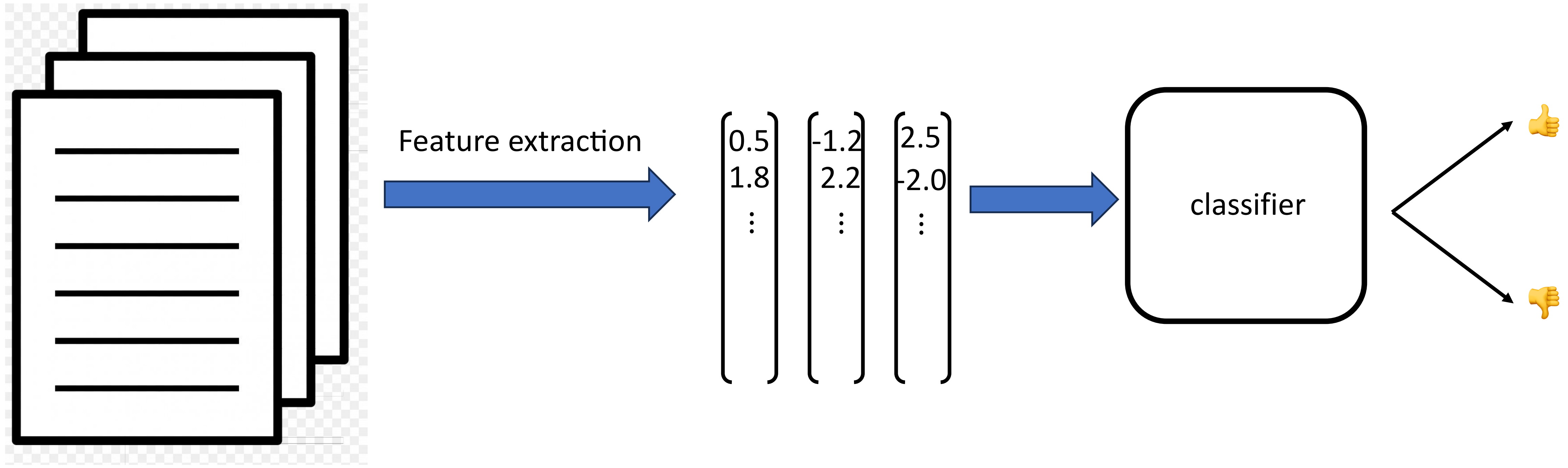
CS6120: Lecture 5

Jiaji Huang

https://jiaji-huang.github.io

Recap of Previous Lectures



documents

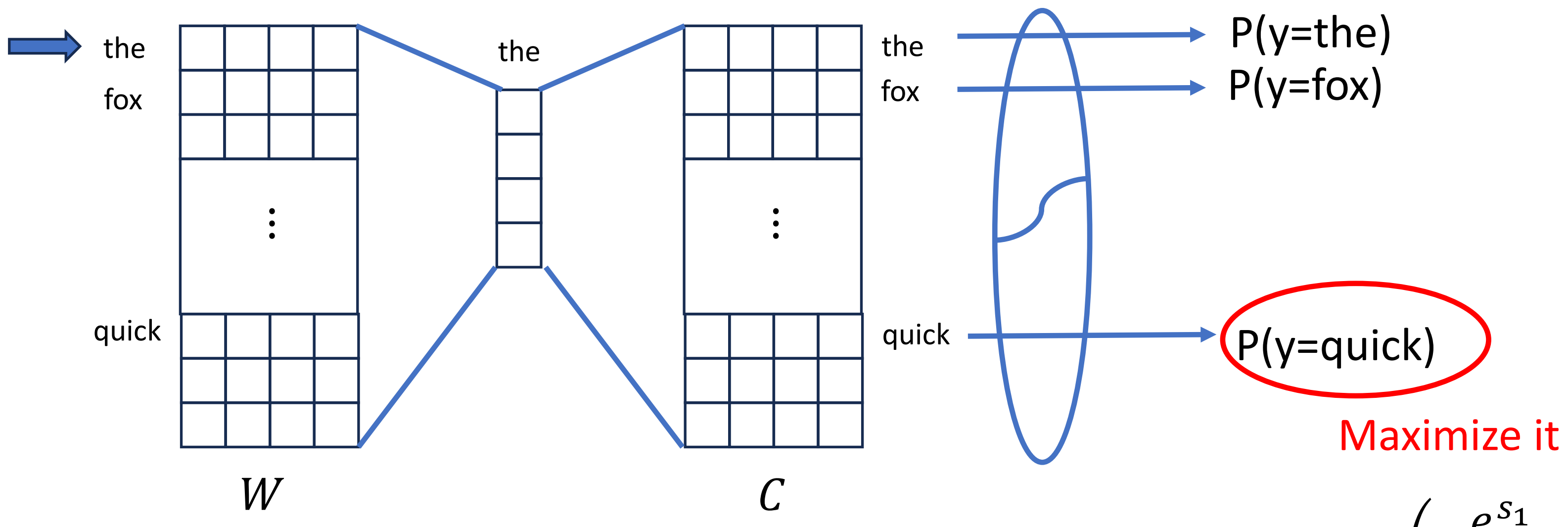
Representations

Recap of Previous Lectures

- Features
  - Bag of words
  - TF-IDF
  - Word2vec, PMI
- Classifiers
  - Naïve Bayesian
  - Logistic regression
  - Softmax classifier

