بسم الله الرحمن الرحیم

یادگیری عمیق

مدرس: محمدرضا محمدی

بهار ۱۴۰۲

مکانیزم‌های توجه

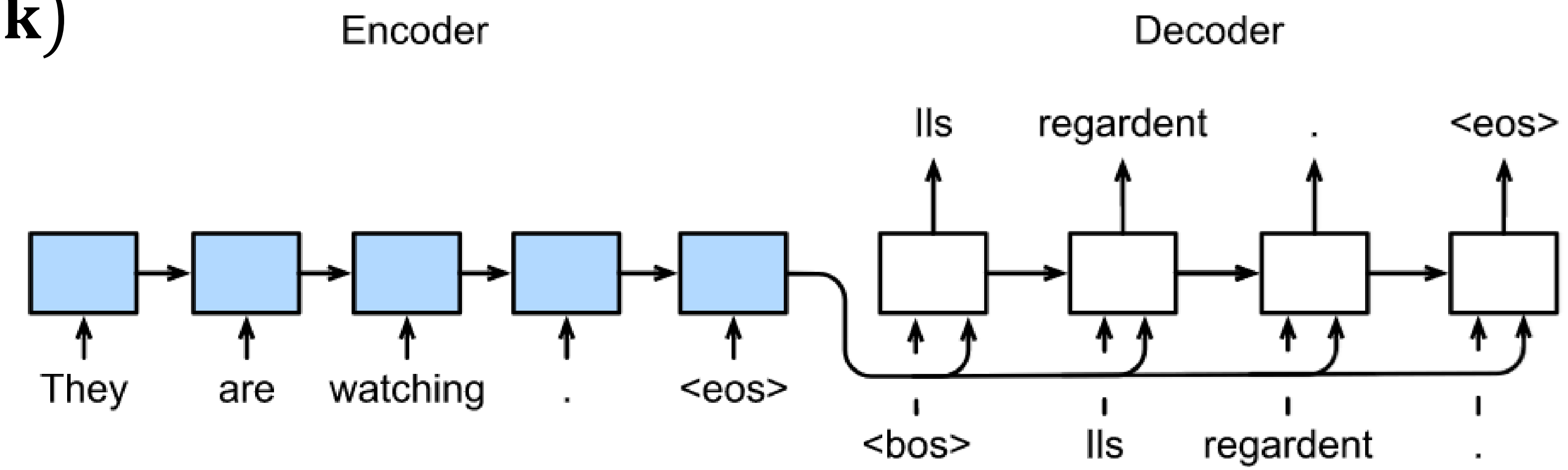
Attention Mechanisms

توجه Bahdanau

- در مدل پیشنهادی، بجای استفاده از $\mathbf{c}$ یکسان در تمام گام‌ها، از $\mathbf{c}_{t'}$ استفاده می‌کنیم
  - فرض کنید دنباله ورودی دارای T توکن باشد، $\mathbf{c}_{t'}$ به صورت زیر محاسبه می‌شود:
  - که $\mathbf{s}_{t'-1}$ حالت پنهان decoder در گام $t'-1$ به عنوان query است
  - و $\mathbf{h}_t$ حالت پنهان encoder در گام t به عنوان key و همچنین value است
  - با توجه به اینکه طول $\mathbf{s}$ و $\mathbf{h}$ می‌تواند متفاوت باشد، از تابع امتیازدهی توجه افزودنی استفاده می‌کنیم

$$\mathbf{c}_{t'} = \sum_{t=1}^{T} \alpha(\mathbf{s}_{t'-1}, \mathbf{h}_t)\mathbf{h}_t$$

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^{\mathrm{T}} \tanh\left(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}\right)$$

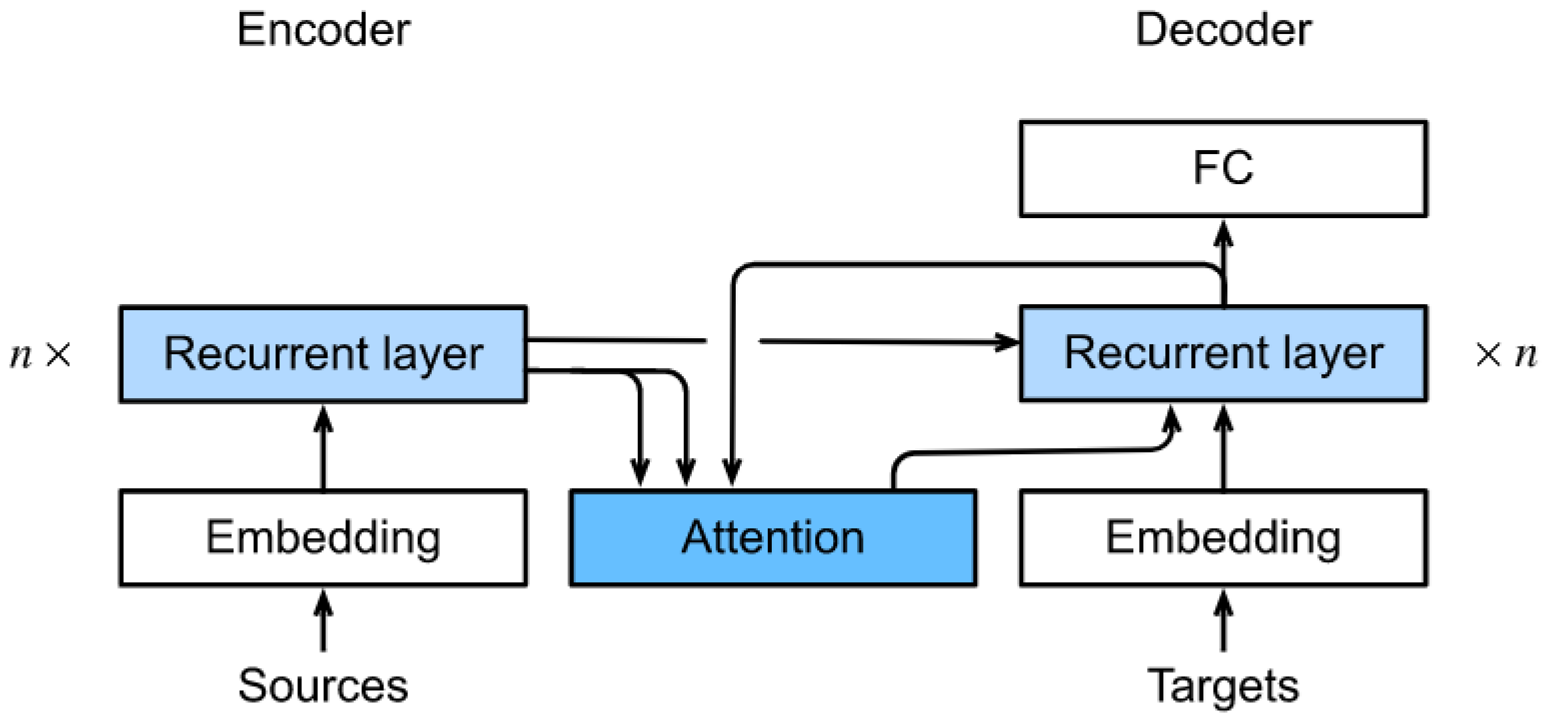


Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint,* 2014.

