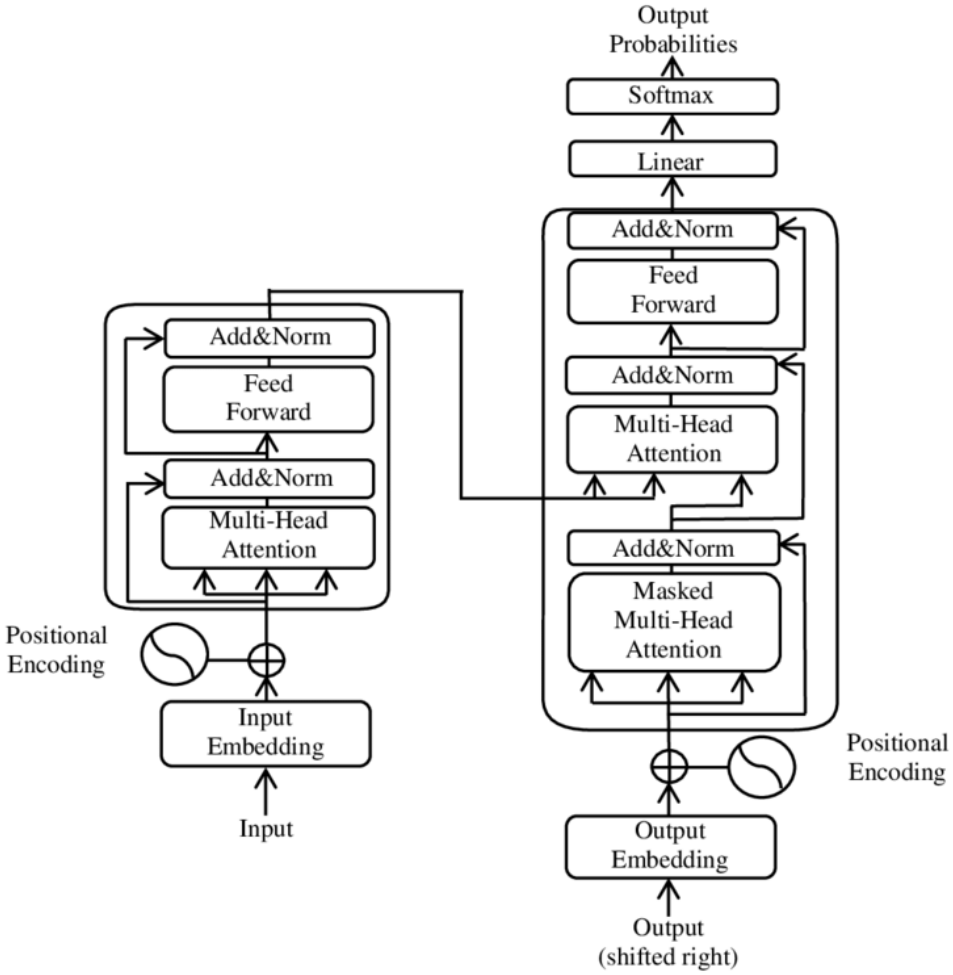
Lecture 14: Transformers

Mark Hasegawa-Johnson



Outline

- Text generation
- Attention
- Multi-headed attention
- Positional embedding
- Layer norm
- Residual connections

Text generation

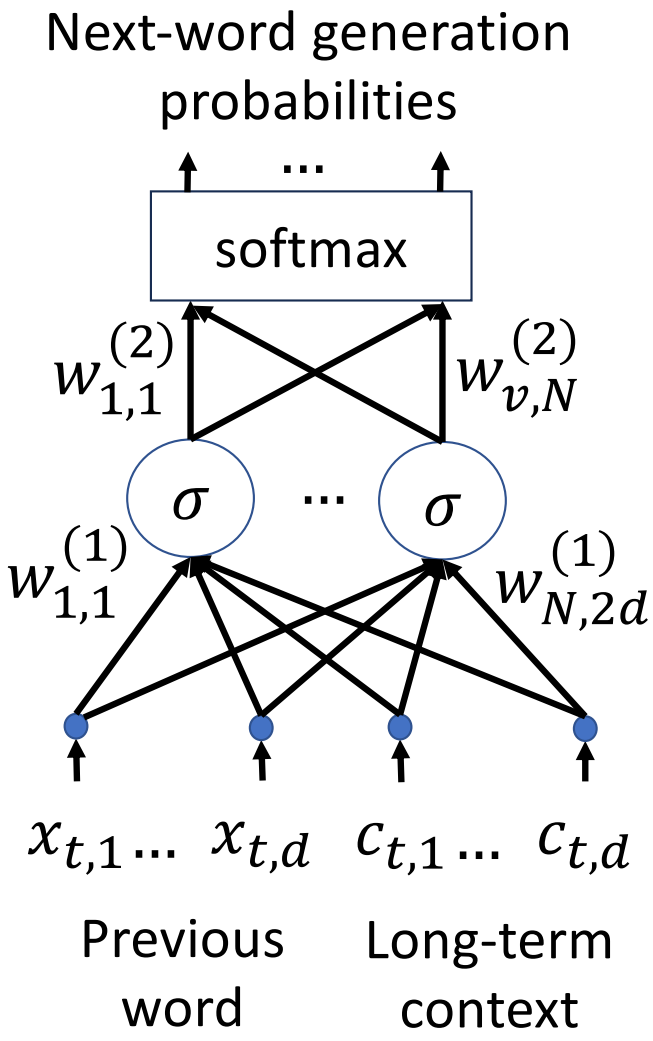
How can a neural net generate text?

- Previous word = $\boldsymbol{x}_t$, encode as a vector
- Long-term context = $\boldsymbol{c}_t$, encode as a vector
- Next word is $O_{t+1} = y$ with probability

$$P(O_{t+1} = y) = \frac{\exp(z_y)}{\sum_k \exp(z_k)}$$

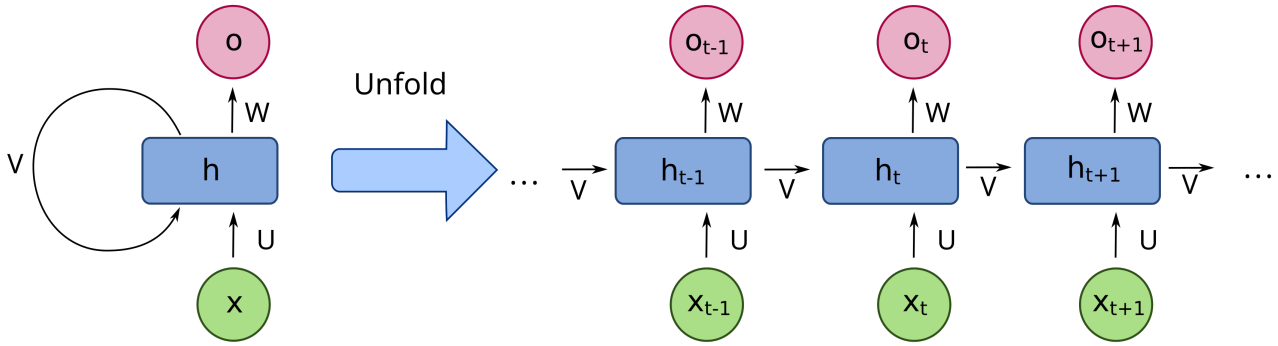
Where z_k is kth output of a feedforward net,

$$\boldsymbol{z} = \boldsymbol{W}^{(2)}\sigma\left(\boldsymbol{W}^{(1)}[\boldsymbol{x}_t; \boldsymbol{c}_t]\right)$$