Recap of Previous Lectures

- Review skip gram: a 2-layer network
- e.g., receive a training sample (x=the, y=quick)



No order information!

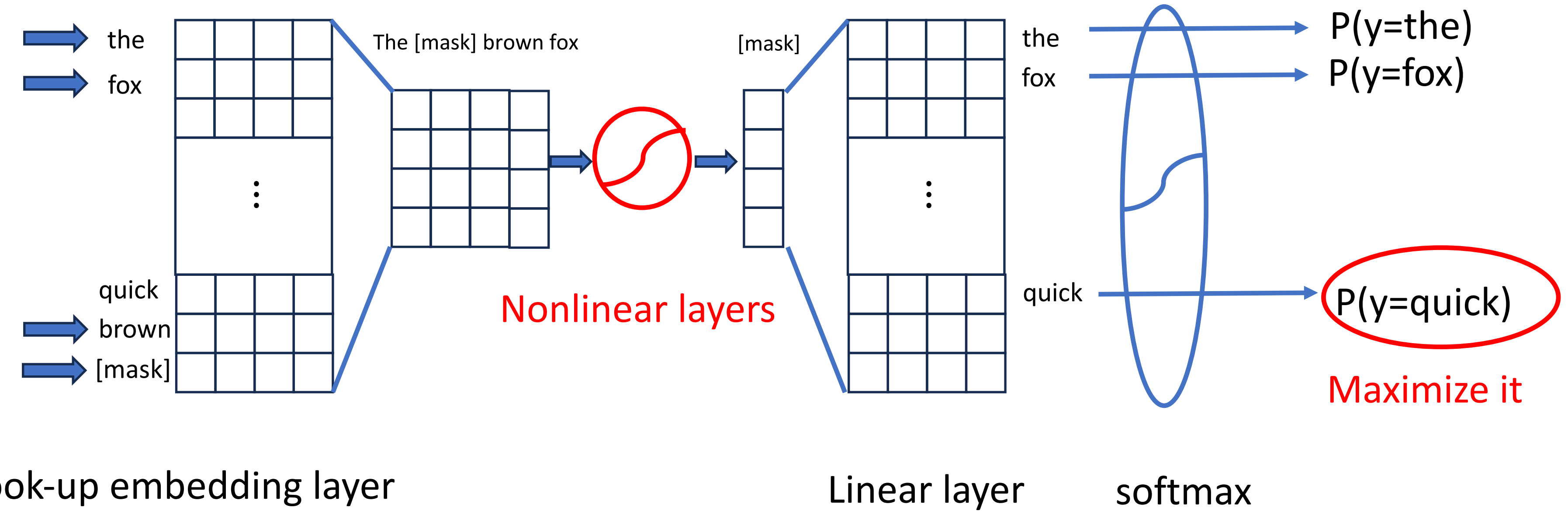
Look-up embedding layer

Linear layer

$$\text{softmax}(s_1, .., s_n) = \left(\frac{e^{s_1}}{\sum_{i=1}^{n} s_i}, \ldots, \frac{e^{s_n}}{\sum_{i=1}^{n} s_i}\right)$$