توجه Bahdanau

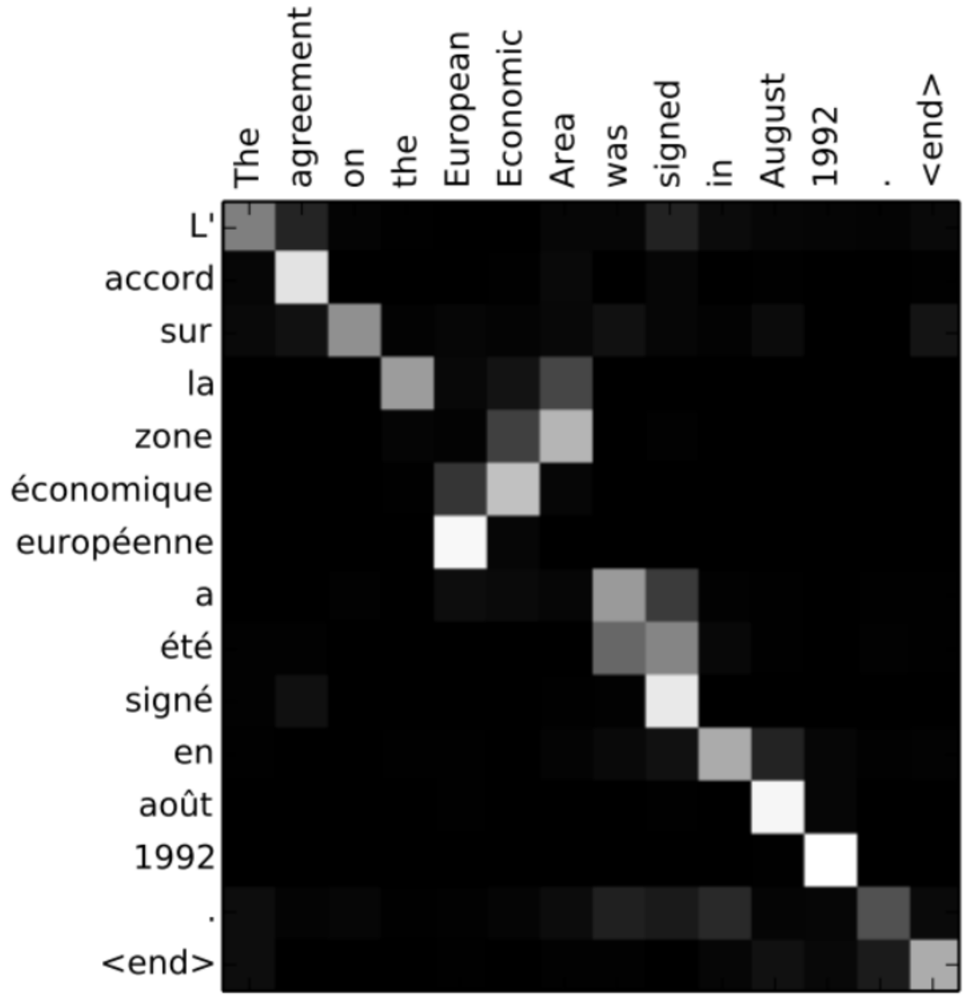
$$\mathbf{c}_{t'} = \sum_{t=1}^{T} \alpha(\mathbf{s}_{t'-1}, \mathbf{h}_t)\mathbf{h}_t$$

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^{\mathrm{T}} \tanh\left(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}\right)$$



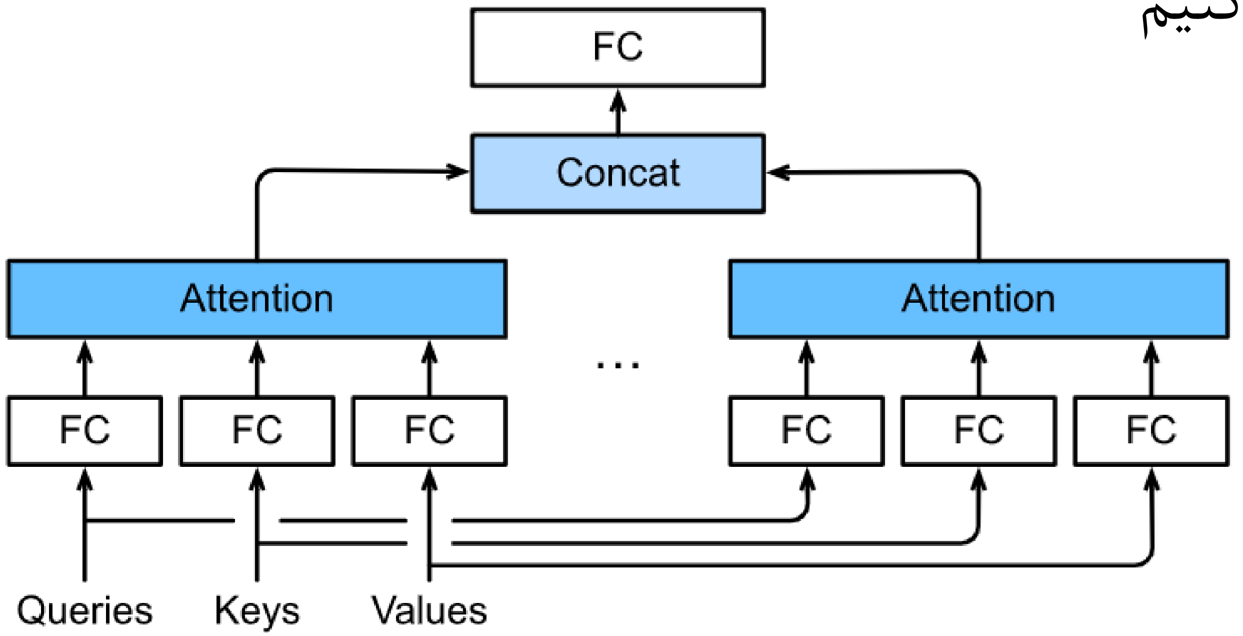
مثال: توجه در ترجمه ماشینی

- می‌توانیم بردار توجه در هر گام از decoder را نمایش دهیم
- در این مثال، یک جمله فرانسوی به طول ۱۵ به یک جمله انگلیسی به طول ۱۴ ترجمه شده است
- همانطور که مشاهده می‌شود، ترتیب توجه به کلمات متناسب است
  - در فرانسه ترتیب کلمات "zone économique européenne" برعکس انگلیسی است ("European Economic Area")



