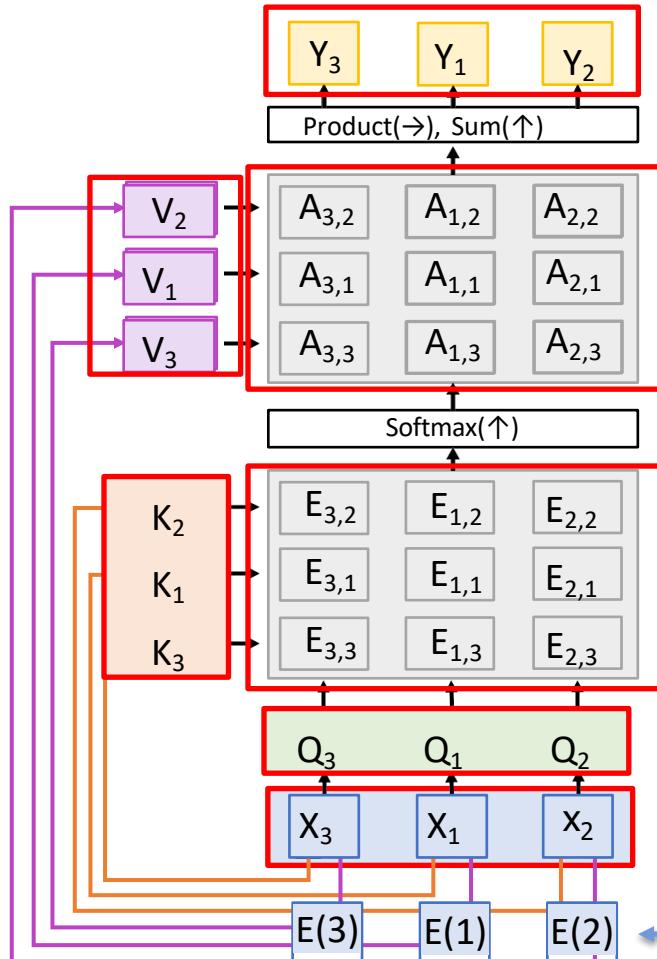


وزن‌ها:  $W^O, W^V, W^K, W^Q$

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

# شبکه‌های بازگشتی: سازوکار توجه ...

## ○ توجه به خود



- فرض کنید ترتیب ورودی را عوض کنیم
  - بردارهای پرسش‌ها و کلیدها یکسان هستند، اما ترتیب‌شان تغییر می‌کند.
  - ماتریس شباهت یکسان خواهد بود، اما ترتیب‌شان تغییر می‌کند
  - ماتریس وزن‌های توجه یکسان خواهد بود اما ترتیب‌شان تغییر یافته است.
  - بردار خروجی یکسان خواهد بود اما ترتیب‌شان تغییر یافته است

لایه  $f$  لایه توجه به خود معادل جایگشت است

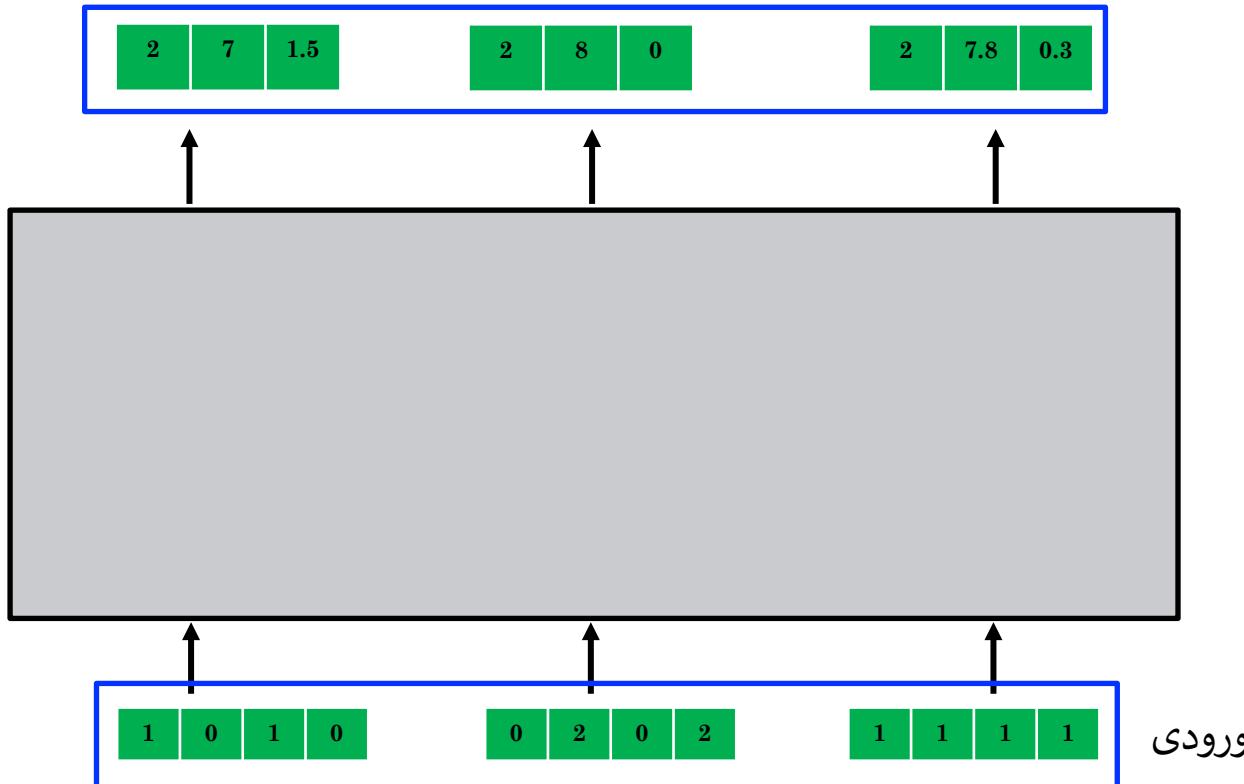
لایه توجه به خود ترتیب بردارهای را که پردازش می‌کند، «نمی‌داند»!

به منظور آگاهی از موقعیت پردازش، یک بردار به بردار هر کلمه که بیانگر موقعیت کلمه در جمله (یا فاصله بین کلمات از هم) است، اضافه می‌کنند.

## شبکه‌های بازگشتی: ساز و کار توجه ...

### توجه به خود: مفهوم

- نمایش جدید هر بردار ورودی که نشان‌دهندهٔ رابطهٔ هر بردار ورودی با تمامی بردارهاست

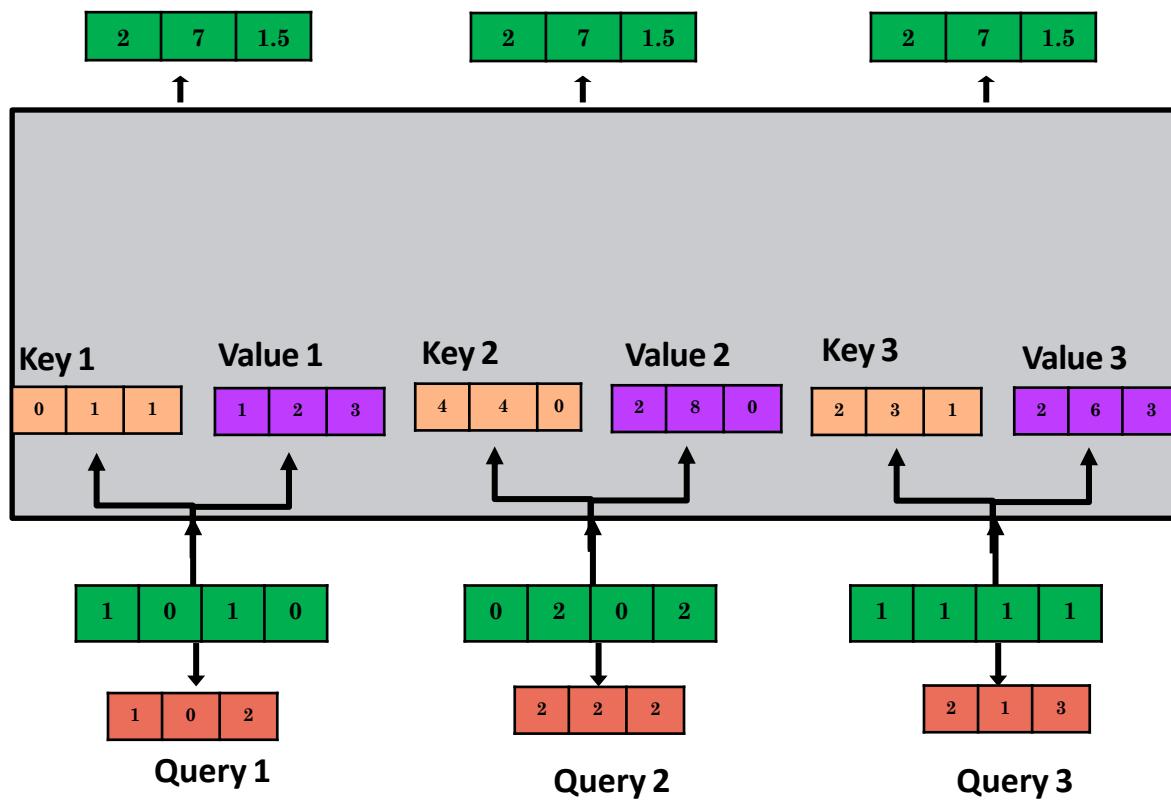




## شبکه‌های بازگشتی: ساز و کار توجه ...

### توجه به خود: مثال ...

- برای هر ورودی ۴ بعدی، سه بردار جدید ۳ بعدی با ضرب در سه ماتریس (که در طول آموزش یاد گرفته می‌شوند)، بدست می‌آید (نمایش جدید ۳ بعدی برای هر ورودی)



e.g., key weights

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

e.g., value weights

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 3 & 0 \\ 1 & 0 & 3 \\ 1 & 1 & 0 \end{bmatrix}$$

e.g., query weights

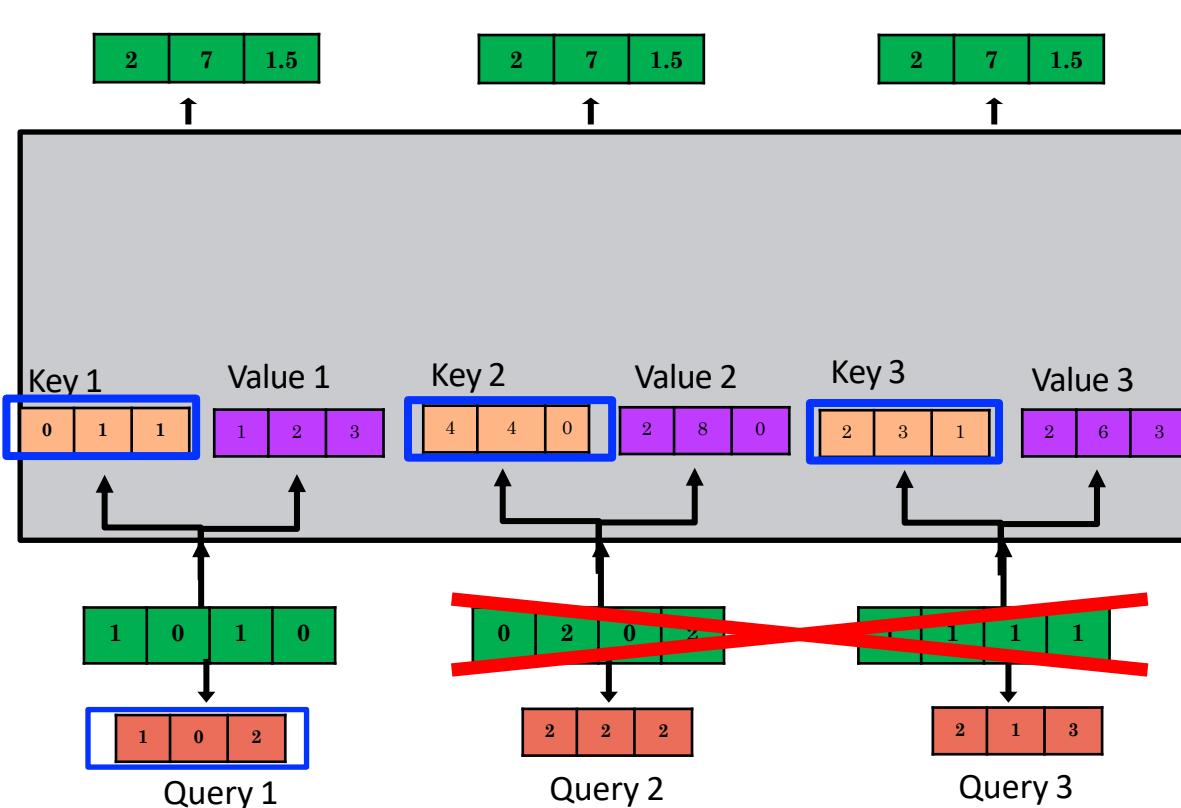
$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

# شبکه های بازگشتی: سازوکار توجه . . .

توجہ به خود: مثال ...

- #### • نحوه یافتن نمایش جدید برای بردار ورودی اول

## ۵ محاسبهٔ ضرب نقطه‌ای بردار پرسش با تمامی بردارهای کلید



	$\times$		$=$	
	$\times$		$=$	
	$\times$		$=$	

## چرا ضرب نقطه‌ای دو بردار شباخت دو بردار را نمایش می‌دهد؟

$x_1 \cdot x_2 = |x_1| |x_2| \cos A^\circ$

$|x_1| = \text{size / magnitude of the vector } x_1$

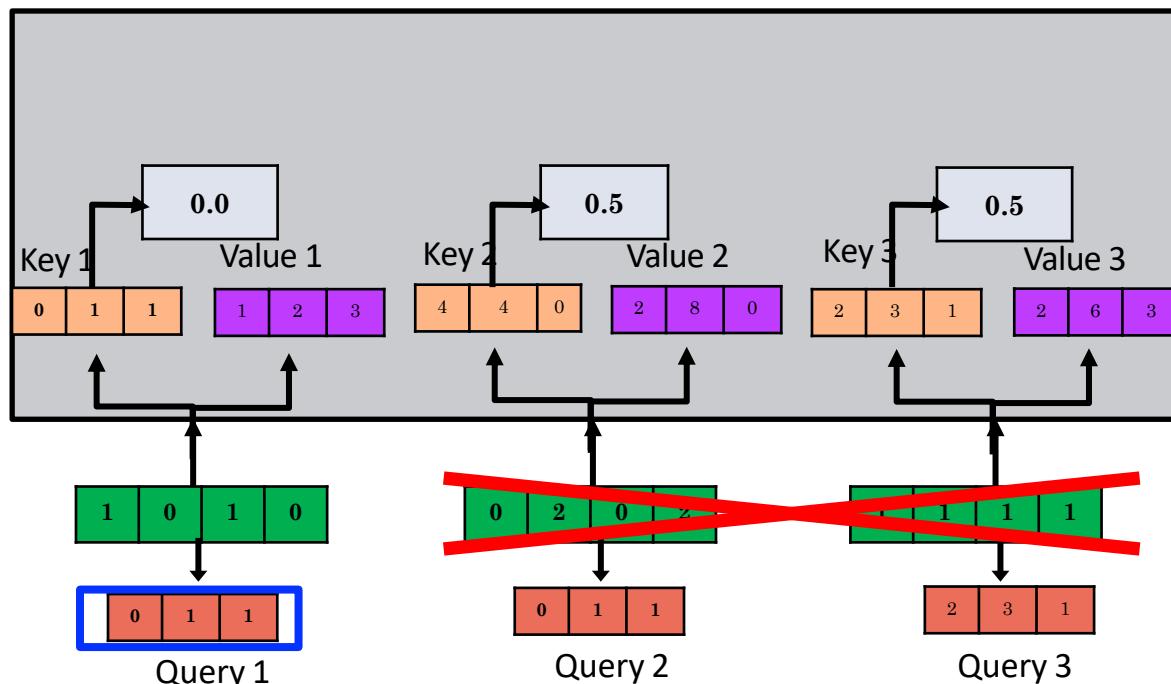
## شبکه‌های بازگشتی: ساز و کار توجه ...