Generating text without context: Bigram RNN

- Without the context vector, $P(O_{t+1})$ only depends on $\boldsymbol{x}_t$
- As a statistical model, it's called a "Bigram language model"
- As a neural net, that's called a "Recurrent neural network"



By fdeloche - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=60109157

Generating text with context

There are many ways to include long-term context:

- We can expand the hidden state vector so it's larger than the input
- We can use special architectures: LSTM, MAMBA, other special architectures that remember long-term context

Today we focus on the Transformer

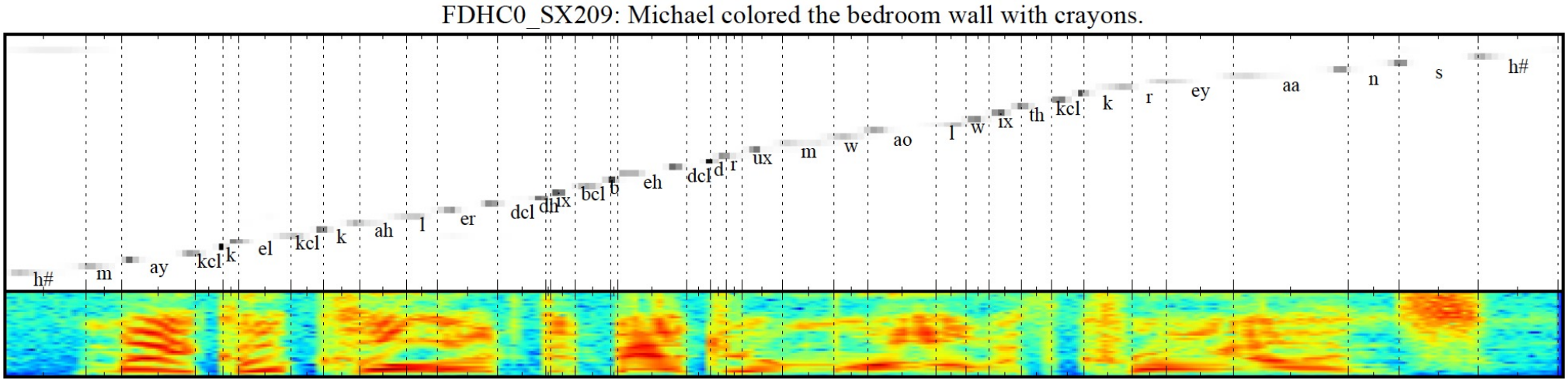
- Attention = a special architecture that creates a weighted summary of long-term context
- Layer norm = a type of normalization to make sure attention works
- Residual connections = each layer is a small change

Outline

- Text generation
- Attention
- Multi-headed attention
- Positional embedding
- Layer norm
- Residual connections

Summarizing short-term memory

- Suppose the most recent 1024 words in the conversation are stored in a “short-term memory buffer”
- To compute the i[th] output, we need context vector $\boldsymbol{c}_i = \text{summary}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{1024})$
- How can we make that summary?



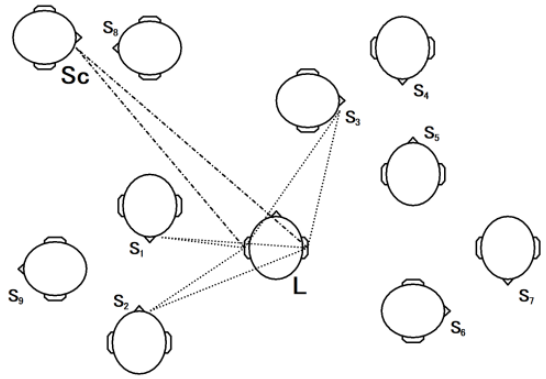
Chorowski, Bahdanau, Serdyk, Cho & Bengio, Attention-Based Models for Speech Recognition, Fig. 1

The Cocktail-Party Effect

- If you are focusing on one person's voice, but hear your name spoken by another person, your attention immediately shifts to the second voice.
- This "cocktail-party effect" suggests a model of hearing in which all sounds are processed preconsciously. Trigger sounds in an unattended source will cause attention to re-orient to that source.

https://commons.wikimedia.org/wiki/File:Cocktail_party_attendees_at_Fuller_Lodge,_1946.jpg

https://commons.wikimedia.org/wiki/File:Cocktail_Party_At_The_Imperial_Hotel_March_13,_1961_(Tokyo,_Japan)_(4966610682).jpg



https://commons.wikimedia.org/wiki/File:Cocktail-party_effect.svg

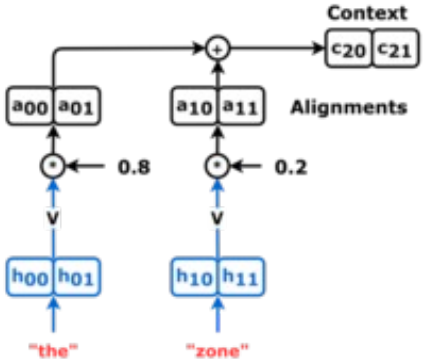
Attention is like probability

- Attention is like probability: You only have a fixed amount of attention, so you need to decide how to distribute it.
- The context vector ($\boldsymbol{c}_i$) summarizes input vectors ($\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{1024}$) using:

$$\boldsymbol{c}_i = \sum_t \alpha_{i,t} \boldsymbol{x}_t$$

- where $\alpha_{i,t}$= importance of $\boldsymbol{x}_t$ for $\boldsymbol{c}_i$:

$$\sum_t \alpha_{i,t} = 1$$

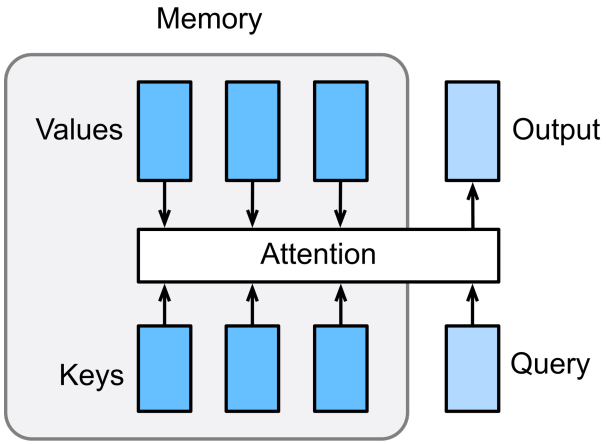


https://commons.wikimedia.org/wiki/File:Encoder_cross-attention,_computing_the_context_vector_by_a_linear_sum.png

What is the "importance of $\boldsymbol{x}_t$ for $\boldsymbol{c}_i$"?