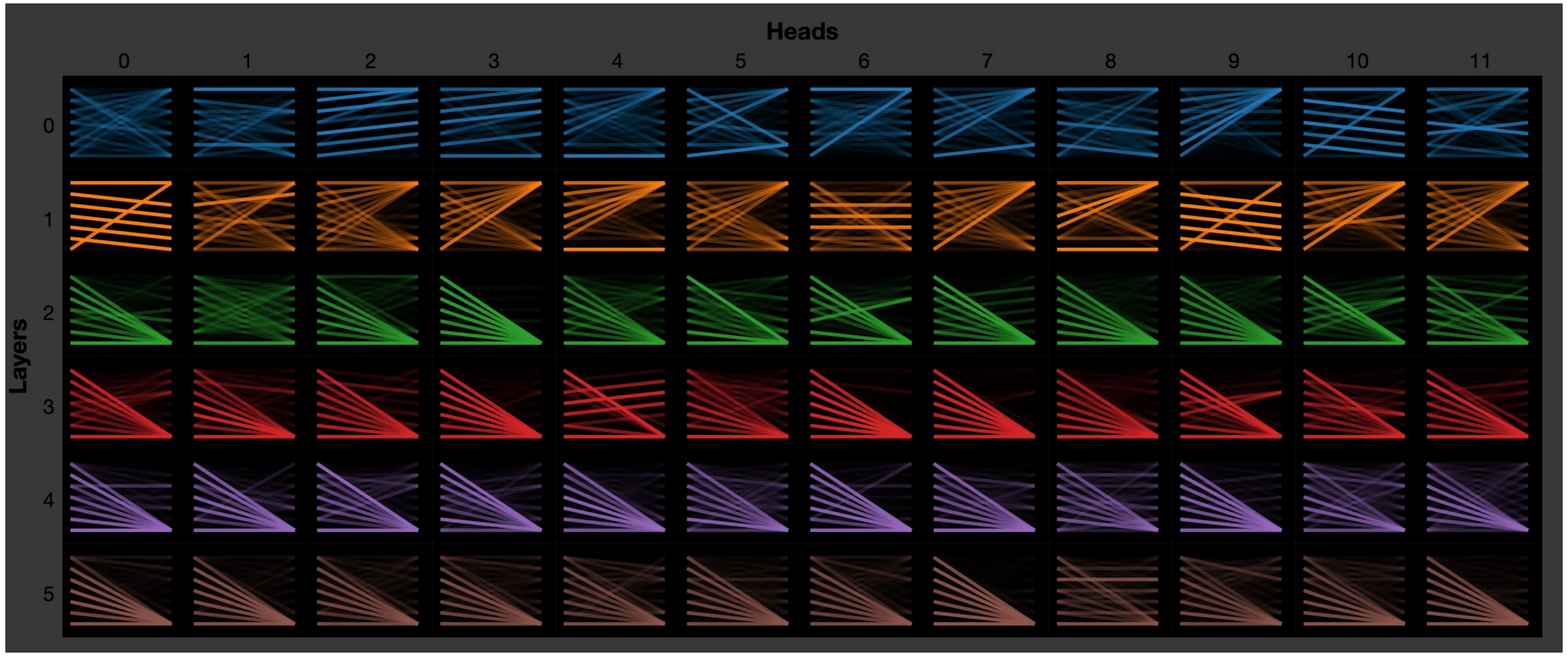
Modern architecture: Order Information

e.g., BERT (masked language modeling)



Recap of Previous Lectures

- Attention
- Visualization
- fine-tuning
- benchmarks



Agenda for Today

- Deep dive of “Nonlinear layers”
  - Recurrent Neural Network (RNN)
  - Architectures
  - Transformer
- Transformer Language Models
- Representations in Transformer

RNN

- Hidden state h_t
- Input x_t
- $h_t = g(Uh_{t-1} + Wx_t)$
- $y_t = f(Vh_t)$
- e.g., $g(\cdot)$ as tanh, $f(\cdot)$ as softmax