- توجه به خود: مثال ...

- محاسبه وزن‌های توجه

○ محاسبه  $\text{Softmax}([2,4,4]) = [0.0, 0.5, 0.5]$

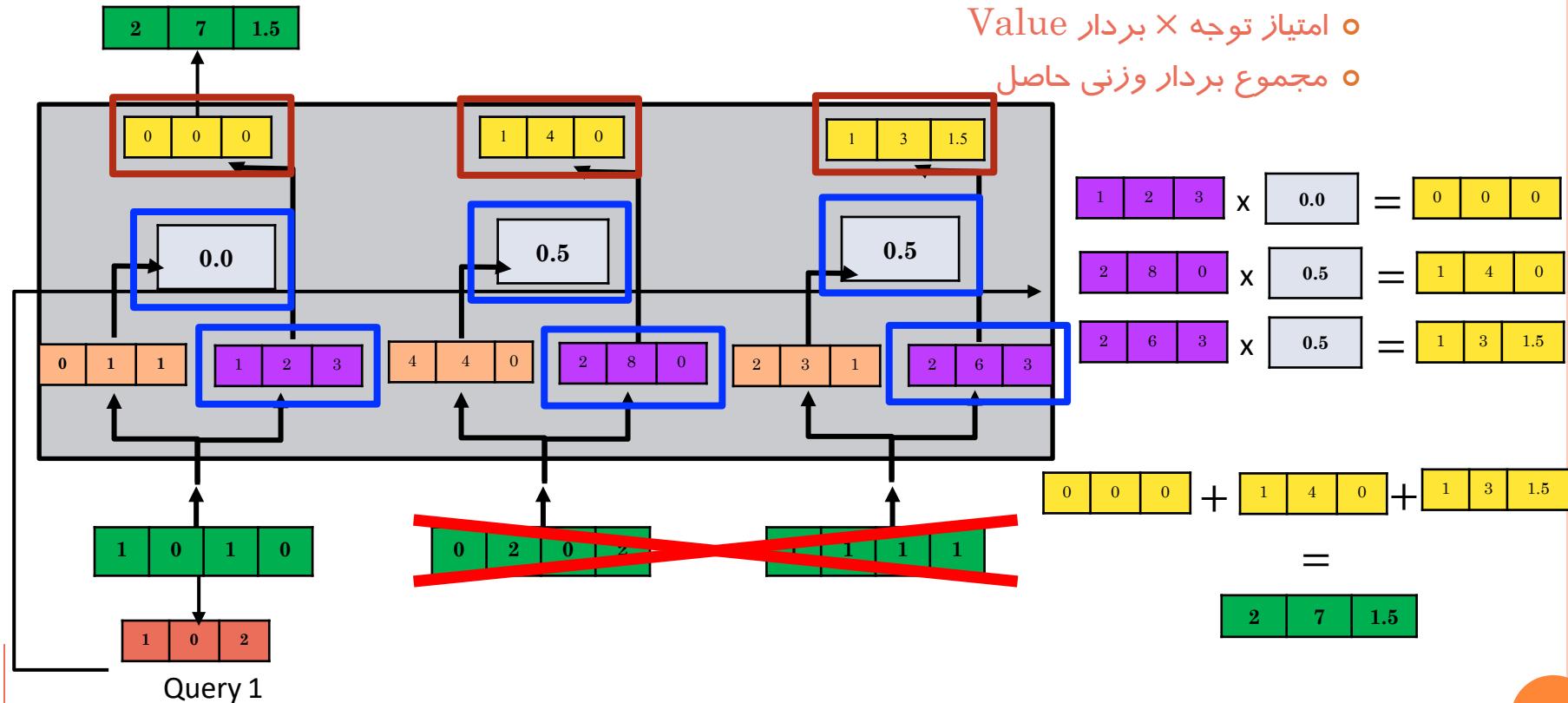
○ ورودی اول به کدام ورودی‌ها بیشتر مرتبط است؟



# شبکه‌های بازگشتی: ساز و کار توجه ...

## ○ توجه به خود: مثال

- محاسبه نمایش جدیدی از ورودی اول به طوری که نشان‌دهنده رابطه‌ی آن با تمام ورودی‌ها نیز می‌باشد.

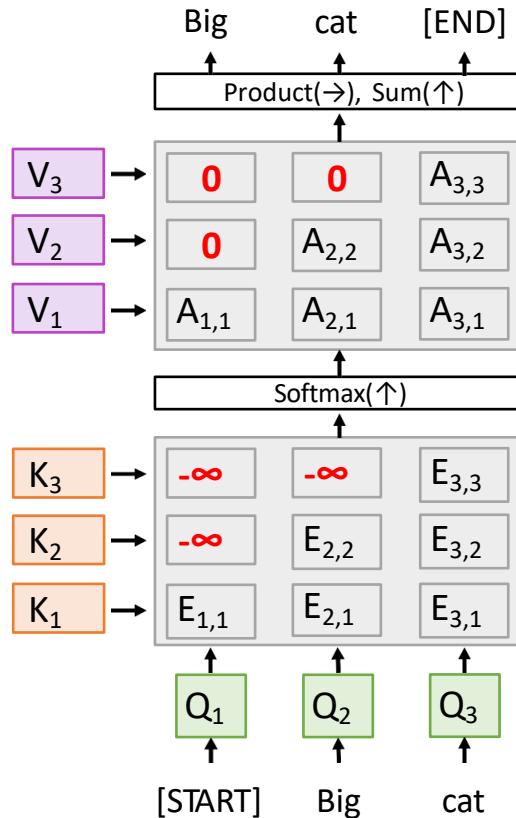


# شبکه‌های بازگشتی: ساز و کار توجه ...

## ○ انواع توجه

- اجازه ندهید که بردارها در دنباله «به جلو نگاه کنند».

- برای مدل سازی زبان استفاده می‌شود (کلمه بعدی را پیش‌بینی کند)

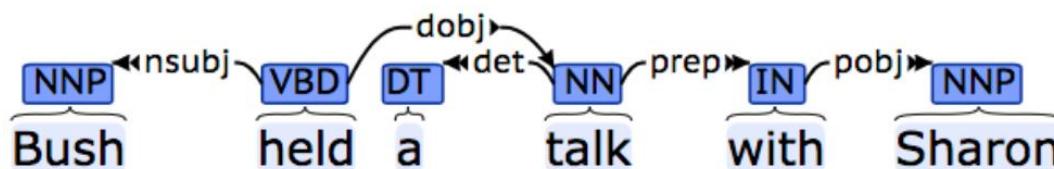
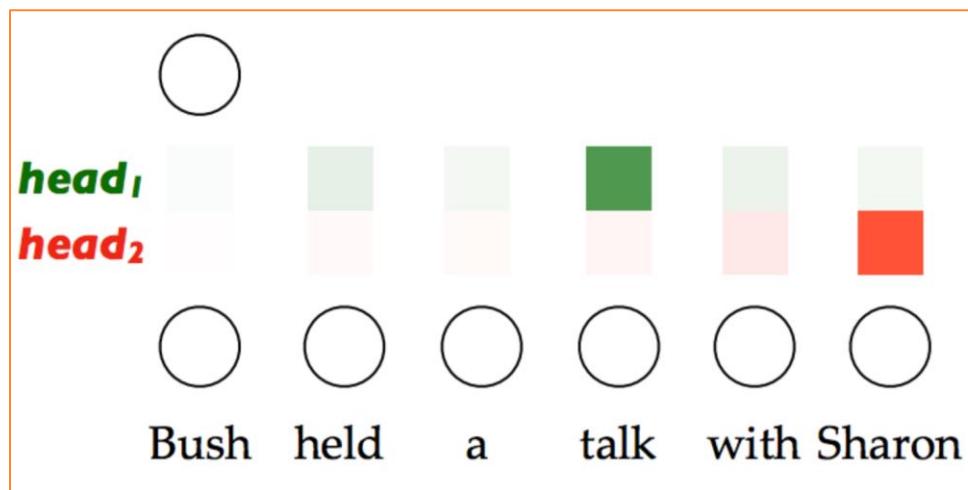


# شبکه‌های بازگشتی: ساز و کار توجه ...

## انواع توجه ...

### • توجه چند هسته‌ای (Multi-Head Attention)

- حالت توسعه یافته Self-Attention که در آن چند لایه توجه نرم به صورت موازی اجرا شده و نتایج آنها با هم ادغام می‌شود



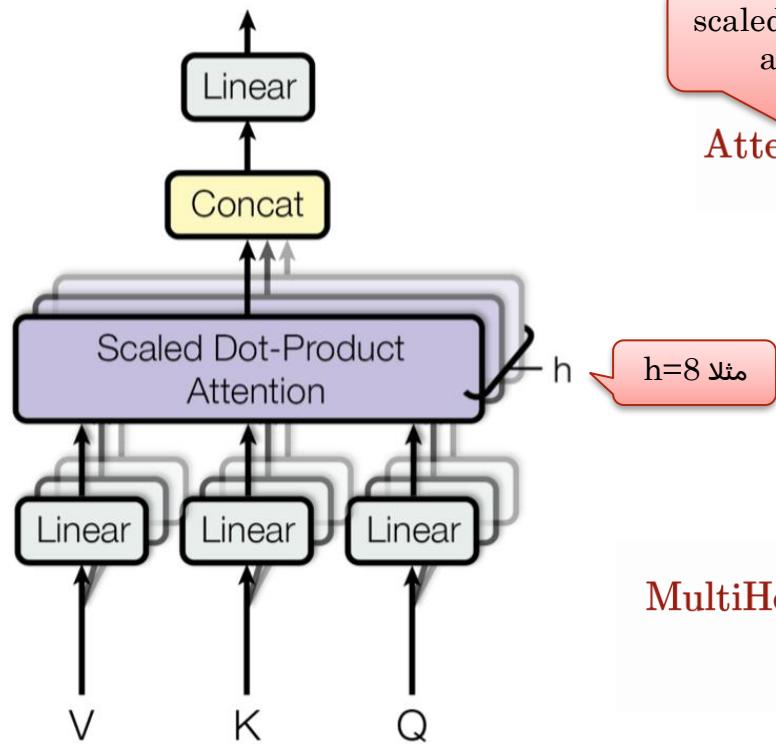


# شبکه‌های بازگشتی: ساز و کار توجه ...

## ○ انواع توجه ...

### • توجه چند هسته‌ای (Multi-Head Attention)

○ اجرای چند چند هسته‌ای Self-Attention به صورت موازی: وزن‌ها (Subspace) متفاوت



scaled dot-product  
attention

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}}\right)\mathbf{V}$$

ابعاد Key

○ ایده: ensembling

لایه وزنی خطی: تنظیم از طریق یادگیری

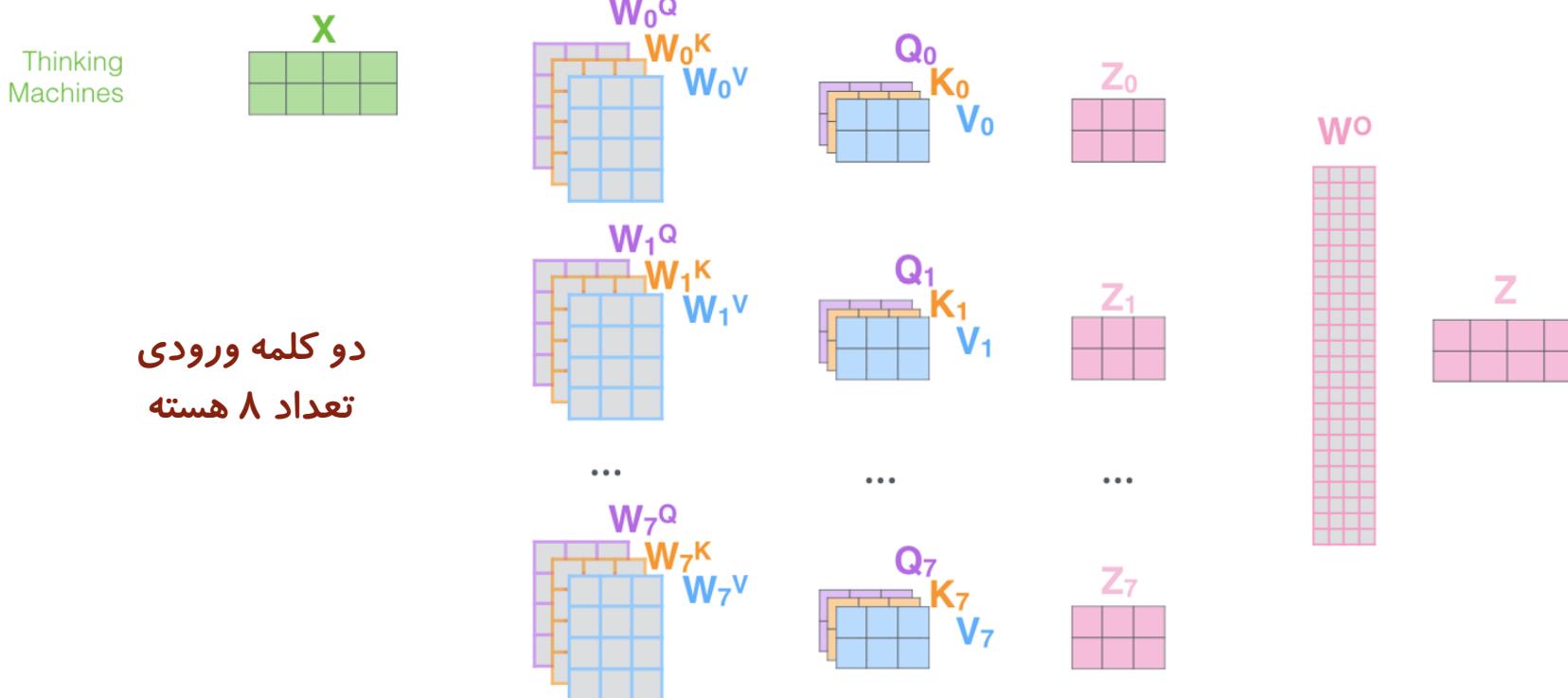
$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= [\text{head}_1; \dots; \text{head}_h] \mathbf{W}^O \\ \text{where } \text{head}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \end{aligned}$$