Let's break the problem down into smaller pieces:

- "Value vector" $\boldsymbol{v}_t$ = information from $\boldsymbol{x}_t$ that will be used in $\boldsymbol{c}_i$
- "Key vector" $\boldsymbol{k}_t$ = describes why $\boldsymbol{v}_t$ might be useful
- "Query vector" $\boldsymbol{q}_i$ = describes what we're looking for

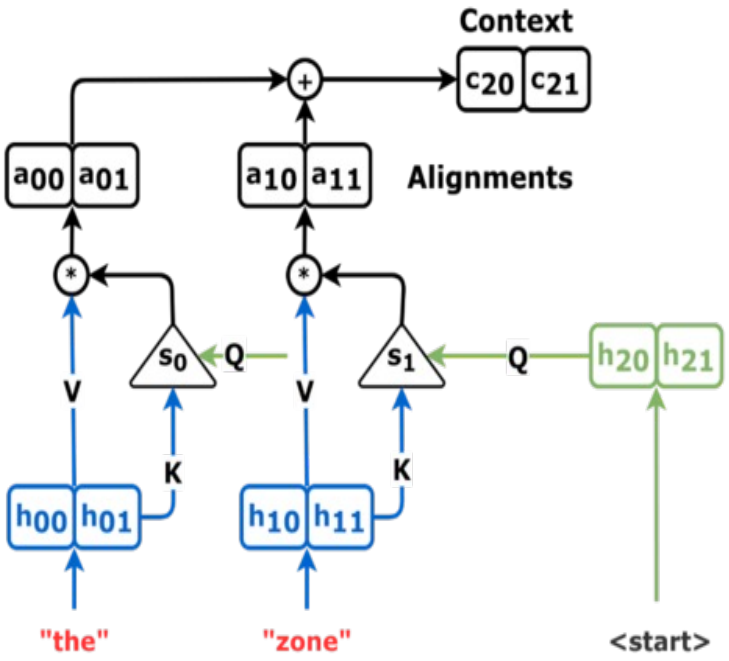


By Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J. - https://github.com/d2l-ai/d2l-en, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=152265713

What is the "importance of $\boldsymbol{x}_t$ for $\boldsymbol{c}_i$"?

Let's break the problem down into smaller pieces:

- "Value vector" $\boldsymbol{v}_t$ = information from $\boldsymbol{x}_t$ that will be used in $\boldsymbol{c}_i$
- "Key vector" $\boldsymbol{k}_t$ = describes why $\boldsymbol{v}_t$ might be useful
- "Query vector" $\boldsymbol{q}_i$ = describes what we're looking for

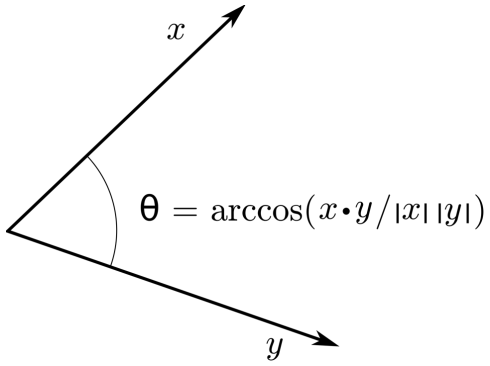


Dot-product attention

How can you decide which value vectors, v_t are most relevant to a particular query? Answer:

1. Create a key vector, $\boldsymbol{k}_t$, such that $\boldsymbol{q}_i^T \boldsymbol{k}_t > 0$ if $\boldsymbol{v}_t$ is relevant to $\boldsymbol{q}_i$, otherwise $\boldsymbol{q}_i^T \boldsymbol{k}_t < 0$.
2. Assume that each the key is roughly zero-mean, unit-variance, so $|\boldsymbol{k}_t|^2 \approx d$, the length of the vector. Assume that the query is unit-length, $|\boldsymbol{q}_i|^2 \approx 1$ More later about why these are reasonable assumptions.
3. On that assumption,

$$\cos(\boldsymbol{k}_t, \boldsymbol{q}_i) \approx \boldsymbol{q}_i^T \boldsymbol{k}_t / \sqrt{d}$$



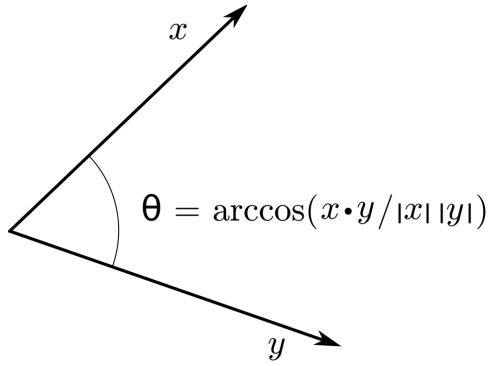
Dot-product attention

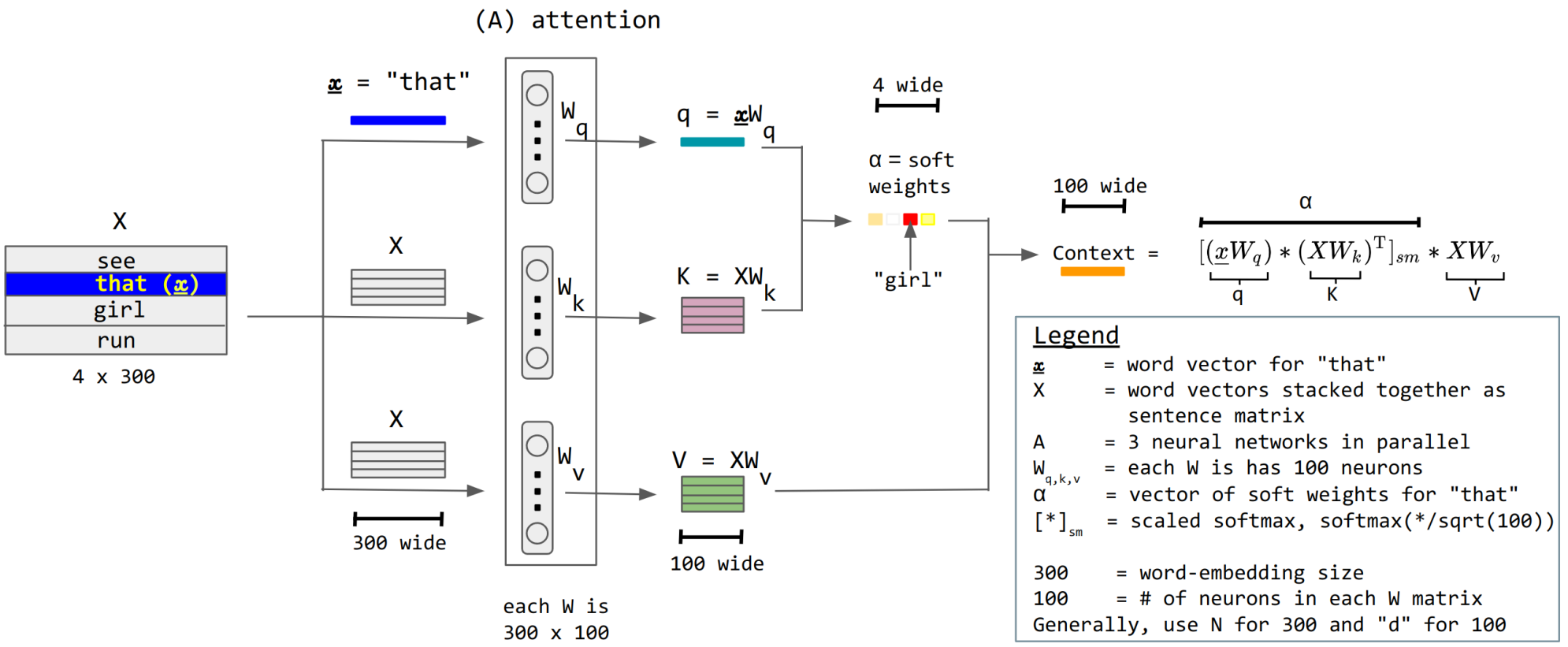
Similarity between the query and the key is:

$$\cos(\boldsymbol{k}_t, \boldsymbol{q}_i) \approx \boldsymbol{q}_i^T \boldsymbol{k}_t / \sqrt{d}$$