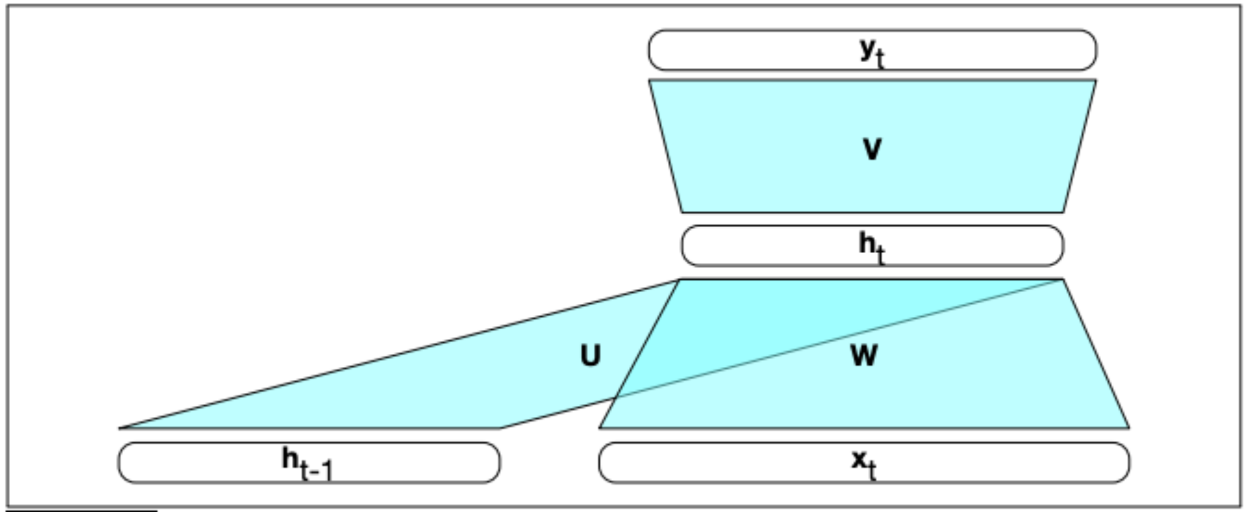


Figure 9.2 Simple recurrent neural network illustrated as a feedforward network.