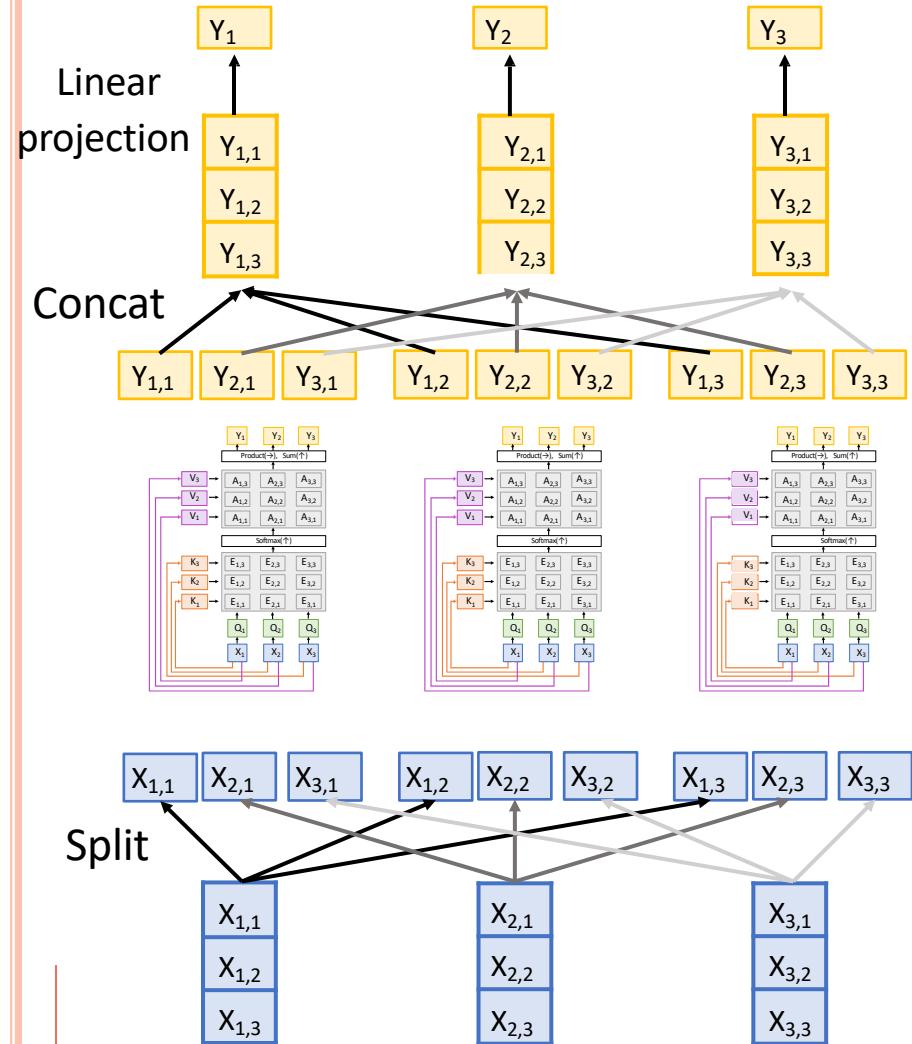
# شبکه‌های بازگشتی: ساز و کار توجه

## توجه چند‌هسته‌ای (Multi-Head Attention): مراحل

- 1) This is our input sentence\*  $X$
- 2) We embed each word\*
- 3) Split into 8 heads. We multiply  $X$  or  $R$  with weight matrices
- 4) Calculate attention using the resulting  $Q/K/V$  matrices
- 5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^O$  to produce the output of the layer



# شبکه‌های بازگشتی: ساز و کار توجه ...



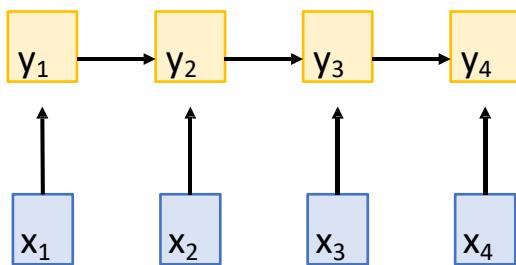
## توجه چند هسته‌ای

- توجه به خود به طور موازی برای هر مجموعه از بردارها اجرا می‌شود.

• به ازای هر head وزن‌ها متفاوت اند

# شبکه های بازگشتی: ساز و کار توجه ...

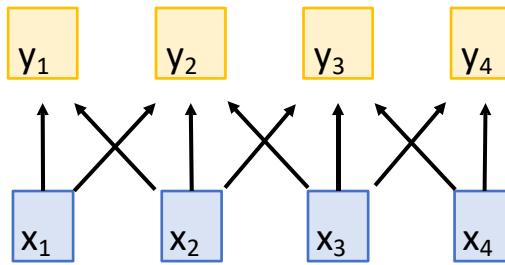
## ○ مقایسه روش های پردازش داده های متولی



بر روی توالی های کار می کند

(+) مناسب بر روی توالی های بلند: بعد از یک لایه  $RNN$ ,  $h_T$  تمامی توالی های قبلی را "به خاطر می اورد"

(-) قابل موازی سازی نیست: نیاز به محاسبه حالت های پنهان به صورت متولی است



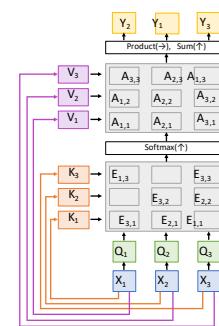
روی شبکه های چند بعدی کار می کند

Works on Multidimensional Grids

(-) نامناسب در دنباله های طولانی: نیاز به انباشت بسیاری از لایه های  $conv$  برای خروجی ها برای "دیدن" کل دنباله دارد

(+) بسیار مناسب برای موازی سازی: هر خروجی می تواند به صورت موازی محاسبه شود

توجه به خود  
Self-Attention

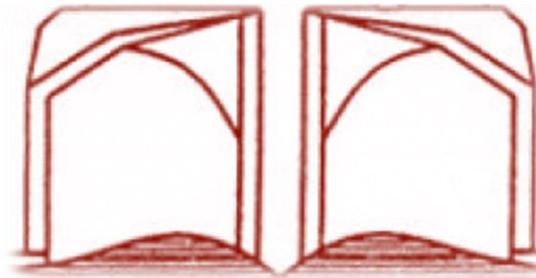


روی مجموعه بردارها کار می کند

(-) مناسب در توالی های طولانی: بعد از یک لایه توجه به خود، هر خروجی همه ورودی ها را "می بیند"!

(+) بسیار مناسب برای موازی سازی: هر خروجی می تواند به صورت موازی محاسبه شود

(-) نیاز به حافظه زیادی دارد



# شبکه‌های عصبی مصنوعی

# مبدل‌ها (Transformers)

## Attention Is All You Need

Ashish Vaswani\*  
Google Brain  
[avaswani@google.com](mailto:avaswani@google.com)

Noam Shazeer\*  
Google Brain  
[noam@google.com](mailto:noam@google.com)

Niki Parmar\*  
Google Research  
[nikip@google.com](mailto:nikip@google.com)

Jakob Uszkoreit\*  
Google Research  
[usz@google.com](mailto:usz@google.com)

Llion Jones\*  
Google Research  
[llion@google.com](mailto:llion@google.com)

Aidan N. Gomez\* †  
University of Toronto  
[aidan@cs.toronto.edu](mailto:aidan@cs.toronto.edu)

Lukasz Kaiser\*  
Google Brain  
[lukaszkaiser@google.com](mailto:lukaszkaiser@google.com)

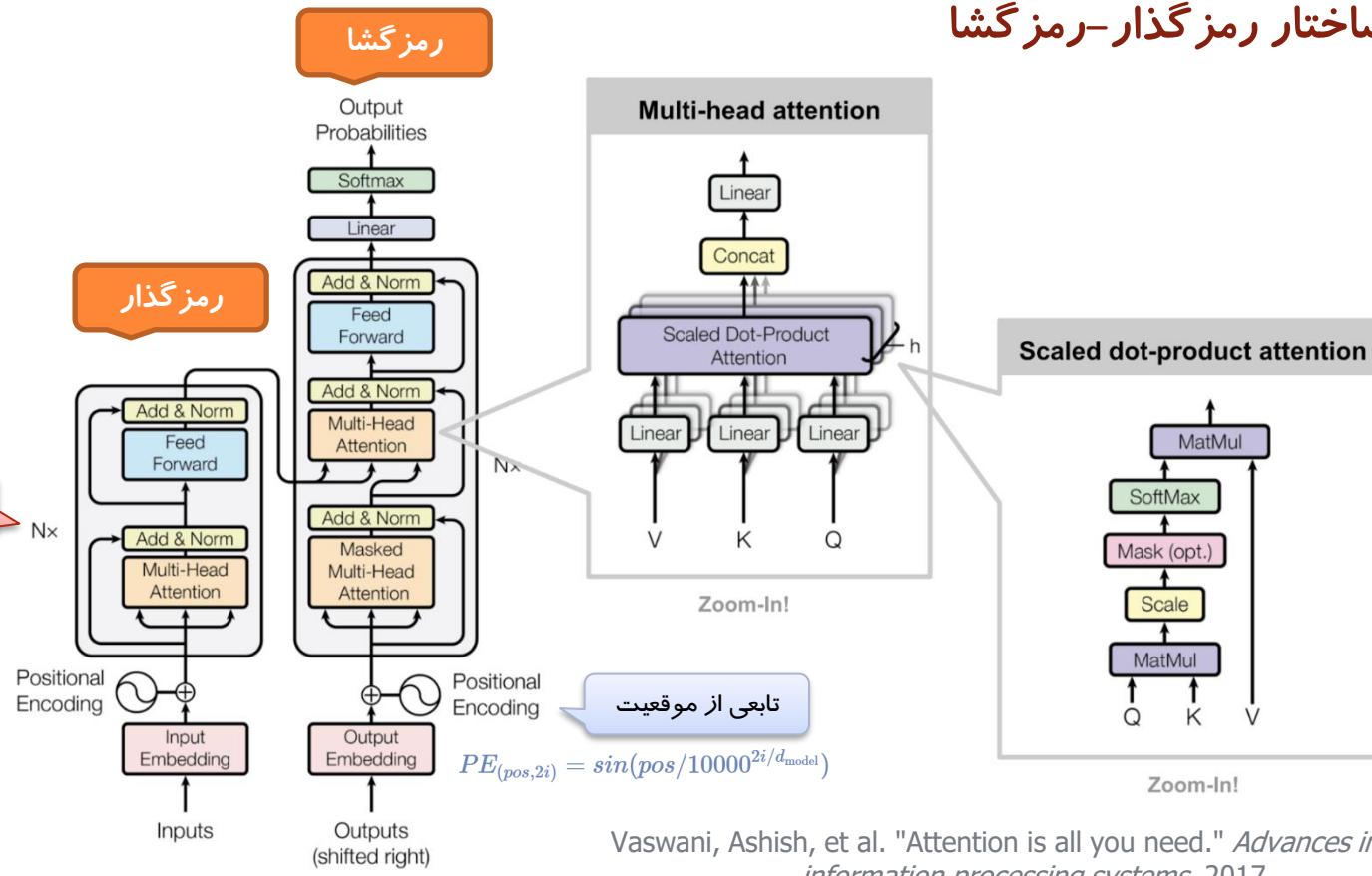
Illia Polosukhin\* ‡  
[illia.polosukhin@gmail.com](mailto:illia.polosukhin@gmail.com)



## مبدل‌ها: معرفی ...

### ... مبدل (Transformer)

- تبدیل ورودی به خروجی و مدل کردن وابستگی بین آنها با ساز و کار توجه (بدون RNN)
- ساختار رمزگذار-رمزگشا



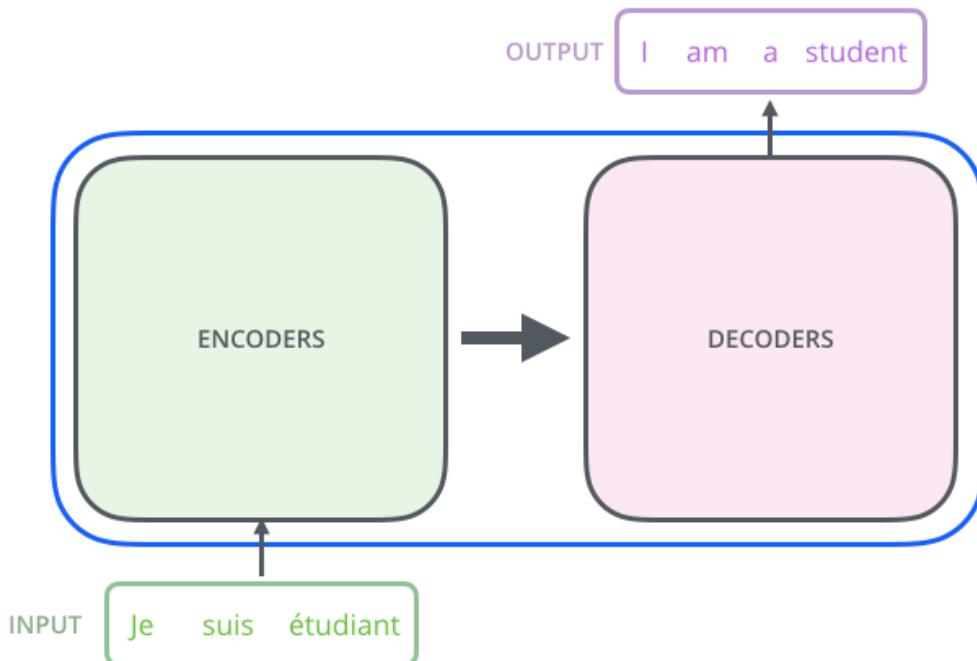
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.  
Hadi Veisi ([h.veisi@ut.ac.ir](mailto:h.veisi@ut.ac.ir))



## مبدل‌ها: معرفی ...

### ○ مبدل (Transformer) ...

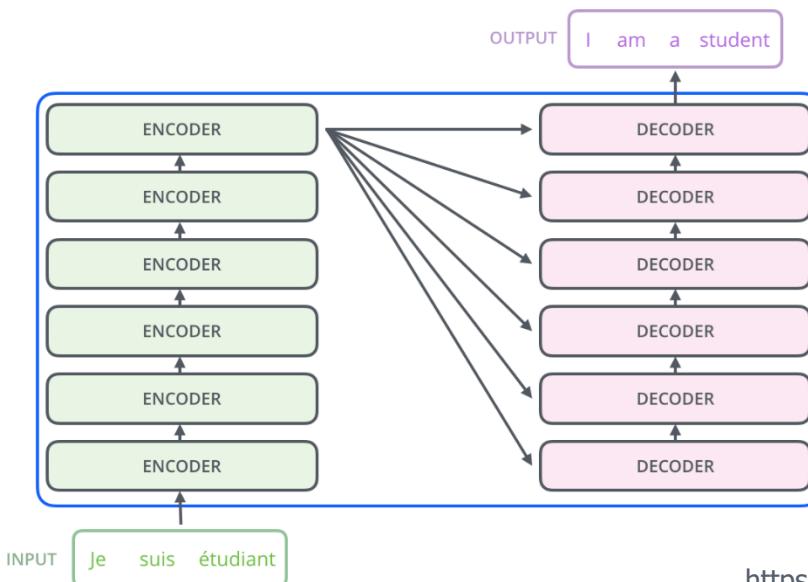
- استفاده از ساختار رمز‌گذار-رمزگشایی



## مبدل‌ها: ساختار ...

### ○ مبدل (Transformer)

- ساختار رمزگذار-رمزگشایی
- چند رمزگذار پشته شده روی هم: یادگیری وزن‌های هر کدام (متفاوت) در فرایند آموزش
- ورودی هر کدام خروجی رمزگذار قبلی است
- چند رمزگشایی پشته شده روی هم (هم تعداد رمزگذارها)
- خروجی آخرین رمزگذار = ورودی همه رمزگشاهای



<https://jalammar.github.io/illustrated-transformer/>

Hadi Veisi (h.veisi@ut.ac.ir)

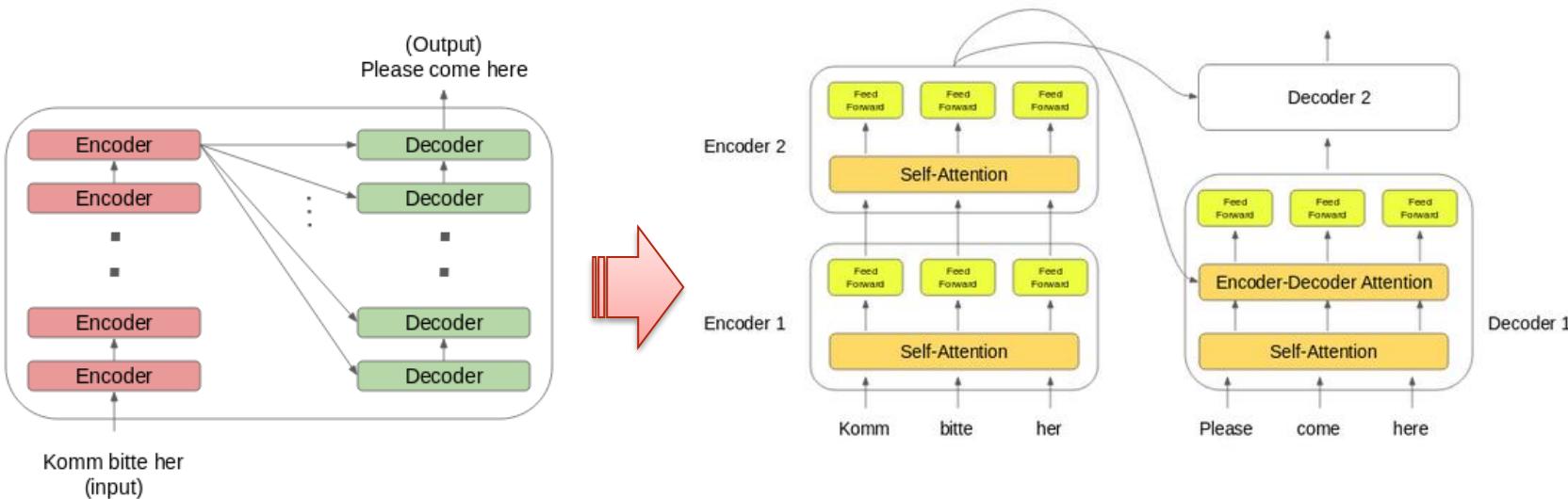
## مبدل‌ها: ساختار . . .