Convert the similarity measures into a probability-like quantity (called attention) using softmax:

$$\alpha_{i,t} = \frac{\exp\left(\boldsymbol{q}_i^T \boldsymbol{k}_t / \sqrt{d}\right)}{\sum_\tau \exp\left(\boldsymbol{q}_i^T \boldsymbol{k}_\tau / \sqrt{d}\right)}$$





Attention

- Stack the input vectors in a matrix, $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^T$
- Multiply by $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ to create query, key, and value vectors

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q, \qquad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K, \qquad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V$$

- Dot-product similarity:

$$\boldsymbol{E} = \boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d} = \frac{1}{\sqrt{d}}\begin{bmatrix} \boldsymbol{q}_1^T\boldsymbol{k}_1 & \cdots & \boldsymbol{q}_1^T\boldsymbol{k}_n \\ \vdots & \ddots & \vdots \\ \boldsymbol{q}_n^T\boldsymbol{k}_1 & \cdots & \boldsymbol{q}_n^T\boldsymbol{k}_n \end{bmatrix}$$

- Attention

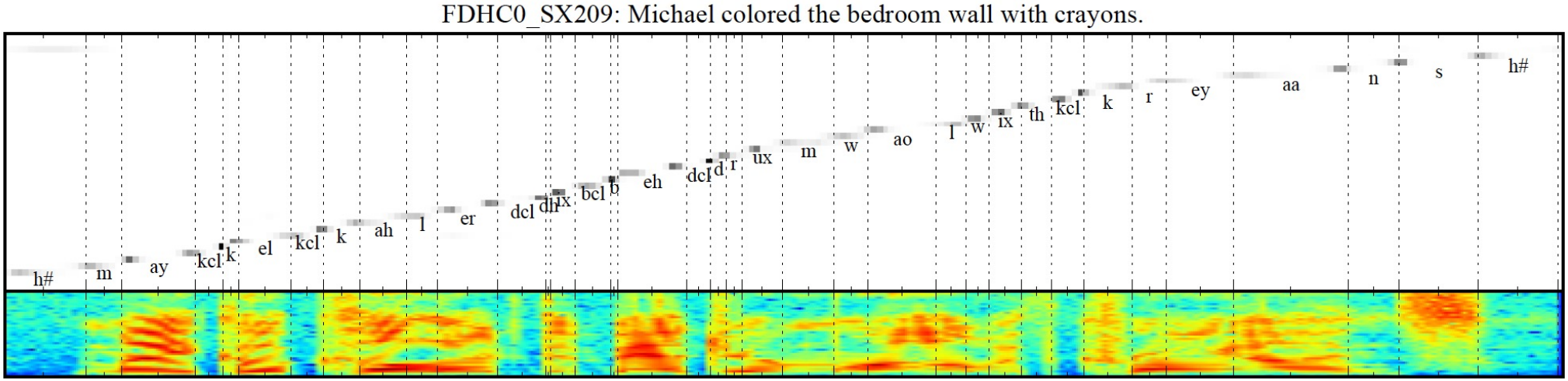
$$\boldsymbol{A} = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,n} \\ \vdots & \ddots & \vdots \\ \alpha_{n,1} & \cdots & \alpha_{n,n} \end{bmatrix} = \mathrm{softmax}(\boldsymbol{E}), \qquad \alpha_{i,t} = \frac{\exp\left(\boldsymbol{q}_i^T\boldsymbol{k}_t/\sqrt{d}\right)}{\sum_\tau \exp\left(\boldsymbol{q}_i^T\boldsymbol{k}_\tau/\sqrt{d}\right)}$$

- Context vectors are the dot product of attention times value:

$$\boldsymbol{C} = [\boldsymbol{c}_1, \dots, \boldsymbol{c}_n]^T = \boldsymbol{A}\boldsymbol{V}, \qquad \boldsymbol{c}_i = \sum_t \alpha_{i,t}\boldsymbol{v}_t$$

Cross-attention visualization

This plot shows $\alpha_{i,t}$ where i = output character, and t = input spectrum



Chorowski, Bahdanau, Serdyk, Cho & Bengio, Attention-Based Models for Speech Recognition, Fig. 1

Quiz!

Go to prairielearn and try the quiz!

Outline

- Text generation
- Attention
- Multi-headed attention
- Positional embedding
- Layer norm
- Residual connections

What is the "importance of $\boldsymbol{x}_t$ for $\boldsymbol{c}_i$"?

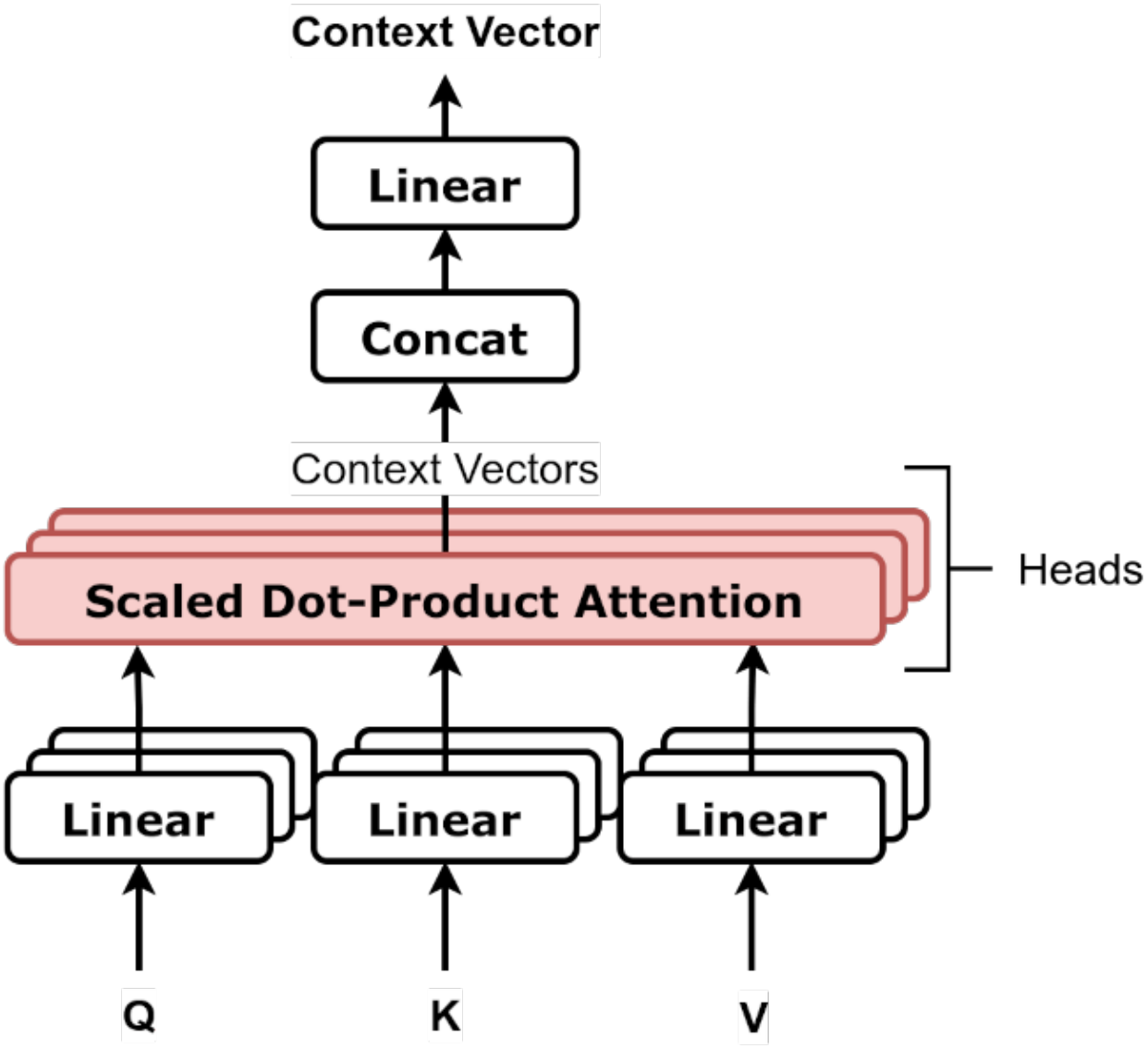
Let's break the problem down into smaller pieces:

- "Value vector" $\boldsymbol{v}_t$ = information from $\boldsymbol{x}_t$ that will be used in $\boldsymbol{c}_i$
- "Key vector" $\boldsymbol{k}_t$ = describes why $\boldsymbol{v}_t$ might be useful
- "Query vector" $\boldsymbol{q}_i$ = describes what we're looking for

…but what if we need different types of context information?

- The general topic of the discussion
- The most recent noun that the human mentioned
- Is next word part of subject, direct object, indirect object, or other?

Multi-headed attention



By dvgodoy - https://github.com/dvgodoy/dl-visuals/?tab=readme-ov-file, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=151216029

Multi-Headed Attention

- In the j[th] head, multiply by $\boldsymbol{W}_{j,Q}$, $\boldsymbol{W}_{j,K}$, and $\boldsymbol{W}_{j,V}$:

$$\boldsymbol{Q}_j = \boldsymbol{X}\boldsymbol{W}_{j,Q}, \qquad \boldsymbol{K}_j = \boldsymbol{X}\boldsymbol{W}_{j,K}, \qquad \boldsymbol{V}_j = \boldsymbol{X}\boldsymbol{W}_{j,V}$$

- Compute context vectors using dot-product attention:

$$\boldsymbol{C}_j = \text{softmax}\left(\frac{\boldsymbol{Q}_j \boldsymbol{K}_j^T}{\sqrt{d}}\right)\boldsymbol{V}_j$$

- Concatenate all heads, and linearly transform one more time:

$$\boldsymbol{O} = [\boldsymbol{C}_1, \ldots, \boldsymbol{C}_h]\boldsymbol{W}_O$$

Multi-headed attention gives multiple contexts

- The general topic of the discussion
- The most recent noun that the human mentioned
- Is next word part of subject, direct object, indirect object, or other?
- What is the head word of the current phrase?
- What is the word that started this sentence?
- What is the most recent word?

…. wait …. those last two need information about time. Where do we get that information?

Outline

- Text generation
- Attention
- Multi-headed attention
- Positional embedding
- Layer norm
- Residual connections

What we have lost…

- With a transformer, each state vector pays attention to the input that is most similar, regardless of what time it happened:

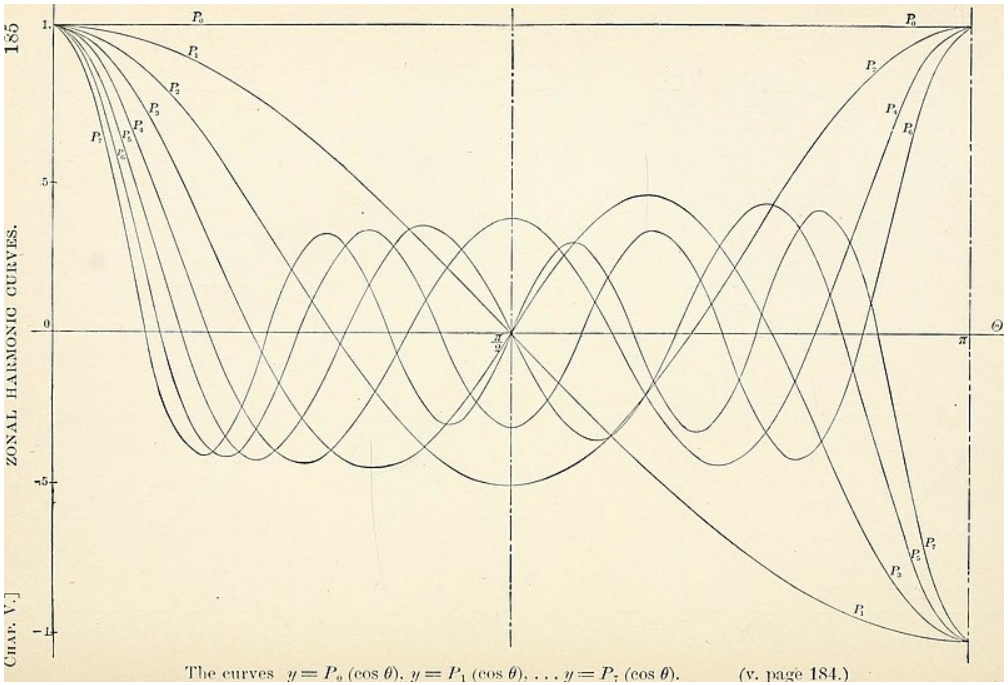
$$\boldsymbol{c}_i = \sum_t \alpha_{i,t}\, \boldsymbol{v}_t, \qquad \alpha_{i,t} = \frac{\exp\left(\boldsymbol{q}_i^T \boldsymbol{k}_t\right)}{\sum_\tau \exp\left(\boldsymbol{q}_i^T \boldsymbol{k}_\tau\right)}$$

- What if we always want $\boldsymbol{h}_t$ to pay special attention to $\boldsymbol{v}_{t-1}$? Is that possible?

Rotary Positional Embedding (RoPE) = Fourier series

The solution is to encode the relative position of each input, $\boldsymbol{v}_t$, using a Fourier basis $\boldsymbol{e}_t$:

$$\boldsymbol{e}_t = \begin{bmatrix} \cos\left(\frac{\pi t}{T}\right) \\ \sin\left(\frac{\pi t}{T}\right) \\ \vdots \\ \cos\left(\frac{\pi D t}{2T}\right) \\ \sin\left(\frac{\pi D t}{2T}\right) \end{bmatrix}$$