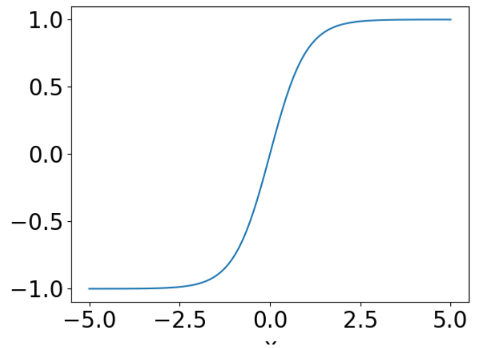


Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

RNN

- “Unroll” along time axis

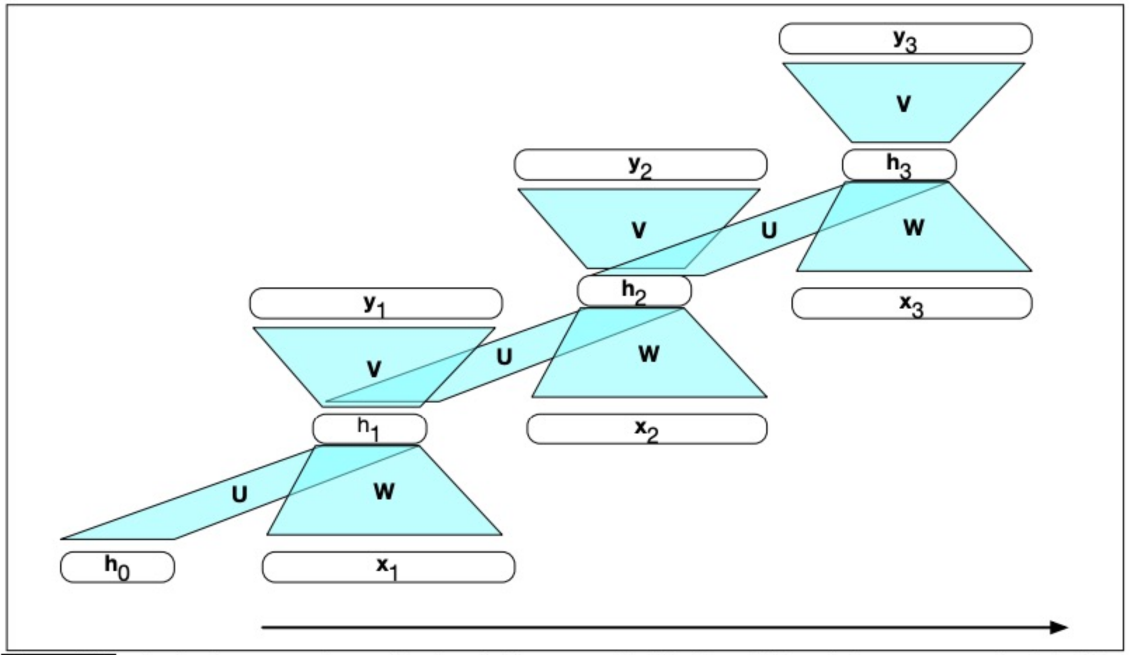


Figure 9.4 A simple recurrent neural network shown unrolled in time. Network layers are recalculated for each time step, while the weights **U**, **V** and **W** are shared across all time steps.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Stacked RNN layers