### ○ معما ری کلی

- تعداد N (مثلا ۶) رمزگذار و رمزگشای پشتی شده
- کل دنباله ورودی (همه کلمات) به اولین رمزگذار داده می‌شود

### ○ تفاوت با RNN‌ها

- خروجی هر رمزگذار به رمزگذار بعدی داده می‌شود
- خروجی آخرین رمزگذار به همه رمزگشایها داده می‌شود

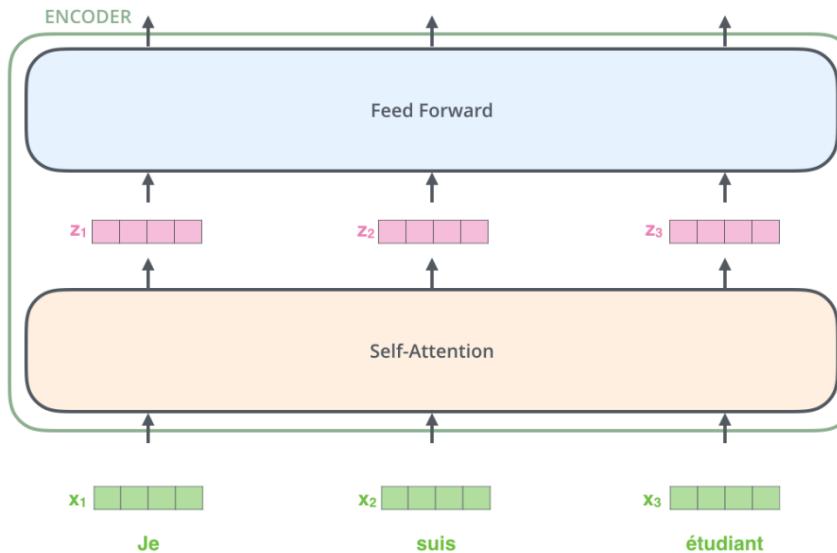


پیاده سازی در <http://nlp.seas.harvard.edu/2018/04/03/attention.html>

## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

- هر رمزگذار دو لایه دارد: توجه به خود و تمام متصل (جلورو)



- ورودی اولین رمزگذار: یک لیست از بردارهای تعبیه همه کلمات (نه یک کلمه)

○ اندازه ورودی شبکه = طول بزرگترین جمله دادگان

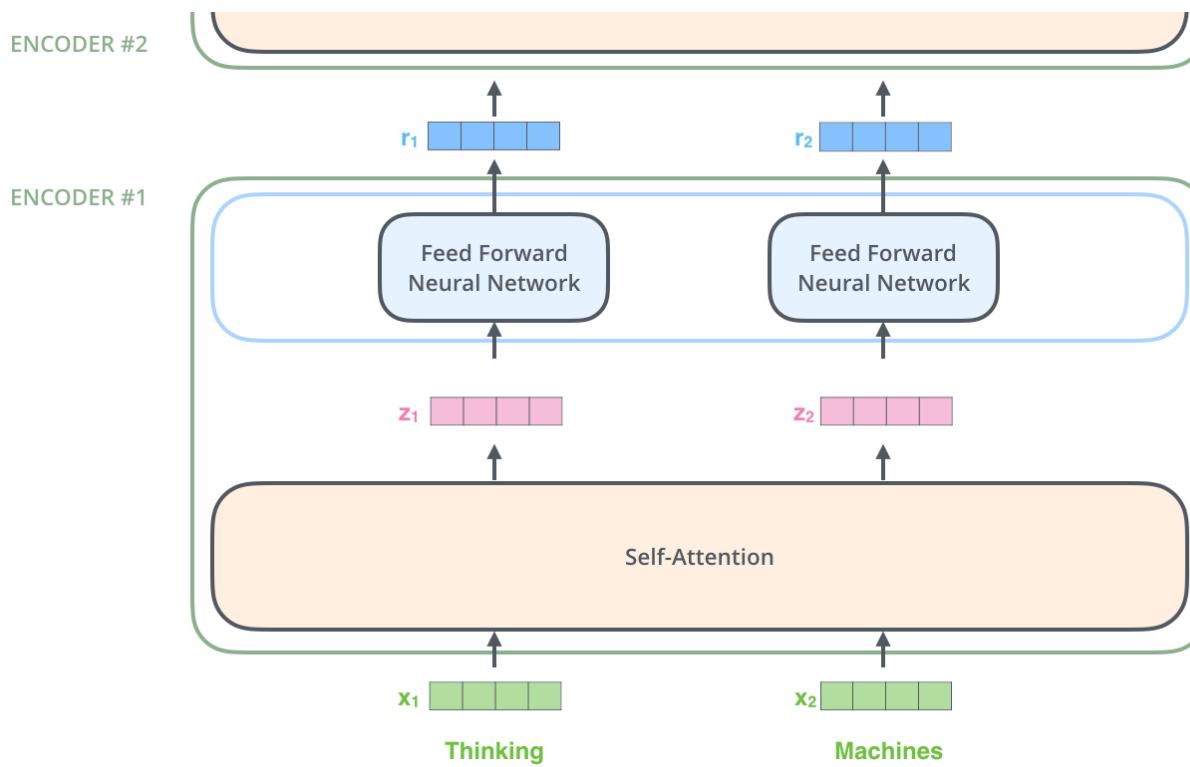
○ لایه توجه به خود = مدلسازی وابستگی بین کلمات + خروجی به تعداد کلمات ورودی

○ لایه جلورو = به تعداد کلمات شبکه مسقل از هم، عبور هر کلمه از یک شبکه

## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

- ورودی هر لایه توجه به خود: یک لیست از بردارهای همه کلمات (مدلسازی وابستگی)
- شبکه‌های مشابه در لایه جلوی و پردازش هر بردار کلمه به صورت مستقل

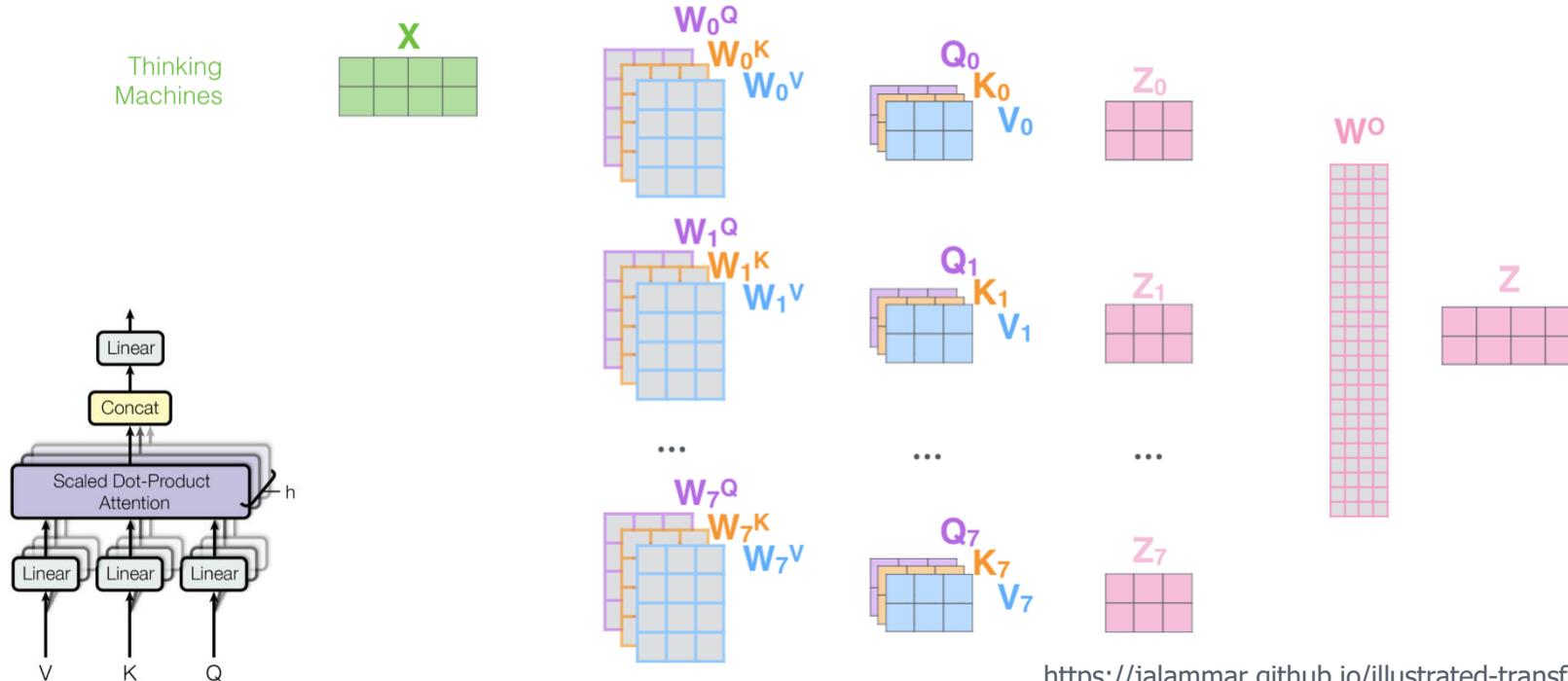


## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

- لایه توجه به خود (مثال: دو کلمه ورودی و ۸ هسته (Head))

- 1) This is our input sentence\*
- 2) We embed each word\*
- 3) Split into 8 heads. We multiply  $X$  or  $R$  with weight matrices
- 4) Calculate attention using the resulting  $Q/K/V$  matrices
- 5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^O$  to produce the output of the layer

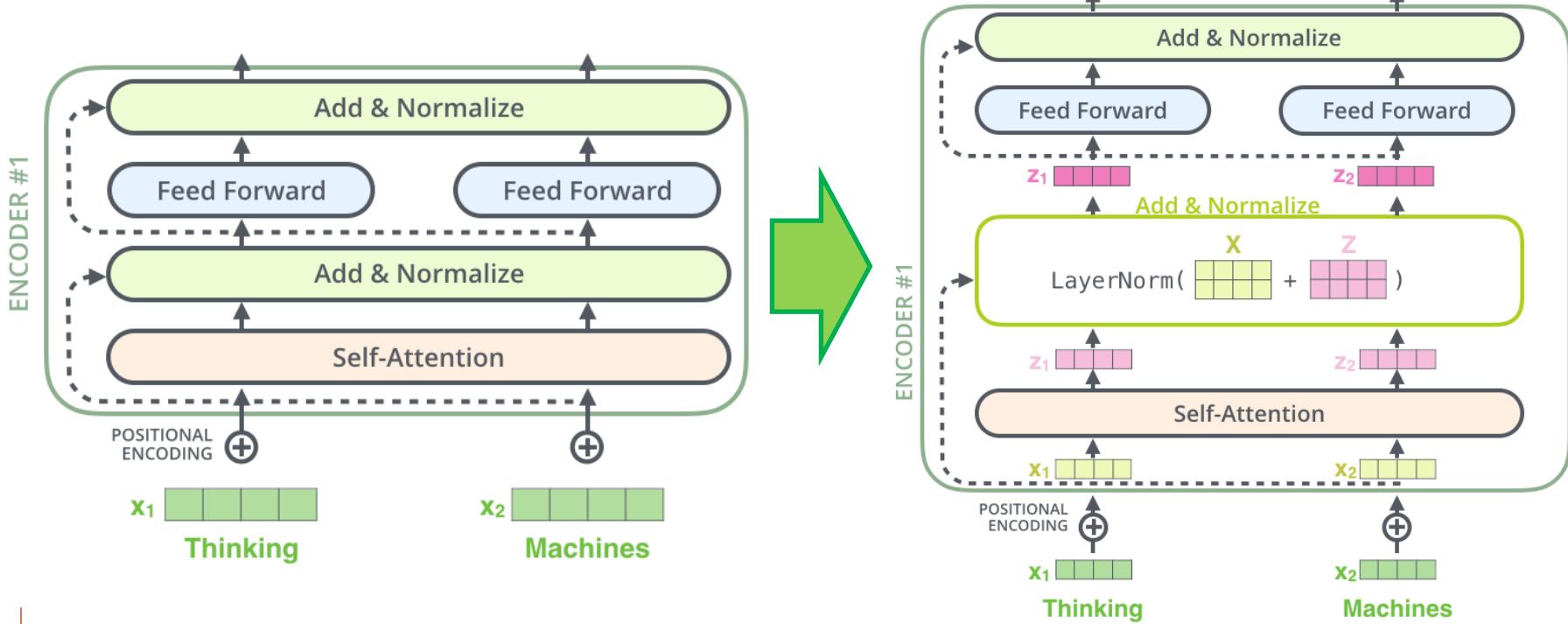




## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

- لایه جمع (باقیمانده) و نرمال‌سازی



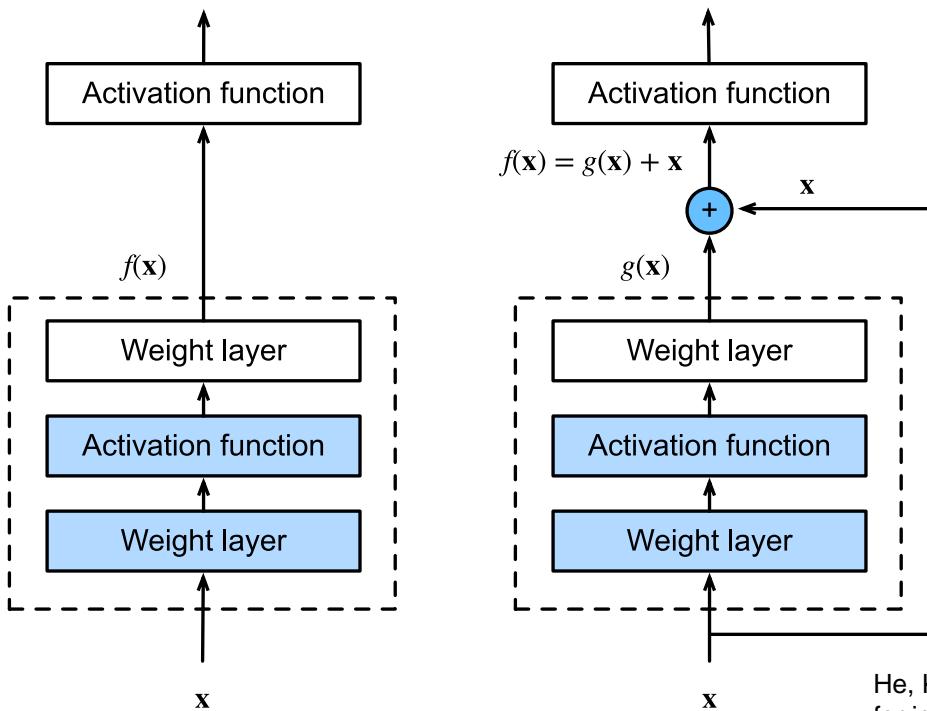
## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