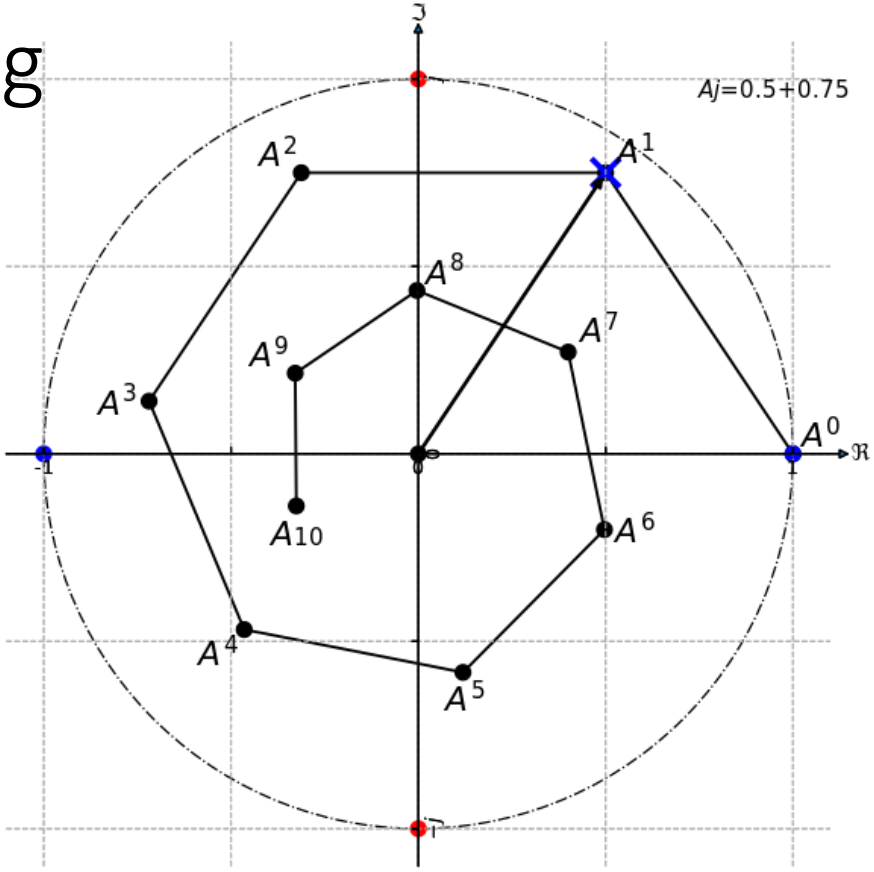
Public domain image, https://commons.wikimedia.org/wiki/File:An_elementary_treatise_on_Fourier%27s_series_and_spherical,_cylindrical,_and_ellipsoidal_harmonics,_with_applications_to_problems_in_mathematical_physics_(1893)_(14780364665).jpg

Rotary positional embedding

The Fourier basis is useful because shifting by a fixed time offset, to $t - \tau$, can be accomplished by a matrix multiplication:

$$\boldsymbol{e}_{t-\tau} = \begin{bmatrix} \cos\left(\frac{\pi\tau}{T}\right) & \sin\left(\frac{\pi\tau}{T}\right) & \cdots \\ -\sin\left(\frac{\pi\tau}{T}\right) & \cos\left(\frac{\pi\tau}{T}\right) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \boldsymbol{e}_t$$

…so if we want a particular query to pay attention to vectors with a time delay of τ, we just set $\boldsymbol{W}_{j,Q}$ to the matrix shown above.



Public domain image, https://commons.wikimedia.org/wiki/File:Exponentials_of_complex_number_within_unit_circle-2.svg

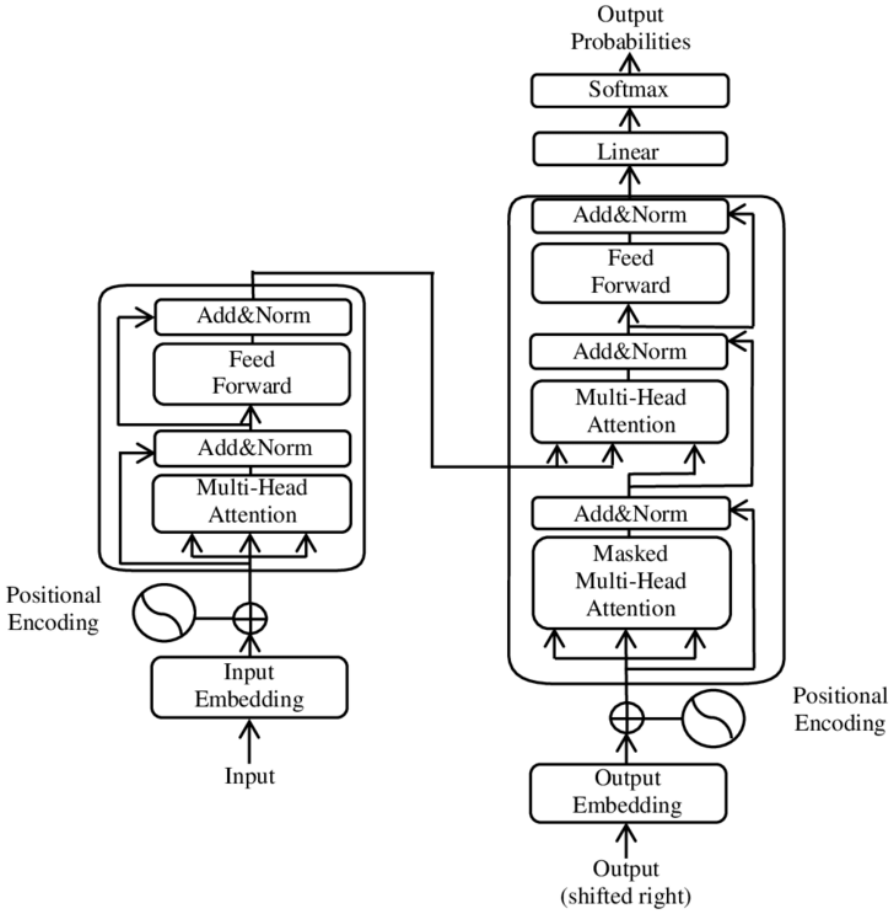
Where do we put the positional encoding?

- Possibility #1: Concatenate it, i.e., $\boldsymbol{x}_t^T = [\boldsymbol{x}_t^T, \boldsymbol{e}_t^T]$
  - Advantage: $\boldsymbol{W}_{j,Q}$ can learn to operate separately on the content $\boldsymbol{x}_t^T$ and the positional encoding $\boldsymbol{e}_t^T$
  - Disadvantage: every vector is twice as large, and every matrix is four times as large
- Possibility #2: Add it, i.e., $\boldsymbol{x}_t = \boldsymbol{x}_t + \boldsymbol{e}_t$
  - Advantage: fewer parameters to learn
  - Disadvantage: $\boldsymbol{W}_{j,Q}$ can only operate directly on $\boldsymbol{e}_t$ if $\boldsymbol{x}_t$ is mostly zero
  - Surprise: this works well in practice. Apparently, the positional encoding can learn to ignore local fluctuations in $\boldsymbol{x}_t$, and pretend that it's mostly 0 on average

Positional encoding

In the standard transformer, position of the input is encoded using

$$\boldsymbol{x}_t = \boldsymbol{x}_t + \begin{bmatrix} \cos\left(\frac{\pi t}{T}\right) \\ \sin\left(\frac{\pi t}{T}\right) \\ \vdots \\ \cos\left(\frac{\pi D t}{2T}\right) \\ \sin\left(\frac{\pi D t}{2T}\right) \end{bmatrix}$$