توجه چند سر (Multi-Head)

- در عمل، با داشتن یک مجموعه یکسان از queries، keys و values، ممکن است بخواهیم از جنبه‌های مختلفی با هم ترکیب شوند
  - به عنوان مثال، وابستگی‌های زمانی با طول مختلف (کوتاه‌مدت و بلندمدت) در یک دنباله
- می‌توانیم چندین تبدیل مستقل و موازی استفاده کنیم
  - سپس با هم الحاق و به صورت خطی ترکیب شوند



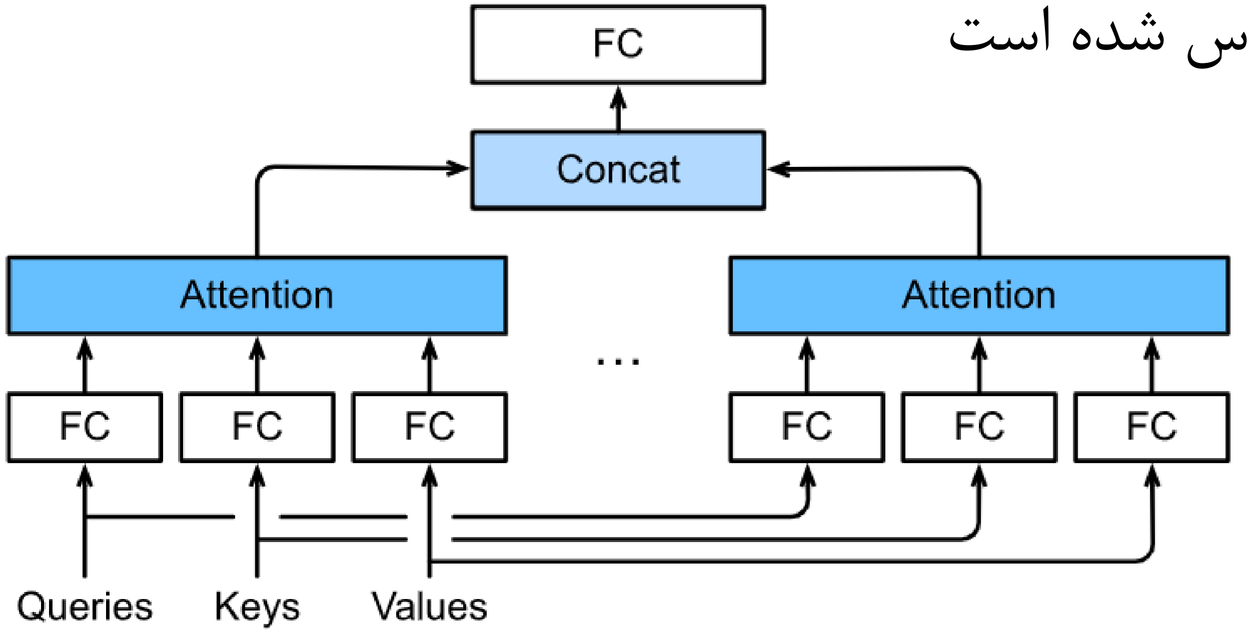
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*, 2017.

توجه چند سر (Multi-Head)

$$\mathbf{h}_i = f\left(\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}\right) \in \mathbb{R}^{p_v} \quad , \mathbf{q} \in \mathbb{R}^{d_q} \quad , \mathbf{k} \in \mathbb{R}^{d_k} \quad , \mathbf{v} \in \mathbb{R}^{d_v}$$

$$\mathbf{W}_i^{(q)} \in \mathbb{R}^{p_q \times d_q} \quad , \mathbf{W}_i^{(k)} \in \mathbb{R}^{p_k \times d_k} \quad , \mathbf{W}_i^{(v)} \in \mathbb{R}^{p_v \times d_v}$$

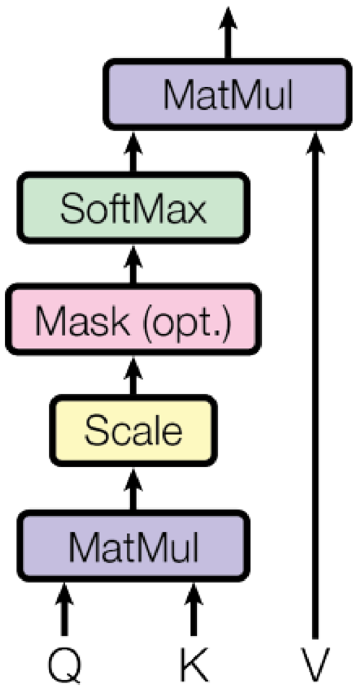
- f یکی از توابع ادغام توجه مانند ضرب‌داخلی مقیاس شده است



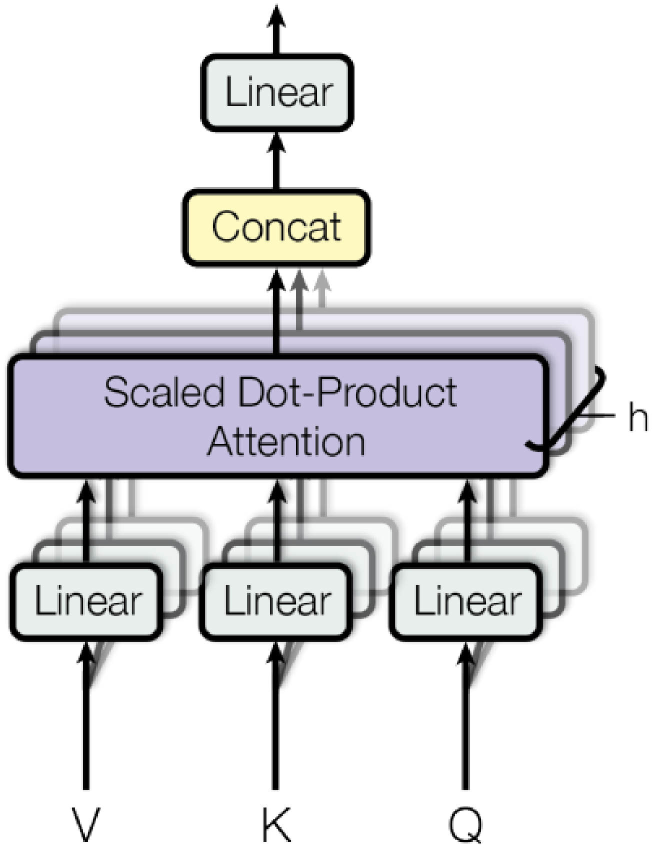
$$o = \mathbf{W}_o \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_h \end{bmatrix} \in \mathbb{R}^{p_o} \quad , \mathbf{W}_o \in \mathbb{R}^{p_o \times h p_v}$$