- باقیمانده (Residuals) = مقدار تفاوت بین مقدار  $x$  و مقدار نگاشت

○ اگر  $f(x)$  نگاشت مورد نظر باشد آنگاه  $g(x) = f(x) - x$  مقدار باقیمانده است

○ ایده واحد باقیمانده: یادگیری مقدار باقیمانده به جای خود نگاشت و بدست آوردن مقدار نگاشت از روی آن، یعنی  $f(x) = g(x) + x$



○ ایده اصلی شبکه ResNet

## مبدل‌ها: ساختار ...

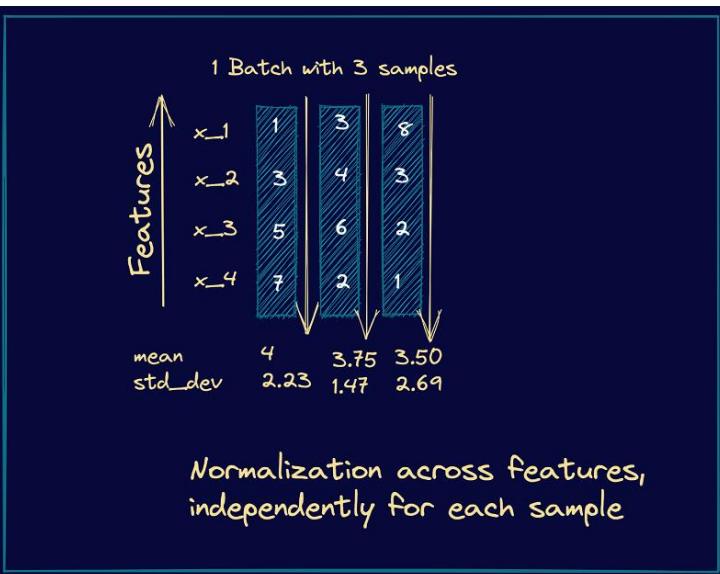
### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

#### • نرمال‌سازی لایه (Layer Normalization)

○ محاسبه میانگین و واریانس روی مقادیر ورودی نرون‌ها در هر لایه

○ جمع بستن روی همه نرون‌های هر لایه (تفاوت با نرمال‌سازی دسته‌ای)

• نرمال‌سازی دسته‌ای: هر نرون جداگانه و جمع بستن روی مقدار داده‌های ورودی یک دسته



$$\mu_l = \frac{1}{d} \sum_{i=1}^d x_i \quad (1)$$

$$\sigma_l^2 = \frac{1}{d} \sum_{i=1}^d (x_i - \mu_l)^2 \quad (2)$$

$$\hat{x}_i = \frac{x_i - \mu_l}{\sqrt{\sigma_l^2}} \quad (3)$$

$$or \hat{x}_i = \frac{x_i - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}} \quad (3)$$

*Adding  $\epsilon$  helps when  $\sigma_l^2$  is small*

$$y_i = \mathcal{LN}(x_i) = \gamma \cdot x_i + \beta \quad (4)$$

$d$ =تعداد نرون‌های هر لایه  
 $x_i$ =مقدار ورودی نرون  $i$ ام

گاما و بتا=پارامترهای قابل تنظیم در زمان آموزش برای ایجاد نوسان در ورودی (جلوگیری از الزام به صفر شدن همه ناشی از نرمال‌سازی)

## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

- در نظر گرفتن ترتیب کلمات
- افزودن یک بردار به بردار هر کلمه که بیانگر موقعیت کلمه در جمله (یا فاصله بین کلمات از هم) است
- بردار موقعیت = یک الگوی خاص که در فرایند آموزش یاد گرفته می‌شود
- استفاده از اندیس موقعیت برای جملات طولانی منجر به اعداد بزرگ می‌شود = عدم تعمیم
  - نیاز به روشی که بازه آن محدود باشد و برای جملات طولانی قابل تعمیم باشد
  - برای هر اندیس منجر به مقدار یکتاپی شود
  - برای فاصله بین هر دو اندیس سازگاری داشته باشد
- استفاده از یک بردار  $d$  بعدی

Sequence	Index of token	Positional Encoding Matrix			
I	0	$P_{00}$	$P_{01}$	...	$P_{0d}$
am	1	$P_{10}$	$P_{11}$	...	$P_{1d}$
a	2	$P_{20}$	$P_{21}$	...	$P_{2d}$
Robot	3	$P_{30}$	$P_{31}$	...	$P_{3d}$

## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

#### • رمزگذاری موقعیت کلمات: استفاده از توابع مثلثاتی

○ تابع سینوس برای اندیس‌های زوج و کسینوس برای اندیس‌های فرد

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

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

○ مقدار بین ۱- و +

○ pos = موقعیت کلمه در جمله (مثلا ۲ برای کلمه دوم)

○ d<sub>model</sub>: اندازه ابعاد بردار تعییه موقعیت (هم اندازه بردار تعییه کلمه)

○ i = اندیس مربوط به ابعاد بردار تعییه: بین صفر و d<sub>model</sub>/2 (استفاده از مقدار یکسان برای هر دو کلمه متوالی، یکسان برای سینوس و کسینوس)

Sequence	Index of token, POS	Positional Encoding Matrix with d=4, n=100			
		i=0	i=0	i=1	i=1
I	0	P <sub>00</sub> =sin(0) = 0	P <sub>01</sub> =cos(0) = 1	P <sub>02</sub> =sin(0) = 0	P <sub>03</sub> =cos(0) = 1
am	1	P <sub>10</sub> =sin(1/1) = 0.84	P <sub>11</sub> =cos(1/1) = 0.54	P <sub>12</sub> =sin(1/10) = 0.10	P <sub>13</sub> =cos(1/10) = 1.0
a	2	P <sub>20</sub> =sin(2/1) = 0.91	P <sub>21</sub> =cos(2/1) = -0.42	P <sub>22</sub> =sin(2/10) = 0.20	P <sub>23</sub> =cos(2/10) = 0.98
Robot	3	P <sub>30</sub> =sin(3/1) = 0.14	P <sub>31</sub> =cos(3/1) = -0.99	P <sub>32</sub> =sin(3/10) = 0.30	P <sub>33</sub> =cos(3/10) = 0.96

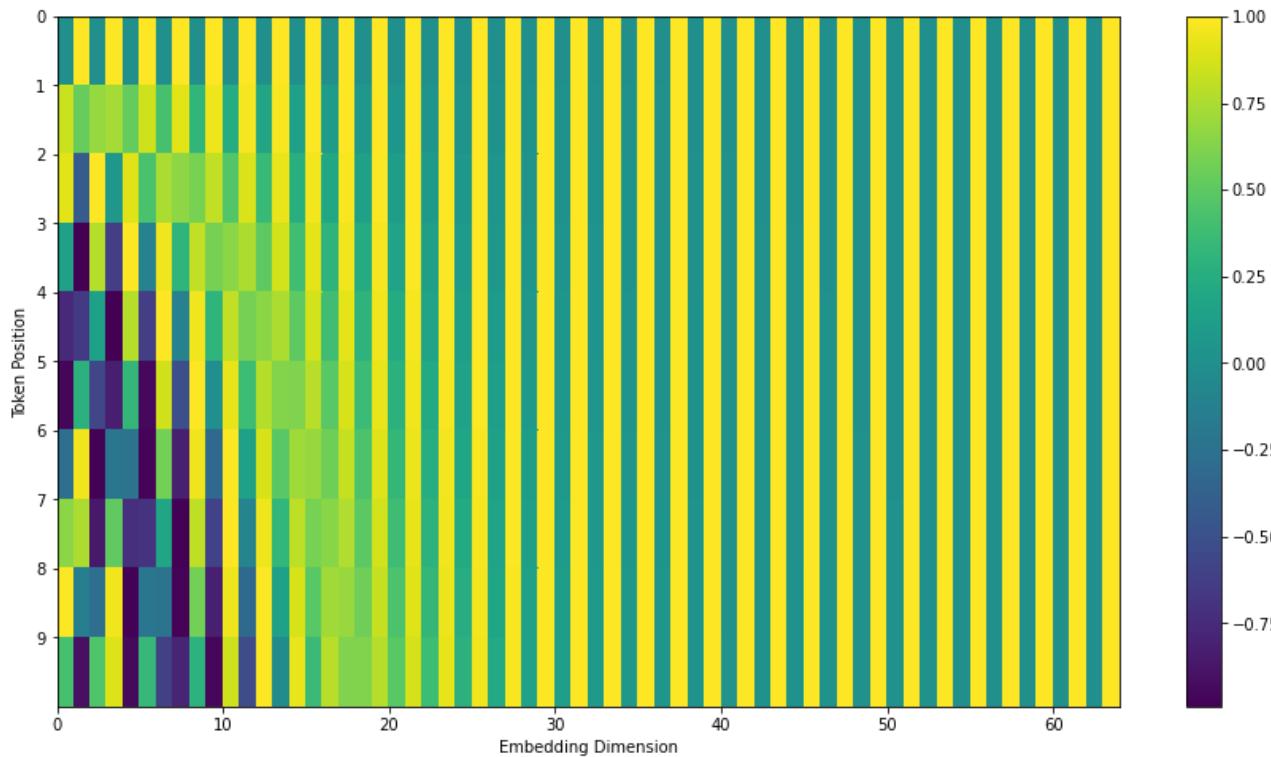
$$\sin\left(\frac{pos}{10000^{2i/d}}\right)$$

## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی) ...

- رمزگذاری موقعیت کلمات

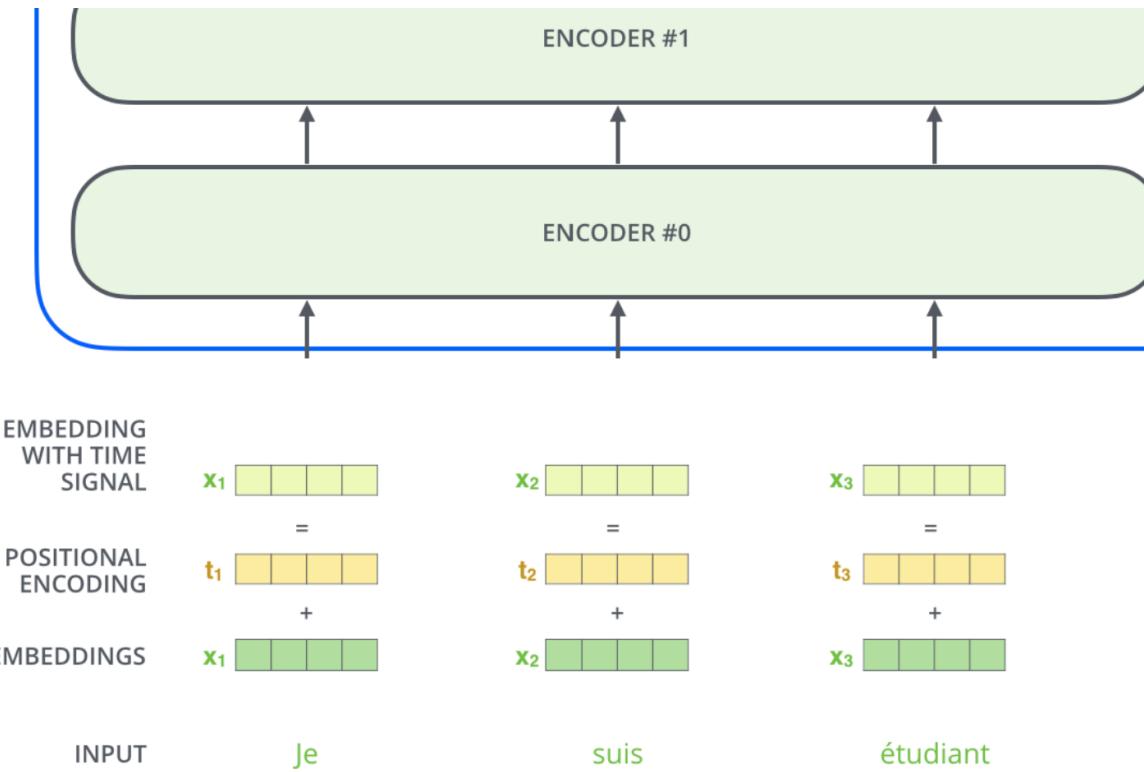
○ مثال: بردارهای موقعیت ۶۴ بعدی برای ۱۰ کلمه (هر ردیف بردار موقعیت یک کلمه)



## مبدل‌ها: ساختار ...

### ○ ساختار رمزگذار (مثال ترجمه ماشینی)

- جمع کردن بردار موقعیت کلمات با خود بردار کلمات و دادن آن به ورودی رمزگذار اول





## مبدل‌ها: ساختار ...

### ○ ساختار رمزگشایی (مثال ترجمه ماشینی) ...

- لایه توجه به خود

◦ ورودی = خروجی‌های تولید شده تاکنون توسط رمزگشا  
(تعییه کلمات خروجی + بردار موقعیت آنها)

- Softmax Mask

- لایه جمع و نرمال‌سازی

◦ ورودی = خروجی لایه قبل

- لایه توجه به خود (Encoder-Decoder Attention)

◦ ورودی = خروجی تبدیل شده (با ضرب در ماتریس خطی)  
آخرین رمزگذار به عنوان Key و Value و خروجی  
لایه قبل خودش در رمزگشا به عنوان Query