Outline

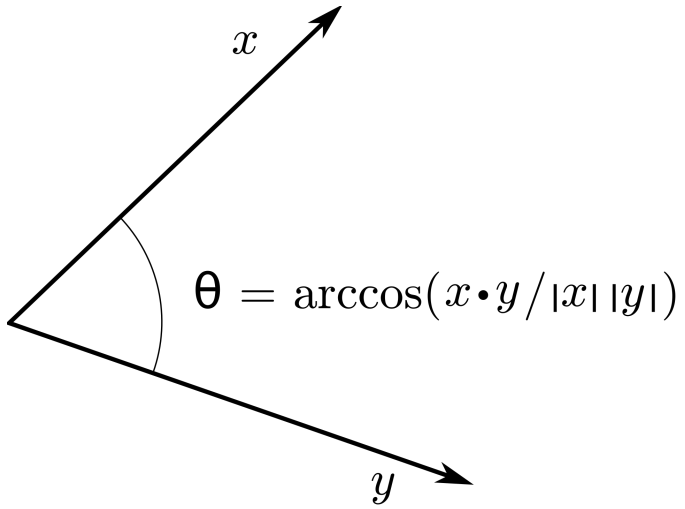
- Text generation
- Attention
- Multi-headed attention
- Positional embedding
- Layer norm
- Residual connections

Dot product = cosine if vectors are unit-norm

$\boldsymbol{q}_i^T \boldsymbol{k}_t = \cos(\boldsymbol{q}_i, \boldsymbol{k}_t)$ iff $|\boldsymbol{q}_i| = 1$ and $|\boldsymbol{k}_t| = 1$.

Incorrect normalization is sometimes useful, e.g.: to include $\boldsymbol{v}_t$ regardless of the query, you can make $|\boldsymbol{k}_t|$ large.

Often, though, allowing very large or very small vectors has bad effects.



How can we normalize the vectors?

- Normalization: subtract the mean, divide by standard deviation
- Batch normalization: Each neuron is normalized over the batch
- Instance normalization: Each instance is normalized over neurons
- Layer normalization: Normalize over the whole input

How can we normalize the vectors?

$$\boldsymbol{O} = \begin{bmatrix} o_{1,1} & \cdots & o_{1,d} \\ \vdots & \ddots & \vdots \\ o_{n,1} & \cdots & o_{n,d} \end{bmatrix}$$

- Normalization: subtract the mean, divide by standard deviation

$$o_{i,k} \leftarrow \frac{o_{i,k} - \mu}{\sigma}$$

- Batch normalization: Each neuron is normalized over the batch
- Instance normalization: Each instance is normalized over neurons
- Layer normalization: Normalize over the whole input

How can we normalize the vectors?

- Batch normalization: Each neuron is normalized over the batch

$$\mu = \frac{1}{n}\sum_{i=1}^{n} o_{i,k}\,, \qquad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(o_{i,k}-\mu\right)^2}$$

- Instance normalization: Each instance is normalized over neurons

$$\mu = \frac{1}{d}\sum_{k=1}^{d} o_{i,k}\,, \qquad \sigma = \sqrt{\frac{1}{d}\sum_{k=1}^{d}\left(o_{i,k}-\mu\right)^2}$$

- Layer normalization: Normalize over the whole input

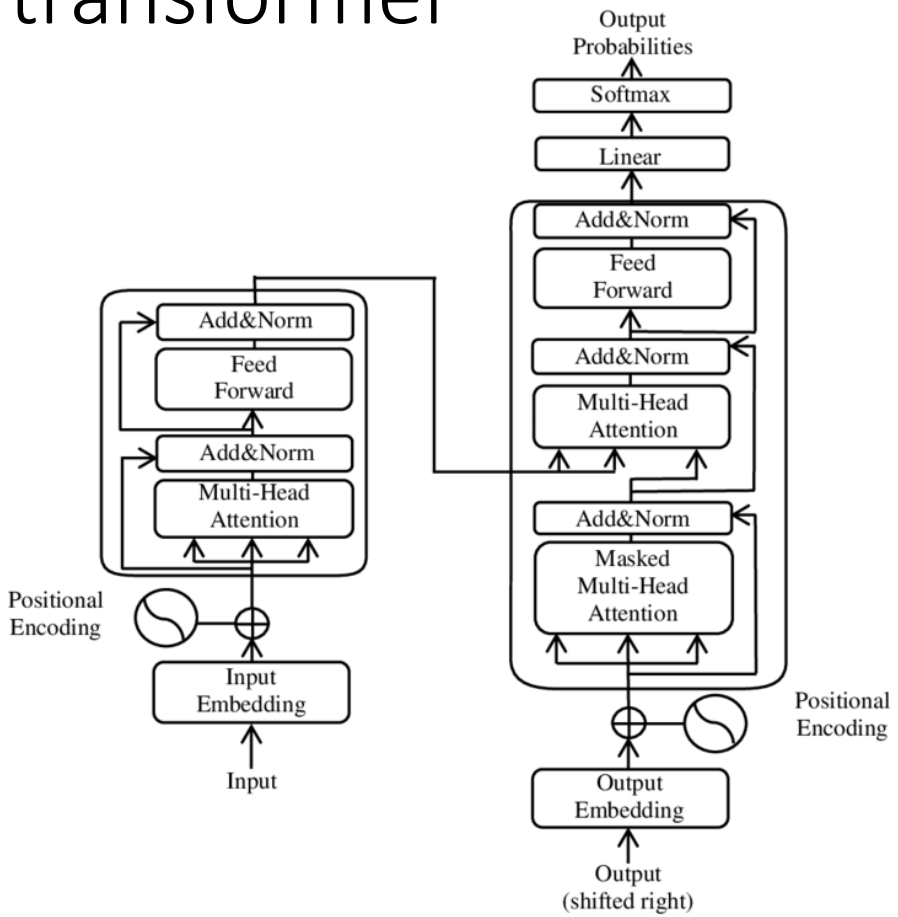
$$\mu = \frac{1}{nd}\sum_{i=1}^{n}\sum_{k=1}^{d} o_{i,k}\,, \qquad \sigma = \sqrt{\frac{1}{nd}\sum_{i=1}^{n}\sum_{k=1}^{d}\left(o_{i,k}-\mu\right)^2}$$

How can we normalize the vectors?

- Batch normalization: $E\left[o_{i,k}\right] = 0$ and $E\left[o_{i,k}^2\right] = 1$ for each k, separately, over the whole batch
- Instance normalization: Each vector $\boldsymbol{o}_i = \left[o_{i,1}, \ldots, o_{i,d}\right]$ is guaranteed to have a constant norm, $|\boldsymbol{o}_i|^2 = d$
- Layer normalization: Each vector has a constant norm on average, i.e., $E[|\boldsymbol{o}_i|^2] = d$
- If $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ are also pretty well normalized, then $E[|\boldsymbol{k}_t|^2] \approx E[|\boldsymbol{v}_t|^2] \approx d$
- The query vector often comes from a different source (the “query”), so it’s reasonable to assume that it’s normalized differently, $E[|\boldsymbol{q}_i|^2] \approx 1$
- Hence $\cos(\boldsymbol{k}_t, \boldsymbol{q}_i) \approx \boldsymbol{q}_i^T \boldsymbol{k}_t / \sqrt{d}$