توجه چند سر (Multi-Head)

Scaled Dot-Product Attention



Multi-Head Attention



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*, 2017.

توجه به خود (Self-Attention)

- برای پردازش یک دنباله، می‌توان از ارتباط معنایی میان آنها استفاده کرد
- در توجه به خود، به دنبال دستیابی به توجه هر توکن نسبت به کل دنباله ورودی هستیم
  - مجموعه توکن‌ها به عنوان queries، keys و values استفاده می‌شوند

$$\mathbf{y}_i = f\left(\mathbf{x}_i, (\mathbf{x}_1, \mathbf{x}_1), \dots, (\mathbf{x}_n, \mathbf{x}_n)\right) \in \mathbb{R}^d$$

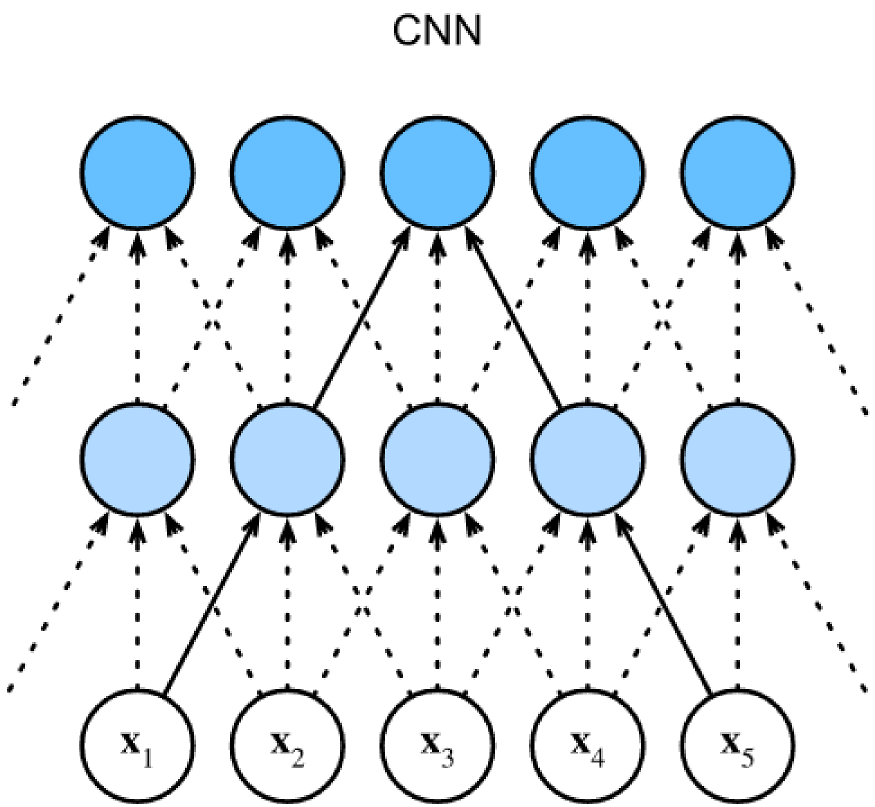
  - فرض کنید توکن‌های $\mathbf{x}_1, \dots, \mathbf{x}_n$ دنباله ورودی باشند که هر کدام dبعدی است و $\mathbf{y}_1, \dots, \mathbf{y}_n$ خروجی توجه به خود است
  - می‌تواند از توجه چند سر استفاده کرد

مقایسه توجه به خود با CNN و RNN

- برای نگاشت یک دنباله به طول n به دنباله‌ای با همان طول می‌توان از معماری‌های مختلفی استفاده کرد
- یادگیری وابستگی‌های دوربرد یک چالش کلیدی در بسیاری از کارهای پردازش دنباله است
- یکی از عوامل کلیدی که بر توانایی یادگیری چنین وابستگی‌هایی تأثیر می‌گذارد، طول مسیرهایی است که سیگنال‌ها باید در شبکه طی کنند
- هر چه این مسیرها بین هر ترکیبی از موقعیت‌ها در دنباله‌های ورودی و خروجی کوتاه‌تر باشد، یادگیری وابستگی های دوربرد آسان تر است

مقایسه توجه به خود با CNN و RNN

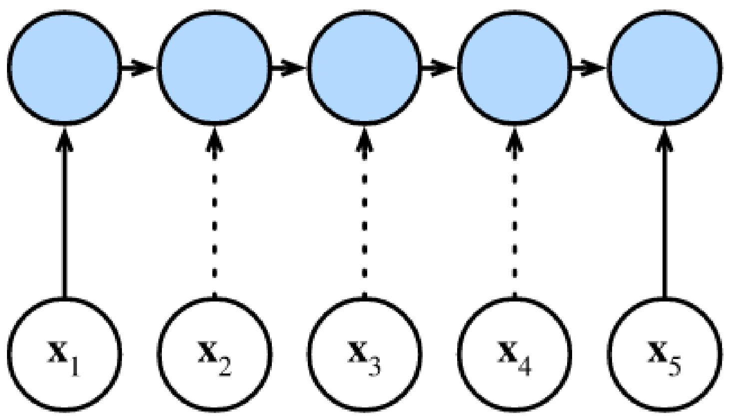
- لایه کانولوشنی با ابعاد k:
  - هزینه محاسباتی $O(knd^2)$ است
  - این محاسبات به صورت کاملا موازی قابل انجام است که معادل با $O(1)$ عملیات متوالی است
  - حداکثر طول مسیر $O(n/k)$ است
    - با استفاده از stride یا dilated convolution می‌توان حداکثر طول مسیر را کاهش داد