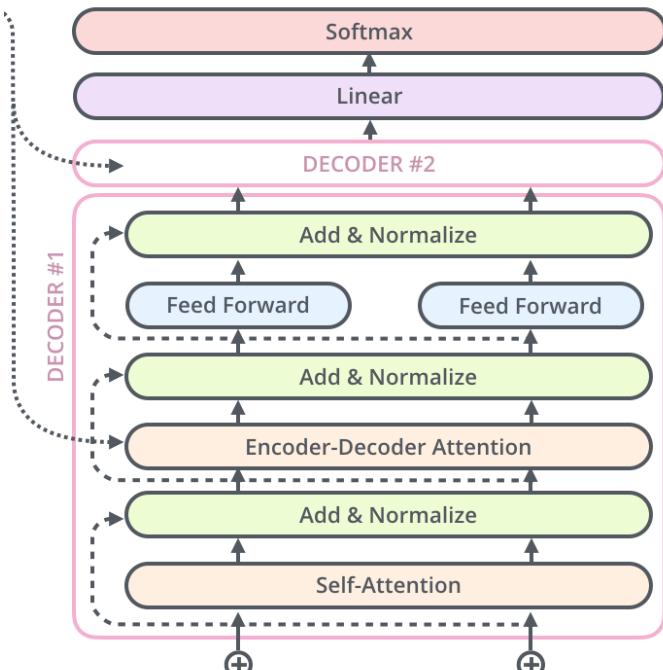
- لایه جمع و نرمال‌سازی

◦ ورودی = خروجی لایه قبل

- لایه جلو رو: شبکه‌های مستقل به تعداد کلمات ورودی

- لایه جمع و نرمال‌سازی

◦ ورودی = خروجی لایه قبل، خروجی = ورودی رمزگشای بعدی  
**Hadi Veisi (h.veisi@ut.ac.ir)**

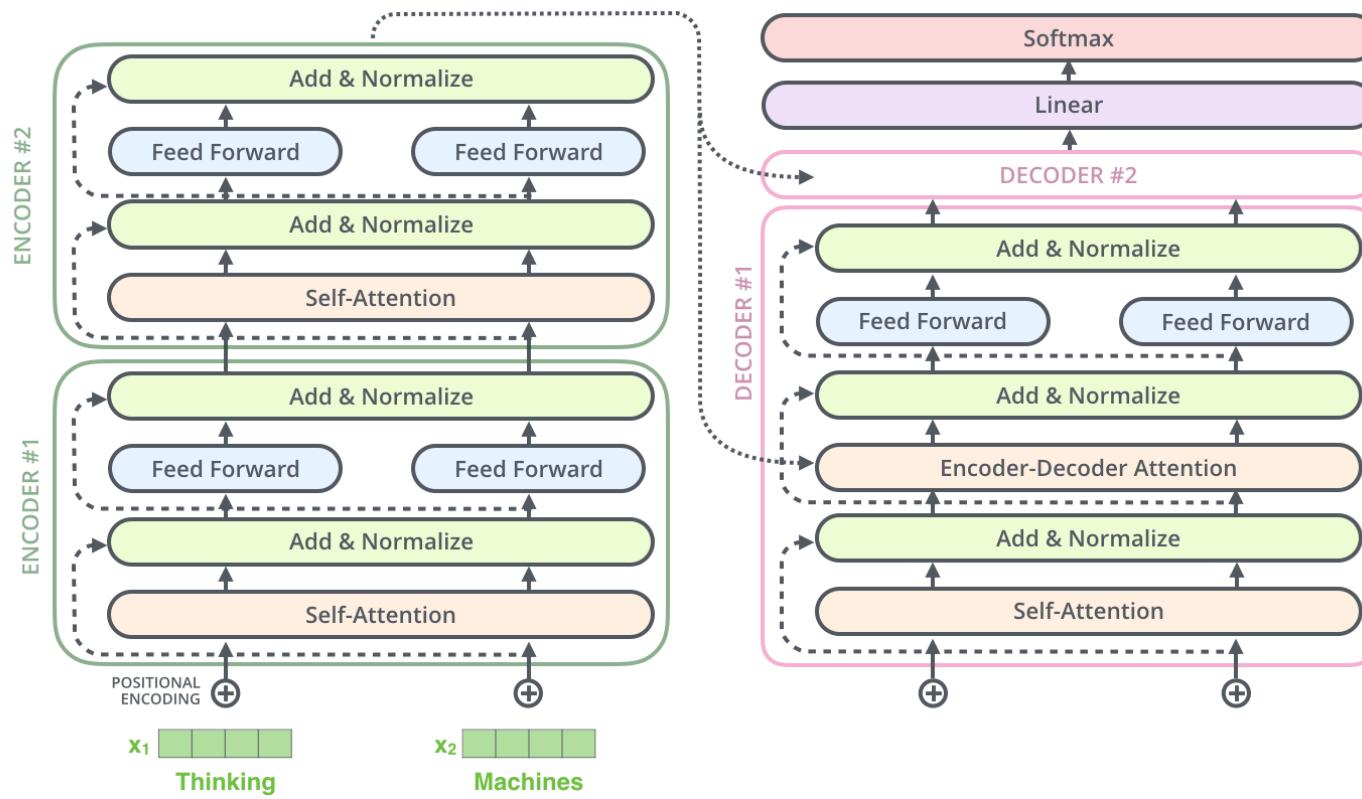




## مبدل‌ها: ساختار ...

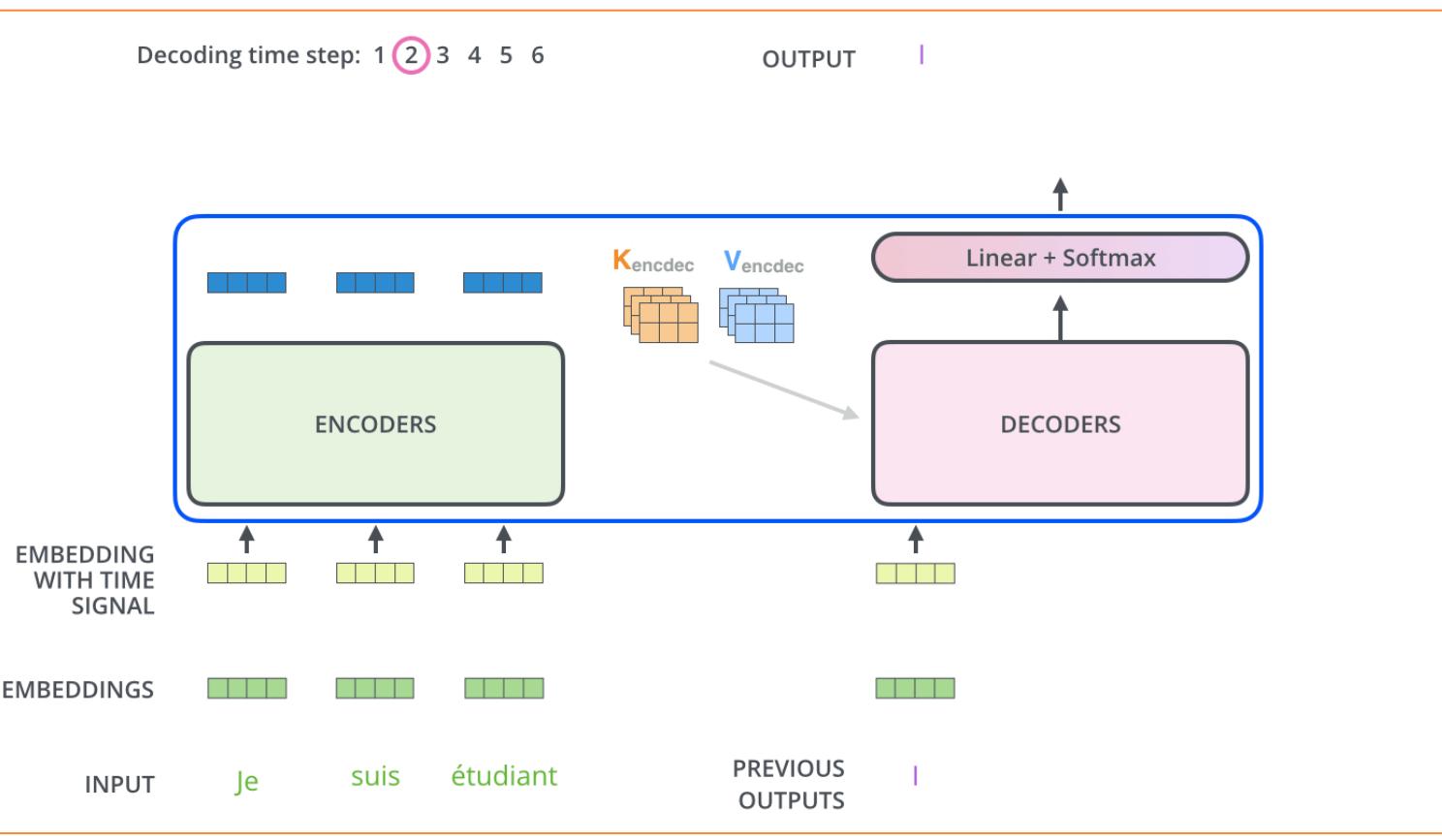
### ○ ساختار رمزگشایی (مثال ترجمه ماشینی) ...

- رابطه رمزگذار و رمزگشایی



## مبدل‌ها: ساختار ...

- ساختار رمزگشایی (مثال ترجمه ماشینی) ...
- به ازای هر کلمه خروجی، رمزگشایی یک بار اجرا می‌شود





## مبدل‌ها: ساختار ...

### ○ ساختار آخرین لایه خطی و SoftMax

#### ○ آخرین لایه خطی بعد از رمزگشایی

○ وظیفه: تبدیل بردار خروجی رمزگشایی به کلمه معادل

○ یک نبدیل خطی برای تبدیل بردار خروجی آخرین رمزگشایی به برداری با ابعاد برابر با تعداد کل کلمات واژگان (بردار Logit با ابعاد چندهزار)

○ لایه سافت‌مکس: تبدیل بردار لاجیت به احتمال و انتخاب کلمه معادل با موقعیتی با بالاترین احتمال

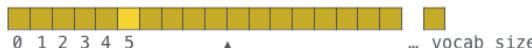
Which word in our vocabulary  
is associated with this index?

am

Get the index of the cell  
with the highest value  
(argmax)

5

`log_probs`



Softmax

`logits`



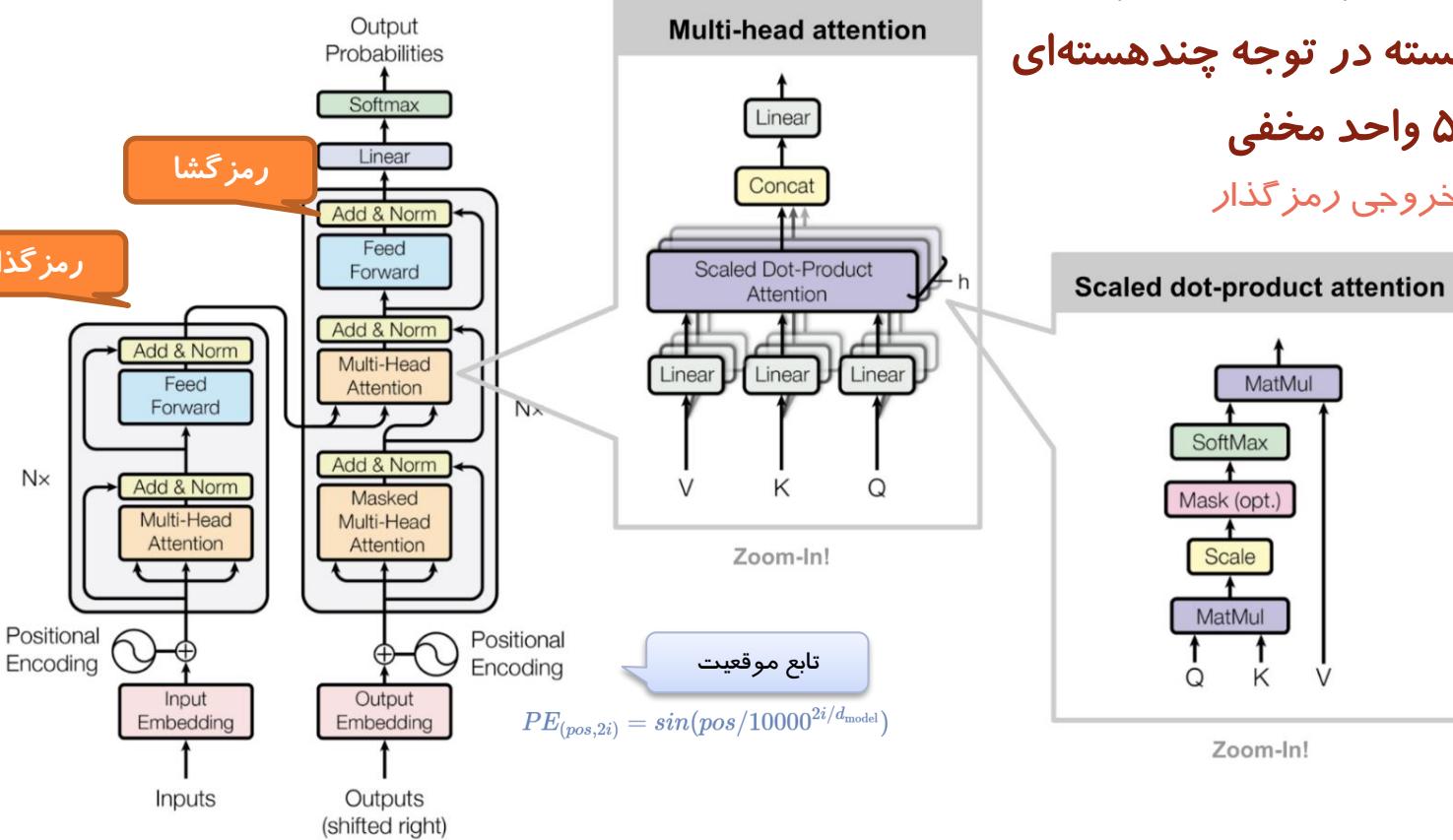
Linear

Decoder stack output

# مبدل‌ها ...

## مرور دوباره کل ساختار

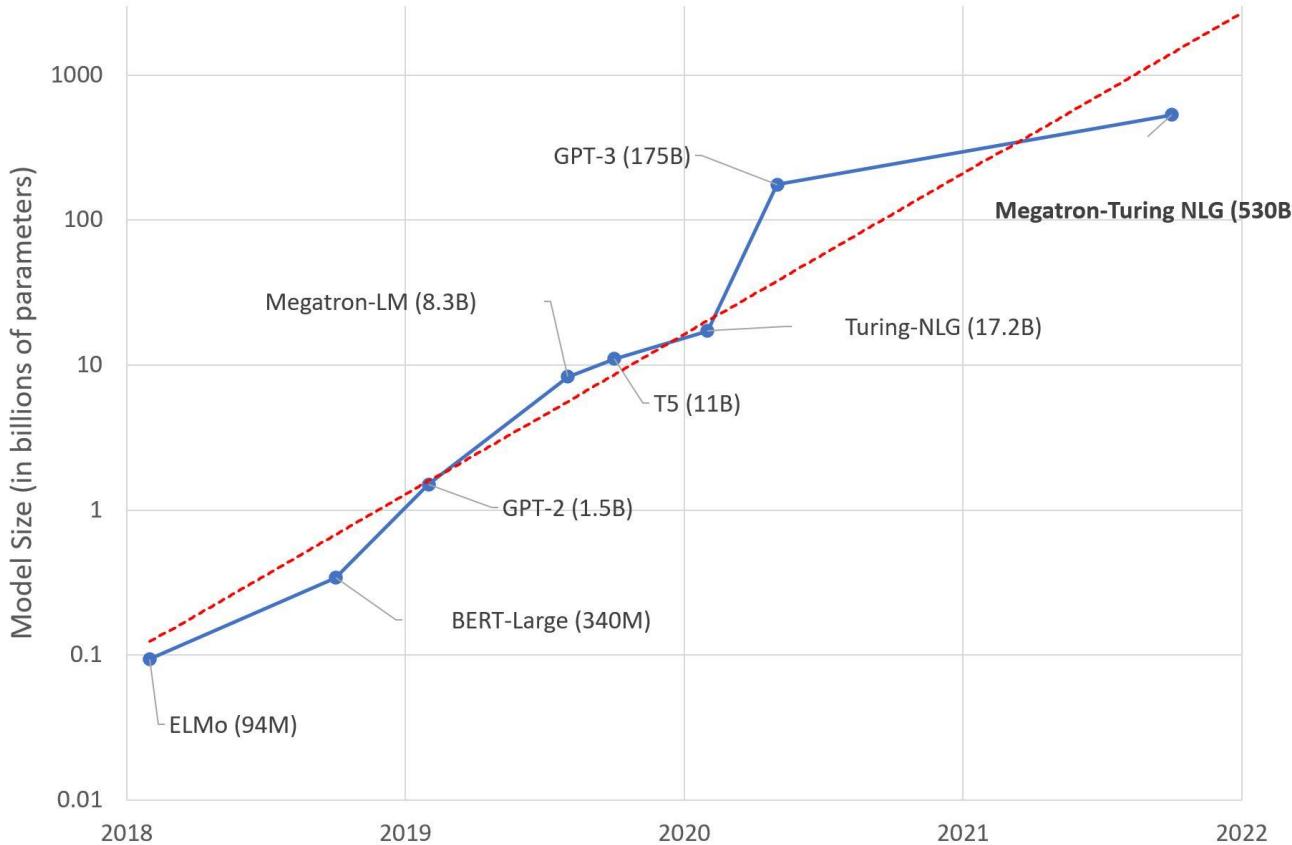
- ۶ لایه رمزگذاری و رمزگشایی
- ۸ هسته در توجه چندهسته‌ای
- ۱۲ واحد مخفی
- خروجی رمزگذار