$$h_t^1 = g(U^1 h_{t-1}^1 + W^1 x_t)$$

$$h_t^2 = g(U^2 h_{t-1}^2 + W^2 h_t^1)$$

$$h_t^3 = g(U^3 h_{t-1}^3 + W^3 h_t^2)$$

$$\vdots$$

Stacked RNN layers

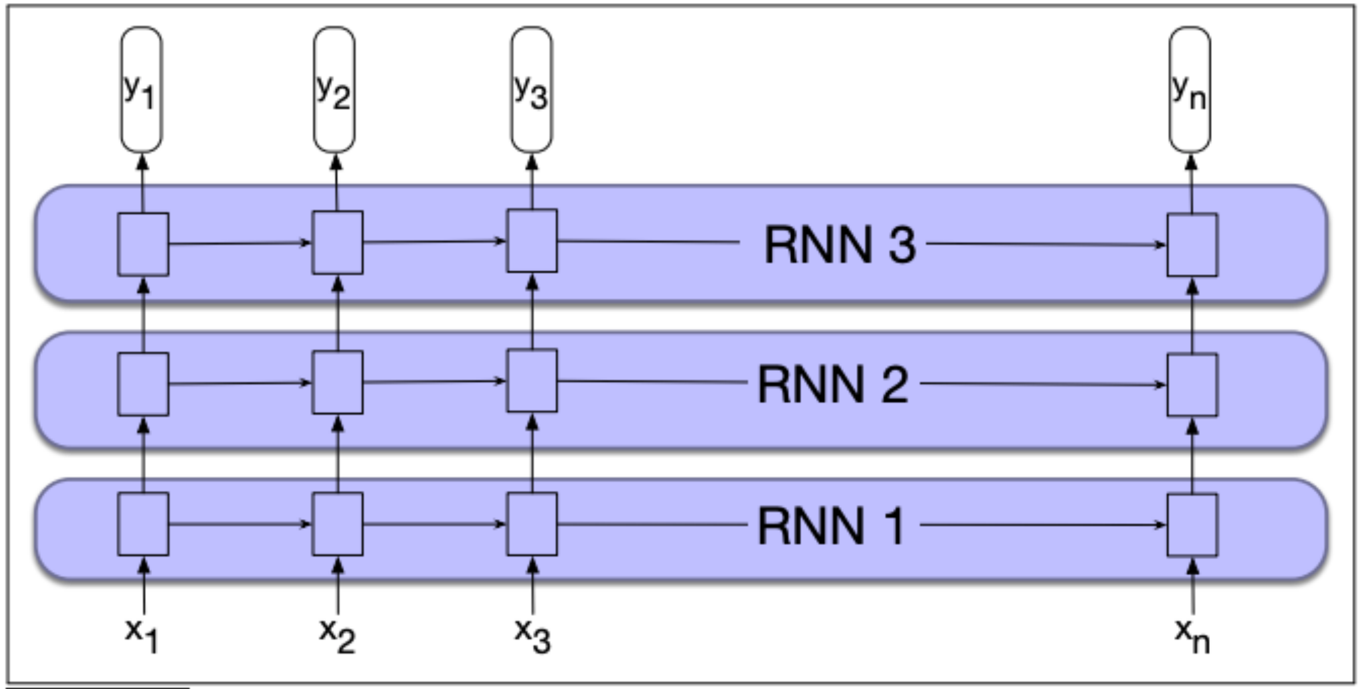


Figure 9.10 Stacked recurrent networks. The output of a lower level serves as the input to higher levels with the output of the last network serving as the final output.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

RNN for (Causal) language Modeling

- Predict next word given previous ones

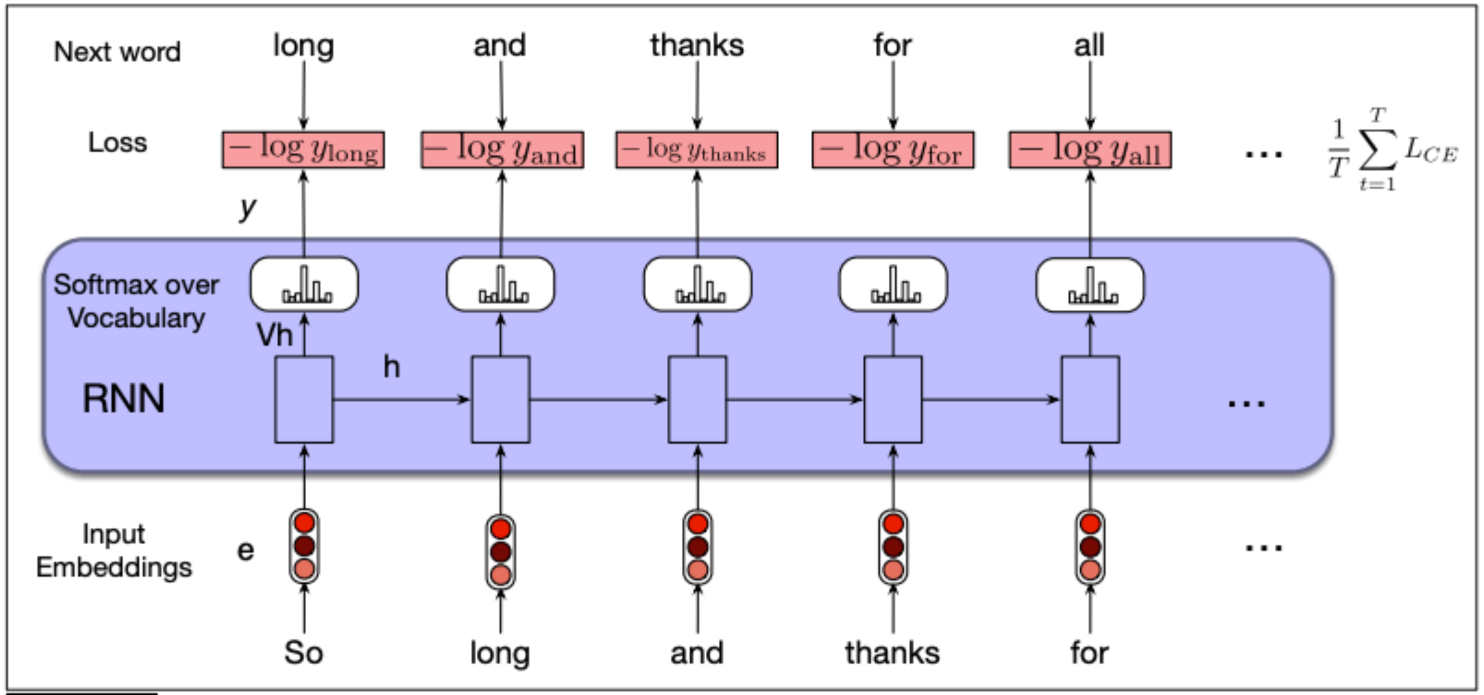


Figure 9.6 Training RNNs as language models.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