مقایسه توجه به خود با CNN و RNN

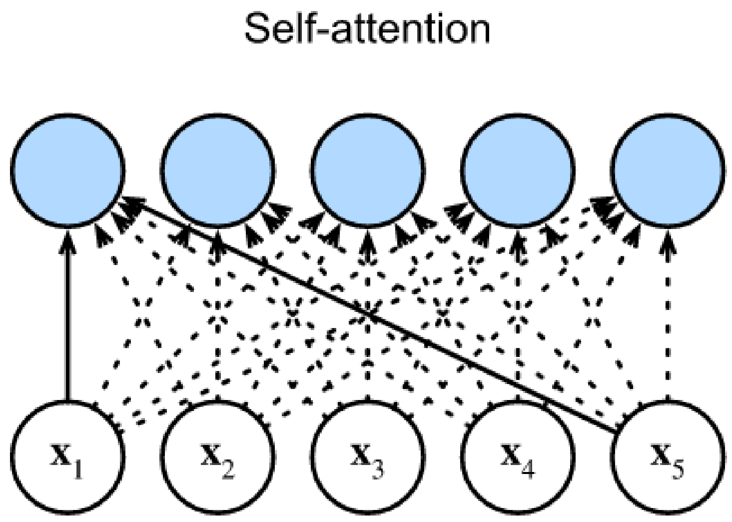
- لایه بازگشتی:
  - هزینه محاسباتی $O(nd^2)$ است
  - این محاسبات باید به صورت متوالی انجام شوند که معادل با $O(n)$ عملیات متوالی است
  - حداکثر طول مسیر $O(n)$ است

RNN



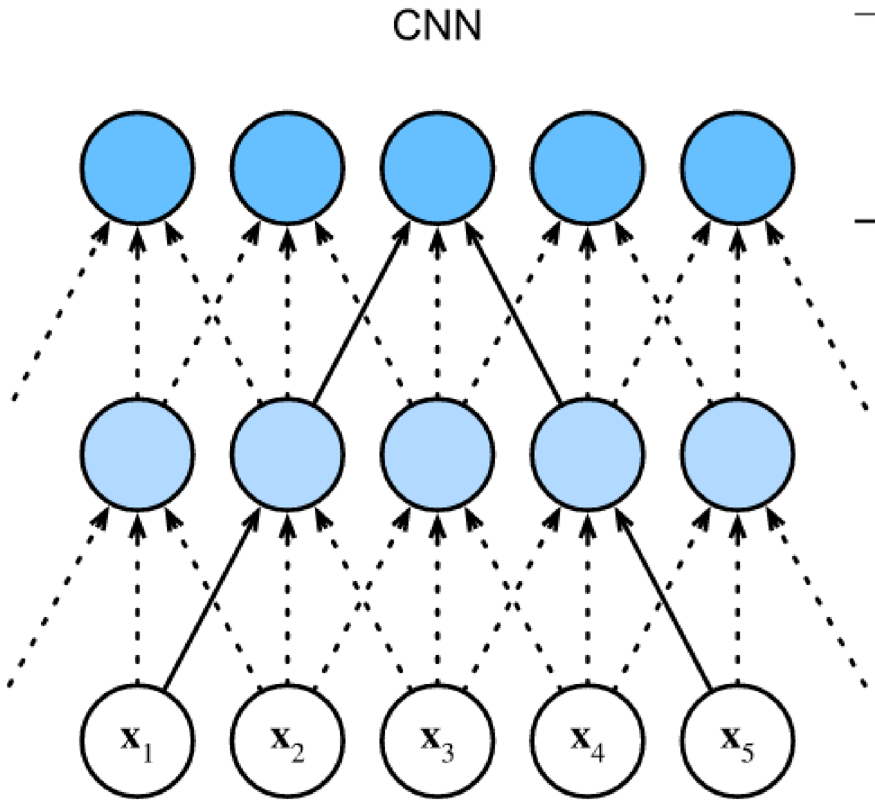
مقایسه توجه به خود با CNN و RNN

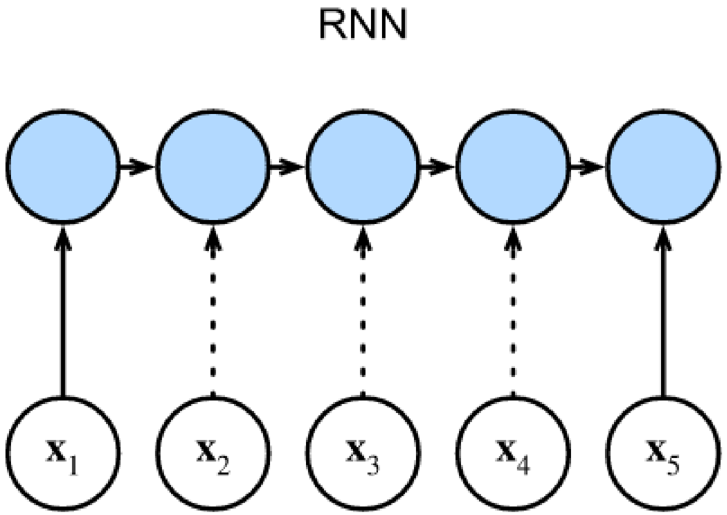
- توجه به خود:
  - هزینه محاسباتی $\mathcal{O}(n^2d)$ است
  - این محاسبات به صورت کاملا موازی قابل انجام است که معادل با $\mathcal{O}(1)$ عملیات متوالی است
  - حداکثر طول مسیر $\mathcal{O}(1)$ است

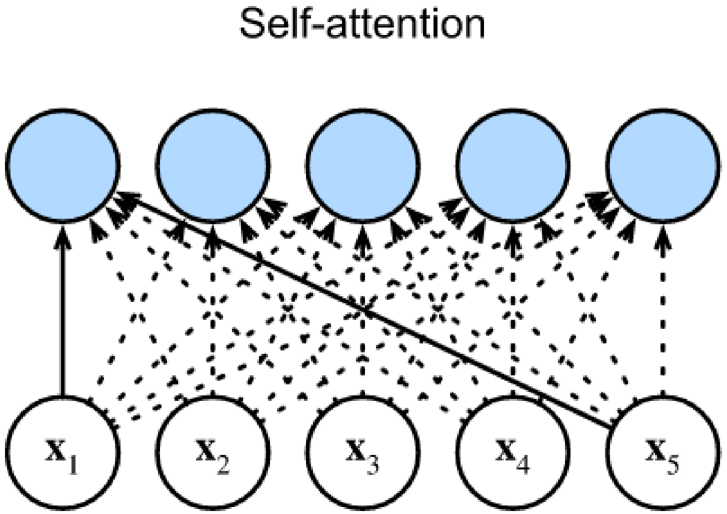


مقایسه توجه به خود با CNN و RNN

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(n/k)$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