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.  
**Hadi Veisi (h.veisi@ut.ac.ir)**

## مبدل‌ها: کاربردها...

### ○ ساخت مدل زبانی‌های بزرگ



<https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/>

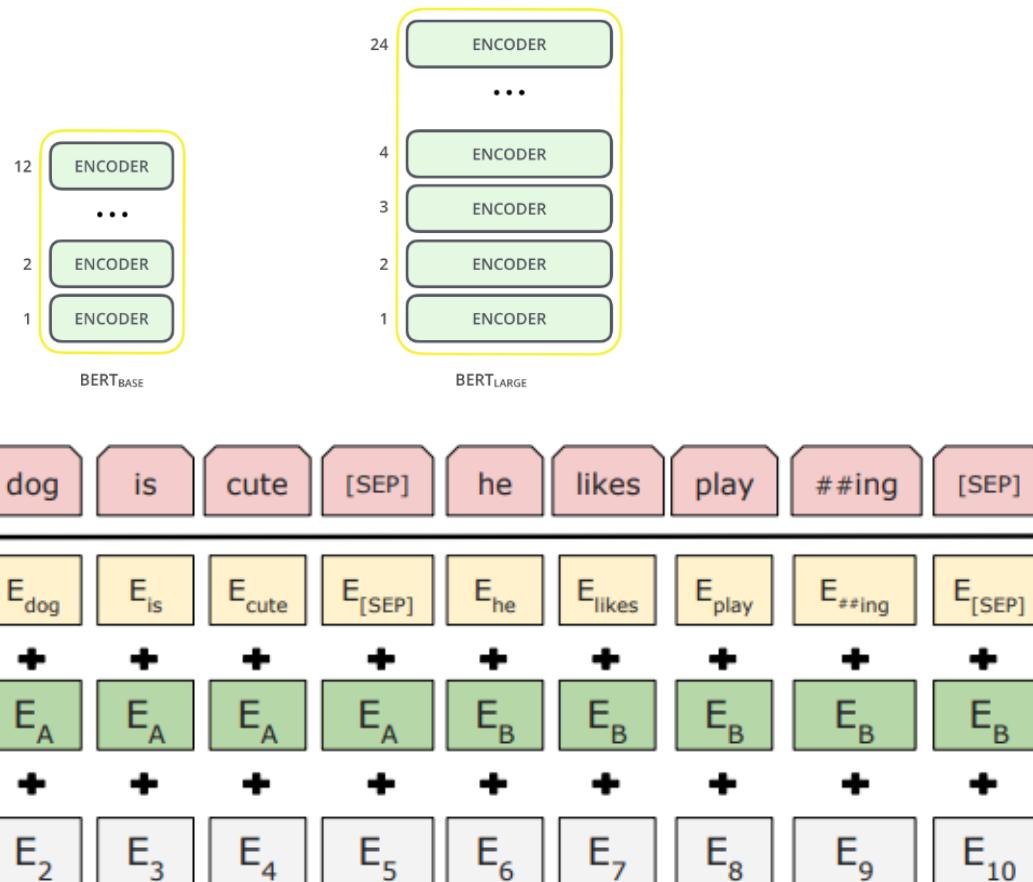
Hadi Veisi ([h.veisi@ut.ac.ir](mailto:h.veisi@ut.ac.ir))



## مبدل‌ها: کاربردها . . .

### ○ مدل زبانی و بردار تعابیر کلمات . . .

Bidirectional Encoder Representations from Transformers (BERT) •





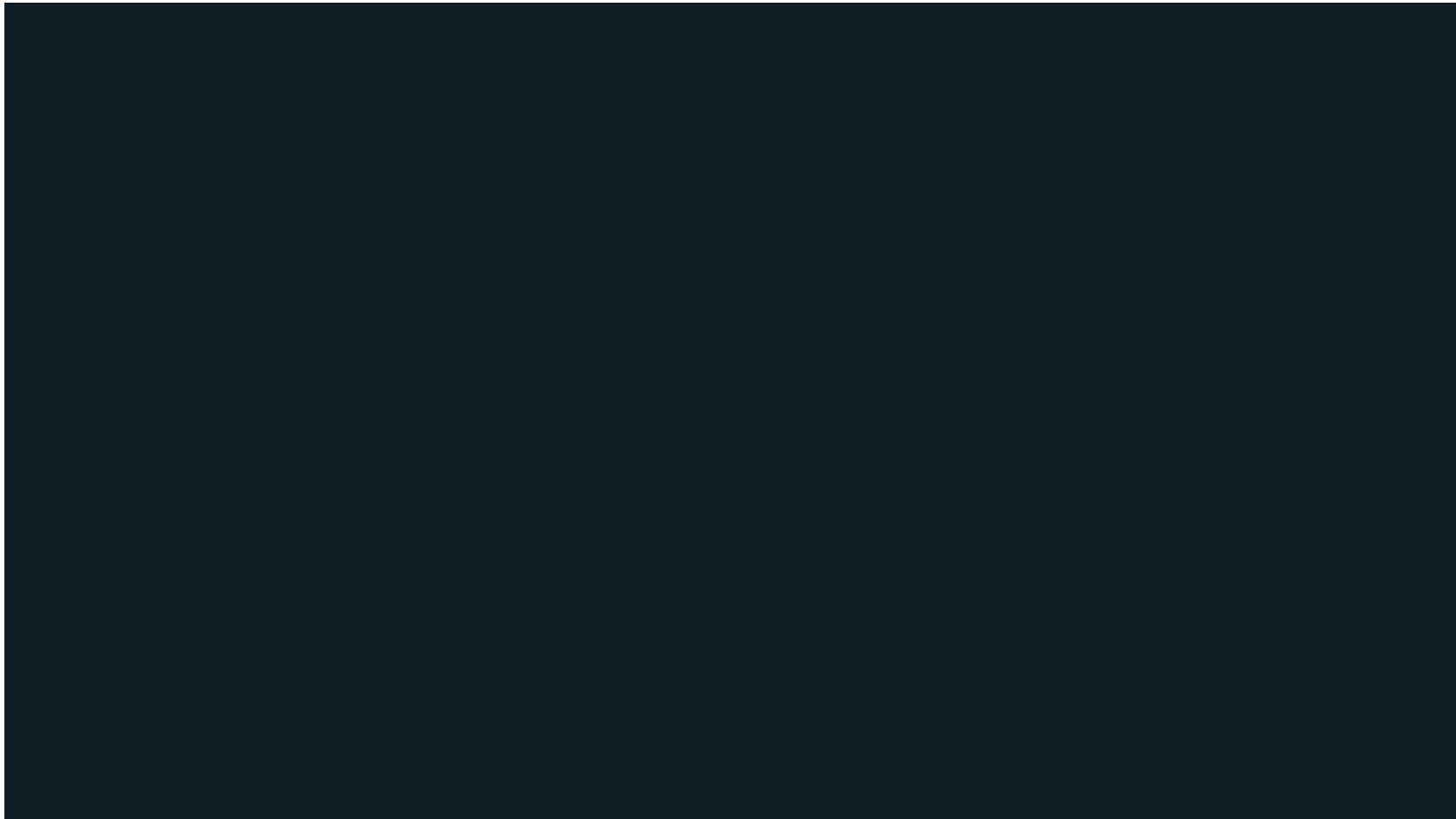
## مبدل‌ها: کاربردها . . .

### ○ مزایای BERT در مقایسه با Word2Vec

- در نظر گرفتن بافت
- تولید بردارهای متفاوت برای کلمه «شیر» با معانی مختلف
- در نظر گرفتن ترتیب
- ترتیب آمدن کلمات در جمله به صورت مستقیم مدل می‌شود
- نوع تعییه مبتنی بر جمله است
- برای در نظر گرفتن بافت لازم است کل جمله به مدل داده شود تا برای جمله و کلمات بردار تعییه ایجاد شود (در نظر گرفتن بافت) اما Word2Vec برای هر کلمه یک بردار تولید می‌کند (روش BERT هم می‌تواند برای تولید بردار کلمات به تنهایی استفاده شود اما در این حالت بافت در نظر گرفته نمی‌شود)
- بردار تعییه برای کلمات خارج از واژگان (OOV)
- امکان تولید بردار برای هر کلمه با توجه به تولید بردار بر اساس زیرکلمات (نویسه‌ها)

## مبدل‌ها: کاربردها . . .

- مدل زبانی و بردار تعابیر کلمات



## مبدل‌ها: کاربردها . . .

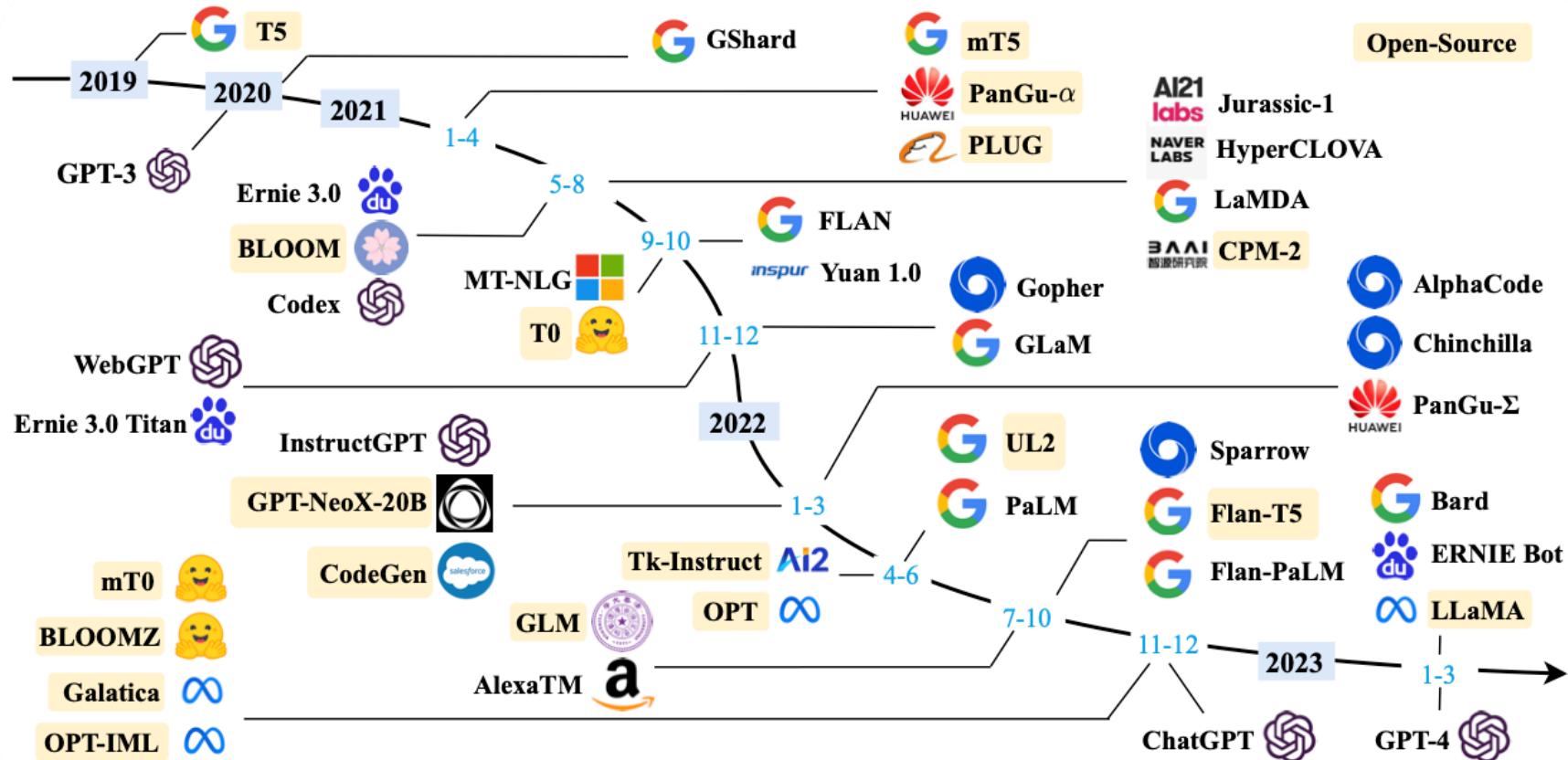
### پروژه GPT-3 (Generative Pre-trained Transformer 3)

- مدل زبانی برای تولید متن طبیعی: تولید داستان، خلاصه سازی، شعر گفتن، کد نوشتن و ...
- توسط OpenAI در ۲۰۲۰
- دارای ۱۷۵ میلیارد پارامتر = بزرگترین شبکه عصبی ساخته شده تاکنون
- هزینه آموزش شبکه = ۴.۶ میلیون دلار
- آموزش روی حدود ۵۰۰ میلیارد واحد (کلمه) از متون مختلف (کتاب، اینترنت و ...)