RNN for Text Generation

- Generate the next word given previous ones

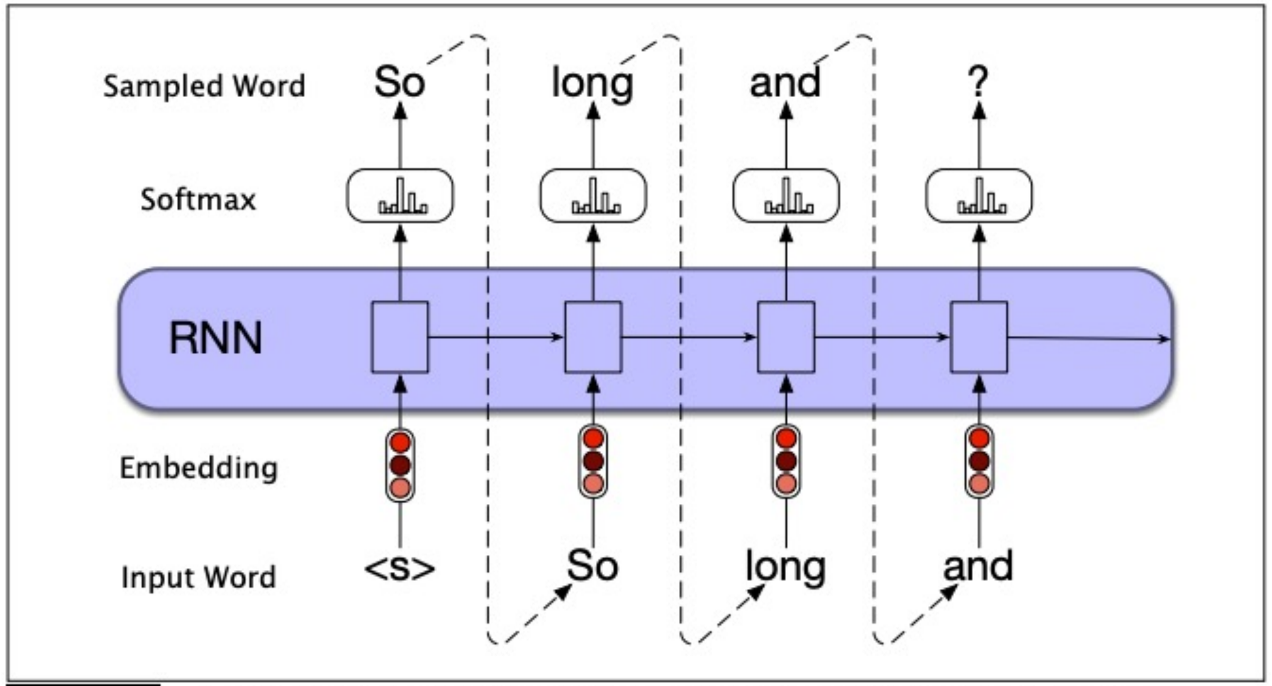


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Bi-directional RNN

- Input $x_1, x_2, \ldots, x_T$
- $h_t^f = RNN_f(x_1, x_2, \ldots, x_t)$
- $h_t^b = RNN_b(x_T, x_{T-1}, \ldots x_t)$
- $h_t = [h_t^f, h_t^b]$