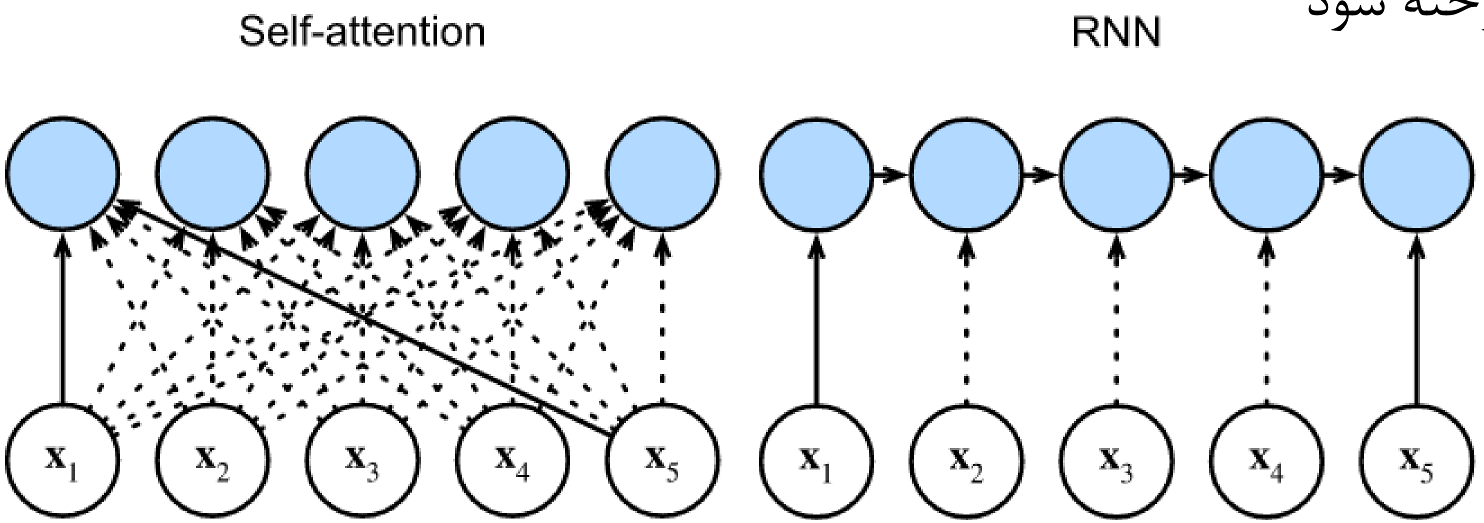






کدگذاری موقعیتی (Positional Encoding)

- برخلاف RNN که به طور متوالی توکن‌های یک دنباله را پردازش می‌کند، توجه به خود محاسبات را به صورت موازی انجام می‌دهد
  - ترتیب توکن‌ها و موقعیت نسبی آنها در محاسبات لحاظ نمی‌شود
  - می‌توانیم اطلاعات مربوط به موقعیت نسبی یا مطلق را با استفاده از کدگذاری موقعیتی به مدل تزریق کنیم
  - کدگذاری موقعیتی می‌تواند ثابت باشد یا آموخته شود
  - در ادامه یک روش ثابت را بررسی می‌کنیم



کدگذاری موقعیتی (Positional Encoding)

- فرض کنید ورودی $\mathbf{X} \in \mathbb{R}^{n \times d}$ حاوی جانمایی‌های dبعدی برای n توکن از یک دنباله باشد
- خروجی کدگذاری موقعیتی $\mathbf{X} + \mathbf{P}$ با استفاده از ماتریس جانمایی موقعیتی $\mathbf{P} \in \mathbb{R}^{n \times d}$ است
- یک ایده می‌تواند این باشد که به هر توکن یک عدد در بازه $[0,1]$ اختصاص بدهیم
  - که 0 برای کلمه اول و 1 برای کلمه آخر استفاده شود و باقی هم به صورت خطی در این بازه باشند
  - یکی از مشکلات این است که طول دنباله‌ها می‌تواند متفاوت باشد و بنابراین تعداد کلمات موجود در یک بازه مشخص قابل تشخیص نیست

کدگذاری موقعیتی (Positional Encoding)

- فرض کنید ورودی $\mathbf{X} \in \mathbb{R}^{n \times d}$ حاوی جانمایی‌های dبعدی برای n توکن از یک دنباله باشد
- خروجی کدگذاری موقعیتی $\mathbf{X} + \mathbf{P}$ با استفاده از ماتریس جانمایی موقعیتی $\mathbf{P} \in \mathbb{R}^{n \times d}$ است
- یک ایده می‌تواند این باشد که به هر توکن یک عدد در بازه $[0,1]$ اختصاص بدهیم
- یک ایده دیگر می‌تواند اختصاص اعداد صحیح به هر موقعیت باشد (1، 2، 3 و ...)
  - مقادیر می‌توانند خیلی بزرگ شوند
  - ممکن است مدل با جملاتی طولانی‌تر از آنچه در زمان آموزش دیده است مواجه شود
  - همچنین، جملات با برخی طول‌ها ممکن است اصلا در زمان آموزش موجود نباشد و تعمیم‌دهی مدل برای آنها را دچار مشکل کند

کدگذاری موقعیتی (Positional Encoding)

- فرض کنید ورودی $\mathbf{X} \in \mathbb{R}^{n\times d}$ حاوی جانمایی‌های dبعدی برای n توکن از یک دنباله باشد
- خروجی کدگذاری موقعیتی $\mathbf{X} + \mathbf{P}$ با استفاده از ماتریس جانمایی موقعیتی $\mathbf{P} \in \mathbb{R}^{n\times d}$ است
- در حالت ایده‌آل، معیارهای زیر باید برآورده شوند:
  - برای هر گام زمانی (موقعیت کلمه در یک جمله) یک کد منحصر به فرد تولید کند
  - فاصله بین هر دو گام زمانی در جملاتی با طول‌های مختلف، یکسان باشد
  - بتواند به جملات طولانی‌تر تعمیم بدهد
    - مقادیر آن باید محدود باشند
  - باید قطعی باشد

کدگذاری موقعیتی (Positional Encoding)