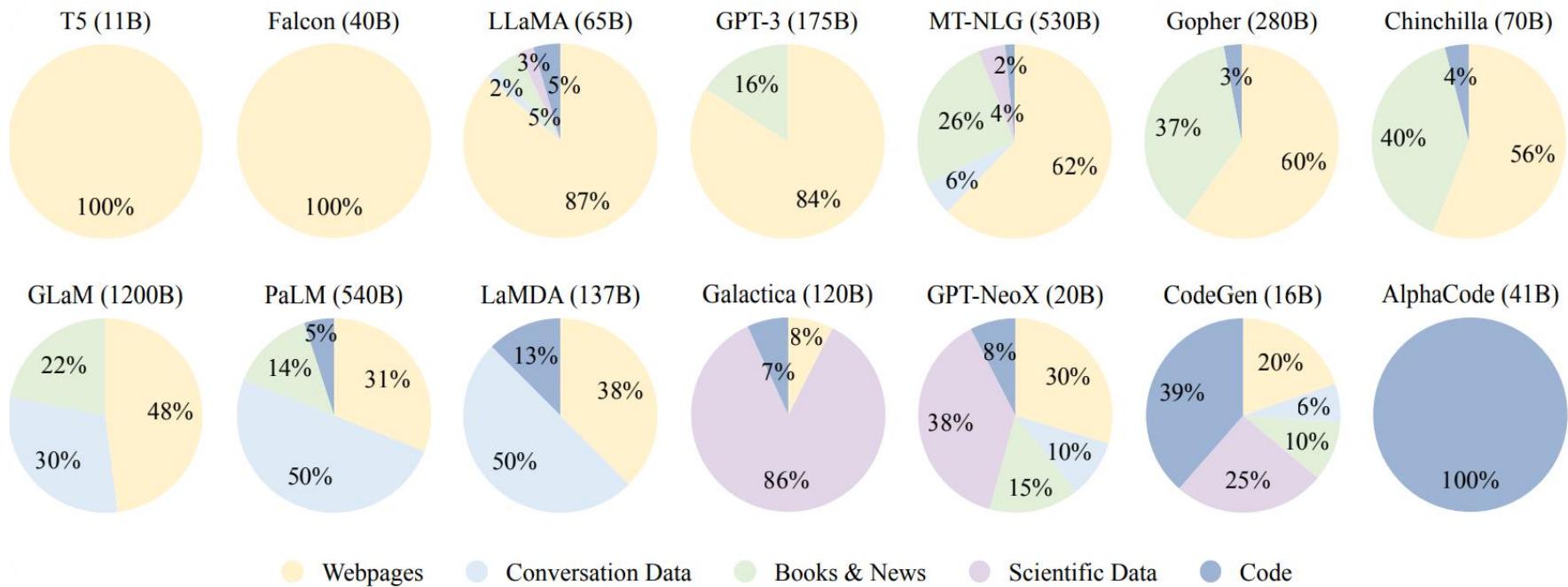


# مبدل‌ها: مدل زبانی‌های بزرگ (LLM) ...



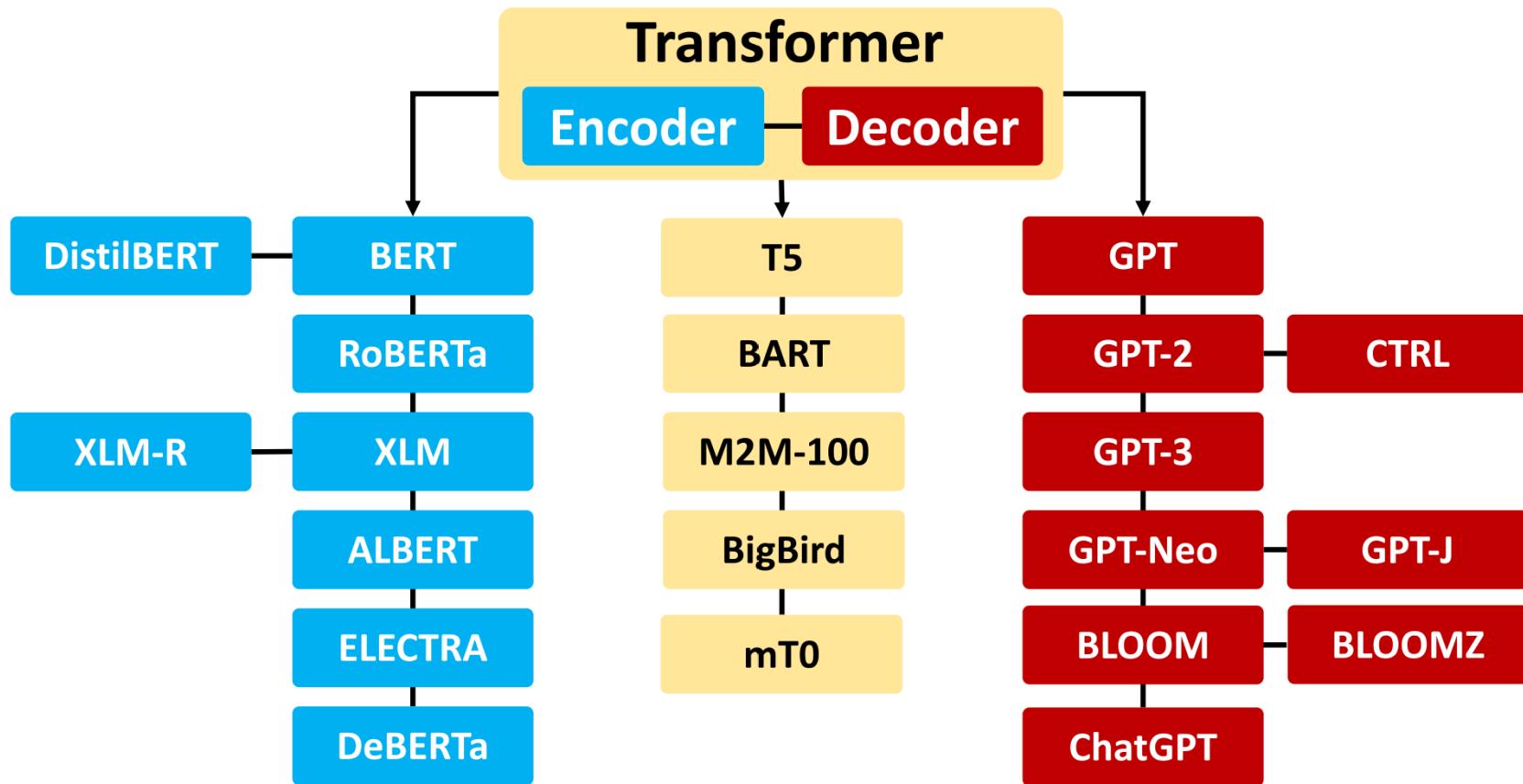


# مبدل‌ها: مدل زبانی‌های بزرگ (LLM) ...



2023, Wayne Xin Zhao, et al., A Survey of Large Language Models, arXive

# مبدل‌ها: مدل زبانی‌های بزرگ (LLM) ...

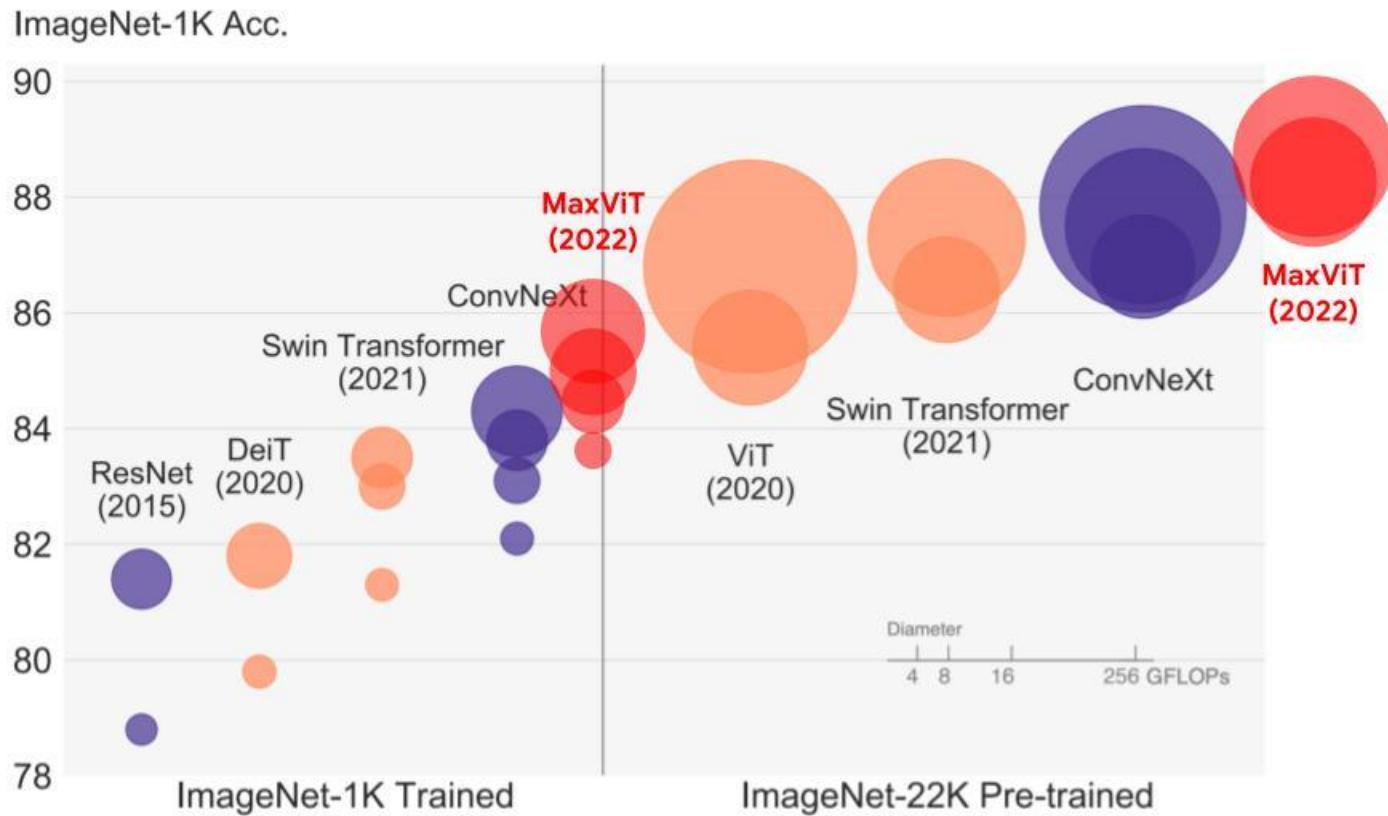




# مبدل‌ها: مبدل بینایی (Vision Transformer)



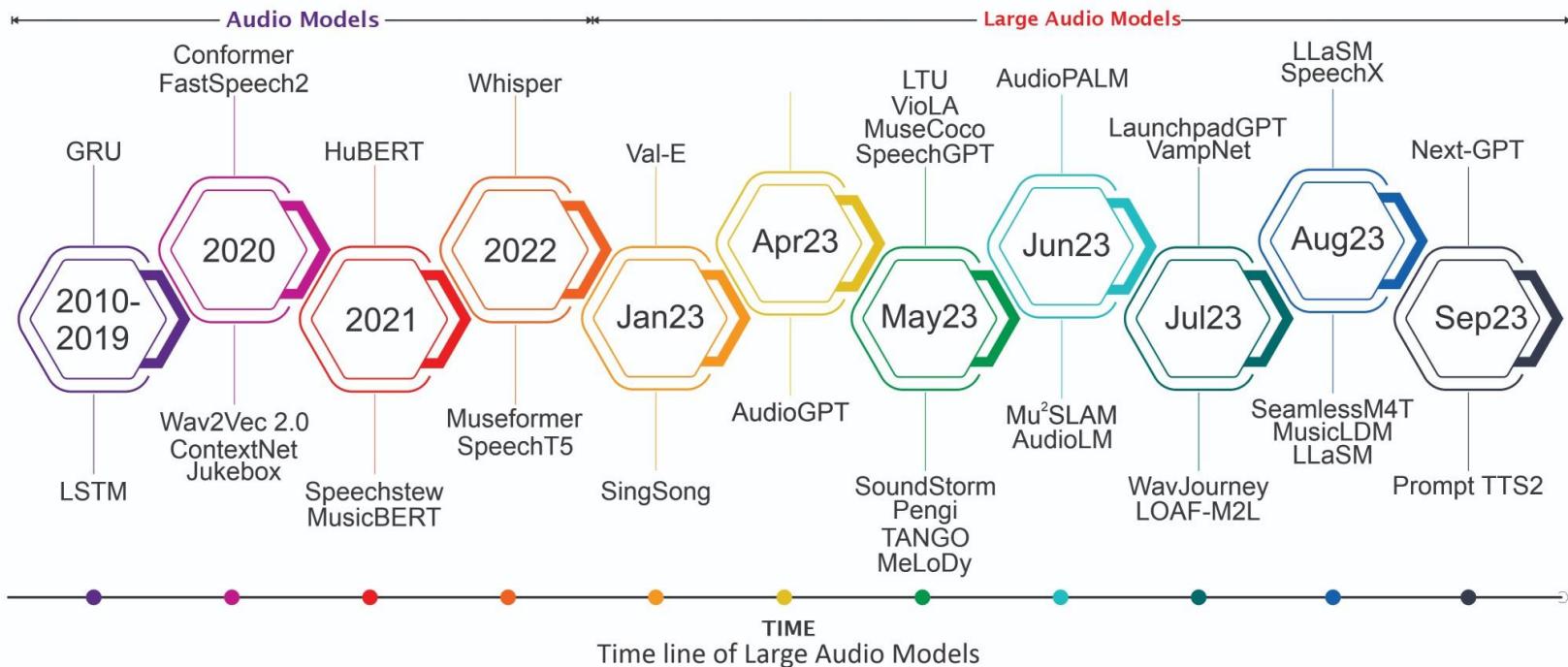
# مبدل‌ها: مبدل بینایی (Vision Transformer)



<https://arxiv.org/abs/2204.01697>



# مبدل‌ها: مدل‌های بزرگ صوتي ...



<https://github.com/EmulationAI/awesome-large-audio-models>  
<https://arxiv.org/pdf/2308.12792.pdf>



# مبدل‌ها: مدل‌های بزرگ صوتي ...

LLM/Paper	Train data	Tasks						Code
		ASR	TTS	ST	SP	SD	Others	
SpeechGPT [91]	Gigaspeech Common Voice LibriSpeech SpeechInstruct	✓	✓	✗	✗	✓	-	✗
AudioPaLM [95]	CoVoST2, CVSS VoxPopuli ASR Common Voice Conversational EsEn LibriSpeech YouTube ASR WMT/TED TTS PaLM MT TTS	✓	✓	✓	✗	✗	Machine Translation	✗
AudioLM [99]	Libri-Light	✗	✗	✗	✗	✗	Piano continuation Speech continuation	✗
LTU [108]	OpenAQA-5M	✗	✗	✗	✗	✗	Audio classification Audio captioning Summarisation	≈
VIOLA [114]	WenetSpeech Libri-Light LibriSpeech AI Challenger WMT2020 EMIME	✓	✓	✓	✗	✗	Machine translation	✗
SpeechX [120]	LibriLight DNS challenge corpus	✗	✓	✗	✗	✗	Noise suppression Speech removal Target speaker extraction Clean speech editing Noisy speech editing	✗
VALL-E [115]	LibriLight	✗	✓	✓	✗	✗	-	✗
Mu <sup>2</sup> SLAM [121]	mC4 dataset VoxPopuli, MLS, Babel, CoVoST FLEURS.	✓	✗	✓	✗	✗	Machine Translation	✗
SoundStorm [100]	LibriLight	✗	✗	✗	✗	✓	-	✗
AudioGPT [122]	LibriTTS MUSTC CHiME4 AudioSet AudioCaption and others	✓	✓	✓	✗	✓	Style Transfer Speech Enhancement Speech Separation Mono-to-Binaural Audio Inpainting Sound Extraction Image-to-Audio Singing Synthesis and others	✓
Pengi [123]	Clotho AudioCaps UrbanSound8K TUT 2017 CREMA-D FSD50K and others	✓	✓	✓	✓	✗	Audio Captioning Audio Question Answering Sound Scene Classification Music Analysis Instrument Classification Vocal Sound Classification and others	≈
SeamlessM4T [124]	1 million hours of open speech audio data	✓	✓	✓	✗	✗	Machine Translation Speech, Text-to-Text -Translation	✗
NExT-GPT [125]	T2M MosIT	✓	✓	✓	✗	✗	Text-to-Image Text-to-Video Text-to-Image	✓

<https://arxiv.org/pdf/2308.12792.pdf>

## مبدل‌ها: کاربردها

روش غالب در همه/بیشتر (!) کاربردهای مدل‌سازی دنباله