Layer norm in the standard transformer

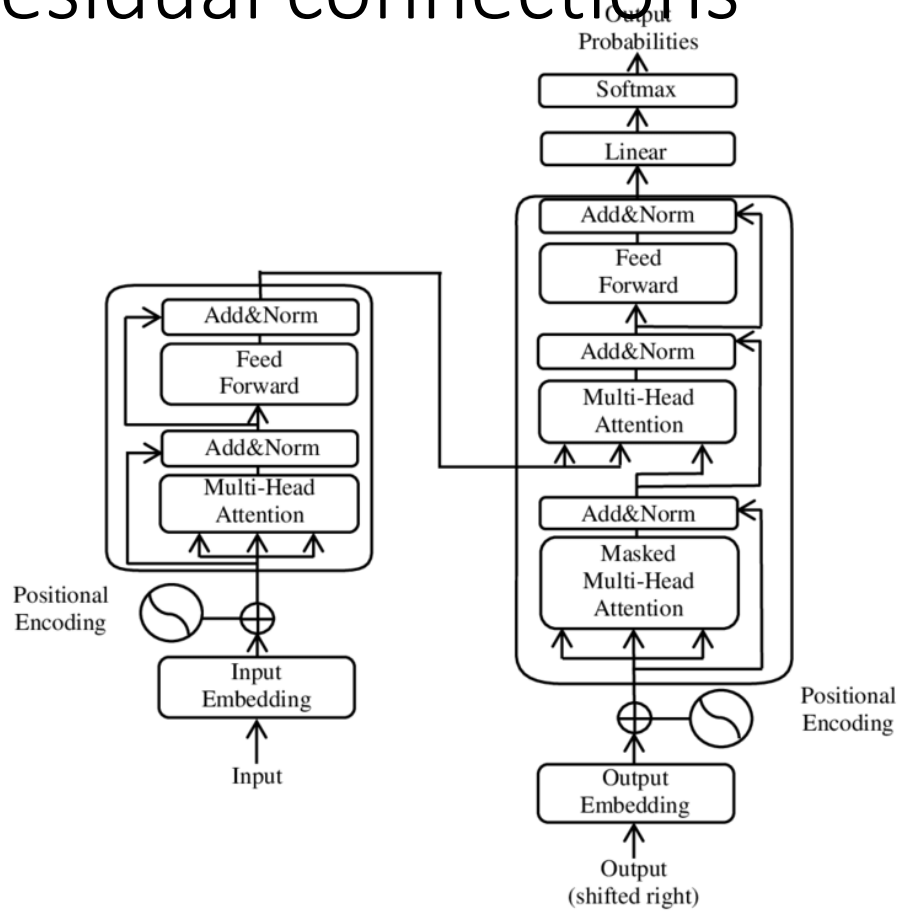
- In the “add & norm” layers, the “norm” is usually a layer norm
- What is the “add”?



Outline

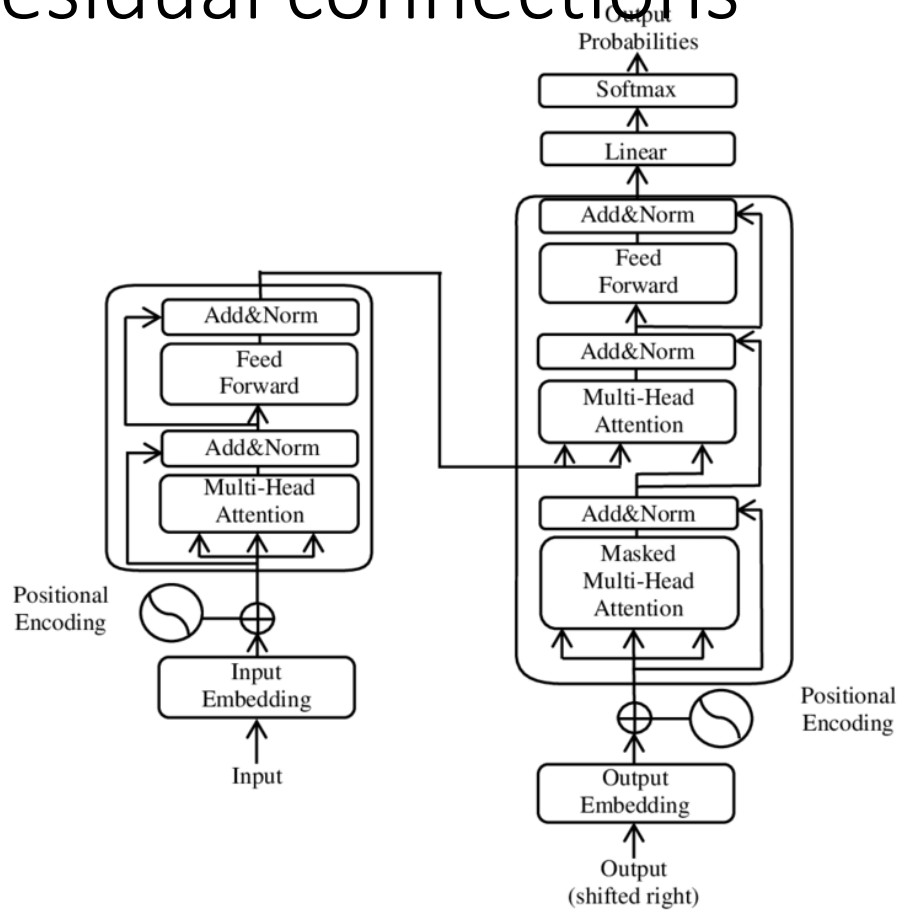
- Text generation
- Attention
- Multi-headed attention
- Positional embedding
- Layer norm
- Residual connections

Residual connections



- Most transformers have multiple layers
- For example, this diagram shows:
  - 2 layers of self-attention in the encoder (self-attention: Q, K, and V all come from the same input)
  - 1 self-attention layer + 1 cross-attention layer in the decoder (cross-attention: Q is from the previous output word, K and V are from the input words)
- More typical would be 12 layers in the encoder and 24 layers in the decoder

Residual connections



- In a residual connection, the input to the l^{th} layer, $\boldsymbol{X}^{(l)}$, is

$$\boldsymbol{X}^{(l)} = \boldsymbol{X}^{(l-1)} + \boldsymbol{O}^{(l-1)}$$

- Thus the output of each layer is only a small modification of its input (a "residual")
- The output of the whole 24-layer transformer is just the input words, plus residuals computed by all layers
  - Residuals can be small, if the input words are all that you need
  - Residuals can be large if the input words are not enough

The standard transformer neural net

- Multi-headed dot-product attention: $\boldsymbol{C}_j = \text{softmax}\left(\frac{\boldsymbol{Q}_j \boldsymbol{K}_j^T}{\sqrt{d}}\right)\boldsymbol{V}_j$
- Concatenate and transform: $\boldsymbol{O} = [\boldsymbol{C}_1, \ldots, \boldsymbol{C}_h]\boldsymbol{W}_O$
- Rotary Positional Encoding (RoPE):
$$\boldsymbol{x}_t \mathrel{+}= [\cos(\pi t/T), \sin(\pi t/T), \cos(2\pi t/T), \ldots]^T$$
- LayerNorm:
$$\mu = \frac{1}{nd}\sum_{i=1}^{n}\sum_{k=1}^{d} o_{i,k}\,, \qquad \sigma = \sqrt{\frac{1}{nd}\sum_{i=1}^{n}\sum_{k=1}^{d}\left(o_{i,k} - \mu\right)^2}$$
- Residual Connections: $\boldsymbol{X}^{(l)} = \boldsymbol{X}^{(l-1)} + \boldsymbol{O}^{(l-1)}$