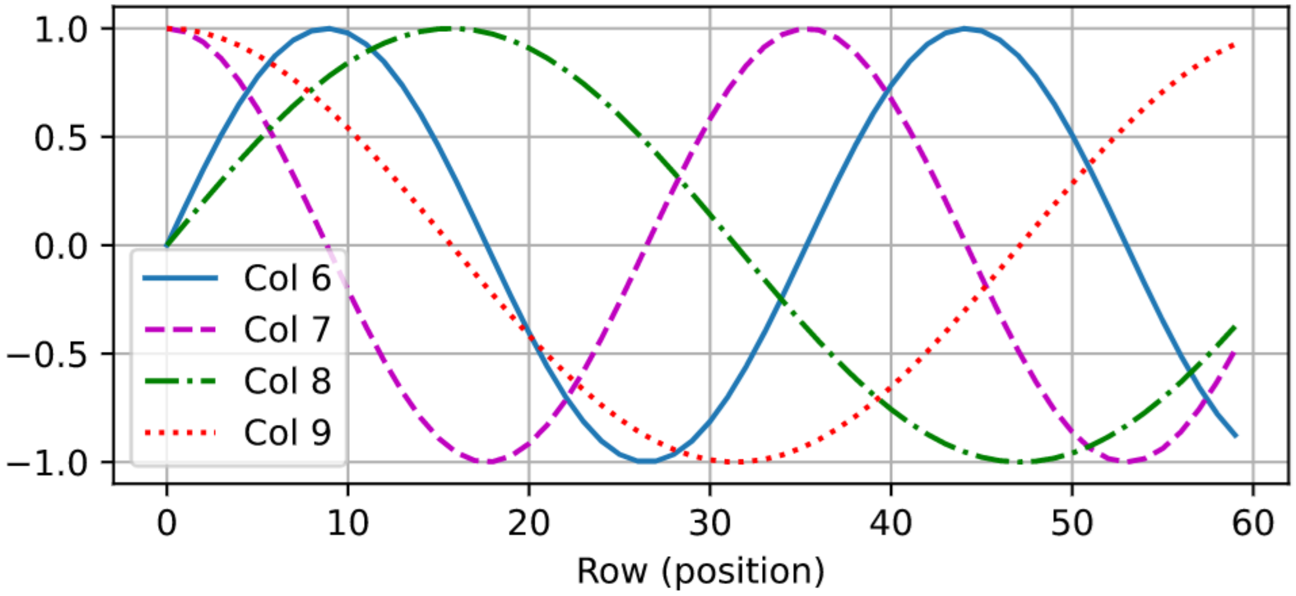
- پیشنهاد مقاله این بوده است که از یک بردار d بعدی برای هر موقعیت استفاده شود
  - این بردار برای موقعیت i به صورت زیر تعریف می‌شود
  - فرکانس ω_j با افزایش j کاهش می‌یابد و تغییرات آهسته‌تر می‌شود

$$p_{i,2j} = \sin(\omega_j i)$$

$$p_{i,2j+1} = \cos(\omega_j i)$$

$$\omega_j = \frac{1}{10000^{2j/d}}$$

$$\mathbf{p}_i = \begin{bmatrix} \sin(\omega_1 i) \\ \cos(\omega_1 i) \\ \sin(\omega_2 i) \\ \cos(\omega_2 i) \\ \vdots \\ \sin(\omega_{d/2} i) \\ \cos(\omega_{d/2} i) \end{bmatrix}$$



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*, 2017.

کدگذاری موقعیتی (Positional Encoding)

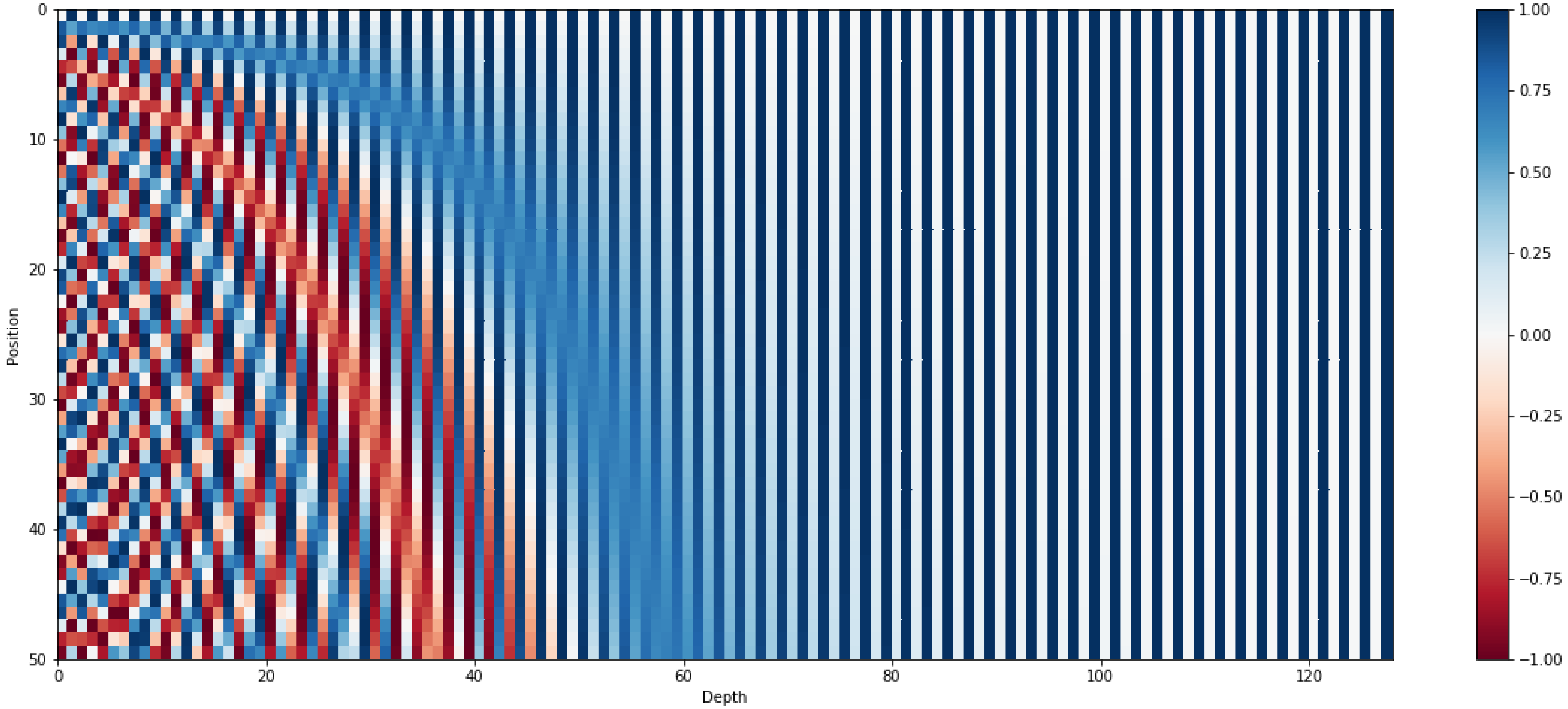
0 :	0	0	0	0
1 :	0	0	0	1
2 :	0	0	1	0
3 :	0	0	1	1
4 :	0	1	0	0
5 :	0	1	0	1
6 :	0	1	1	0
7 :	0	1	1	1
8 :	1	0	0	0
9 :	1	0	0	1
10 :	1	0	1	0
11 :	1	0	1	1
12 :	1	1	0	0
13 :	1	1	0	1
14 :	1	1	1	0
15 :	1	1	1	1

- برای شهود بهتر، نمایش باینری اعداد صحیح را در نظر بگیرید
  - بیت کم‌ارزش با نرخ بالایی تغییر می‌کند (در هر موقعیت تغییر می‌کند)
  - بیت‌های با ارزش بالاتر فرکانس کمتری دارند
  - زمانیکه حافظه و محاسبات اعشاری هستند، استفاده از اعداد باینری بهینه نیست
    - استفاده از توابع سینوسی مناسب‌تر است!

$$\mathbf{p}_i = \begin{bmatrix} \sin(\omega_1 i) \\ \cos(\omega_1 i) \\ \sin(\omega_2 i) \\ \cos(\omega_2 i) \\ \vdots \\ \sin(\omega_{d/2} i) \\ \cos(\omega_{d/2} i) \end{bmatrix}$$

کدگذاری موقعیتی (Positional Encoding)

$$\mathbf{p}_i = \begin{bmatrix} \sin(\omega_1 i) \\ \cos(\omega_1 i) \\ \sin(\omega_2 i) \\ \cos(\omega_2 i) \\ \vdots \\ \sin(\omega_{d/2} i) \\ \cos(\omega_{d/2} i) \end{bmatrix}$$



https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

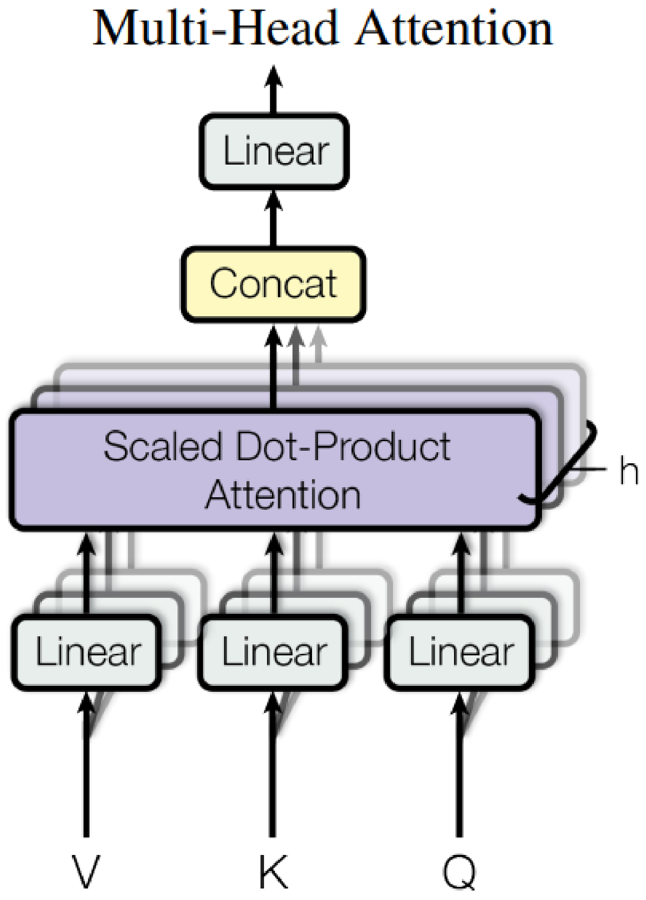
اطلاعات موقعیتی نسبی

- علاوه بر اطلاعات موقعیتی مطلق، این روش به مدل اجازه می‌دهد تا به راحتی توجه به موقعیت‌های نسبی را یاد بگیرد
  - برای هر آفست موقعیت ثابت δ، کدگذاری موقعیتی در موقعیت $i+\delta$ را می‌توان با یک نگاشت خطی از موقعیت i بدست آورد
  - این ماتریس تبدیل مستقل از موقعیت i است

$$\mathbf{p}_i = \begin{bmatrix} \sin(\omega_1 i) \\ \cos(\omega_1 i) \\ \sin(\omega_2 i) \\ \cos(\omega_2 i) \\ \vdots \\ \sin(\omega_{d/2} i) \\ \cos(\omega_{d/2} i) \end{bmatrix}$$

$$\begin{aligned}
&\begin{bmatrix} \cos(\delta\omega_j) & \sin(\delta\omega_j) \\ -\sin(\delta\omega_j) & \cos(\delta\omega_j) \end{bmatrix} \begin{bmatrix} p_{i,2j} \\ p_{i,2j+1} \end{bmatrix} \\
&= \begin{bmatrix} \cos(\delta\omega_j)\sin(i\omega_j) + \sin(\delta\omega_j)\cos(i\omega_j) \\ -\sin(\delta\omega_j)\sin(i\omega_j) + \cos(\delta\omega_j)\cos(i\omega_j) \end{bmatrix} \\
&= \begin{bmatrix} \sin\left((i+\delta)\omega_j\right) \\ \cos\left((i+\delta)\omega_j\right) \end{bmatrix} = \begin{bmatrix} p_{i+\delta,2j} \\ p_{i+\delta,2j+1} \end{bmatrix}
\end{aligned}$$

خلاصه



- توجه چندسر
- در توجه به خود، queries، keys و values یکسان هستند
- توجه به خود و CNNها از محاسبات موازی لذت می‌برند
- توجه به خود کوتاه‌ترین طول مسیر را دارد
- پیچیدگی محاسباتی درجه دوم با توجه به طول دنباله باعث می‌شود که توجه به خود برای دنباله‌های بسیار طولانی بسیار کند شود
- برای استفاده از اطلاعات ترتیب دنباله، می‌توانیم اطلاعات موقعیتی مطلق یا نسبی را با افزودن کدگذاری موقعیتی به ورودی تزریق کنیم