Bi-directional RNN

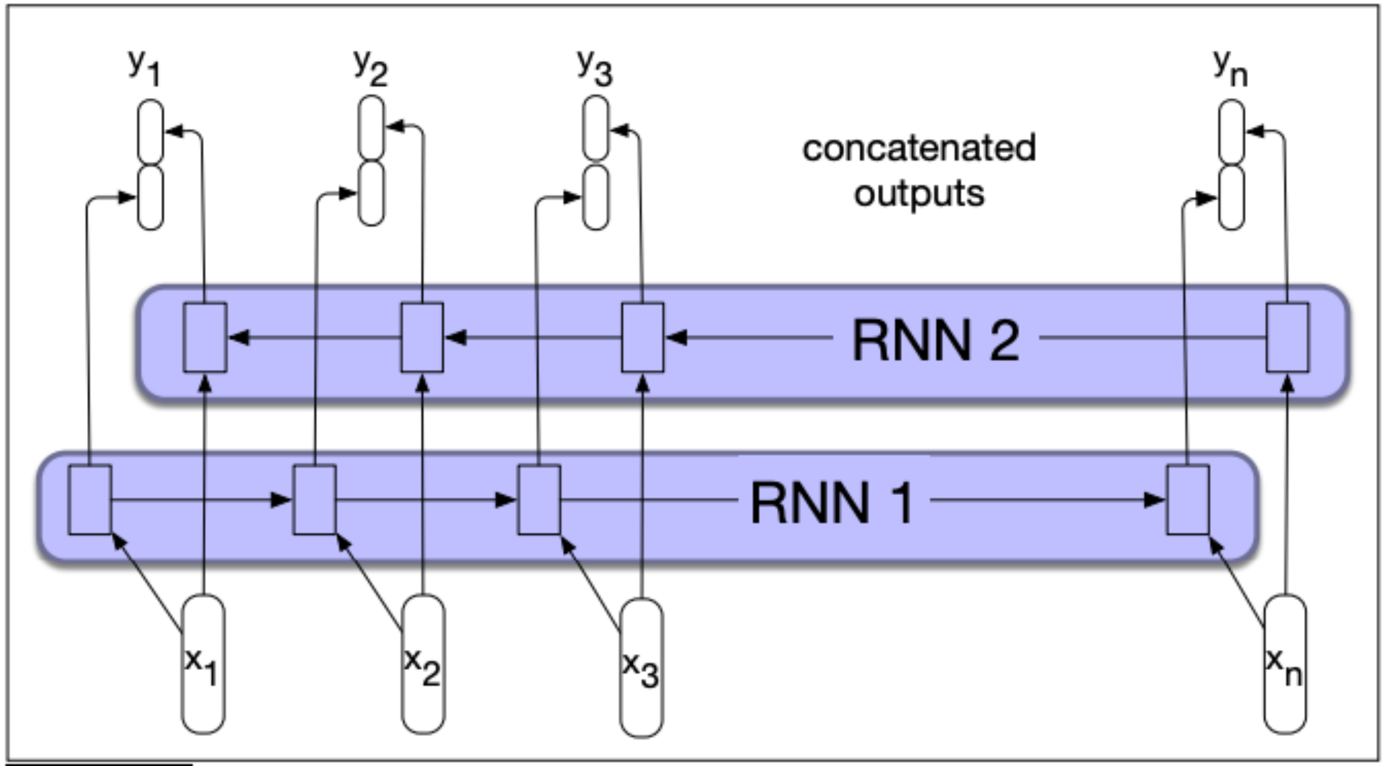
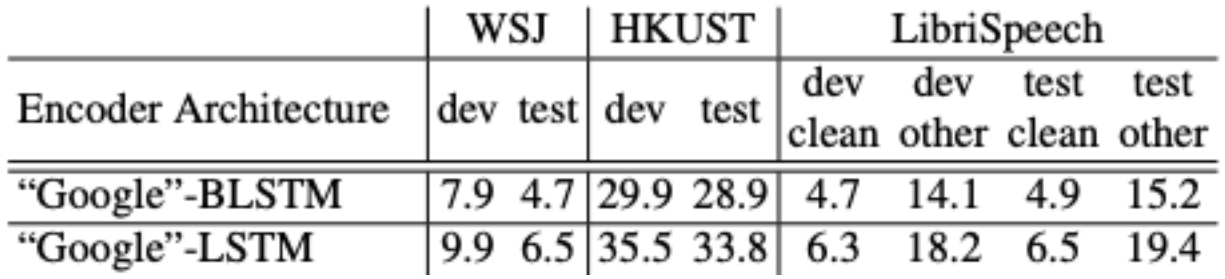


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Why bi-directional

- Some applications allow us to see the entire sequence, e.g.,
  - Text Classification
  - Speech Recognition
- Bi-directional often beats uni-directional:
  - e.g., a speech recognition example (lower WER is better)

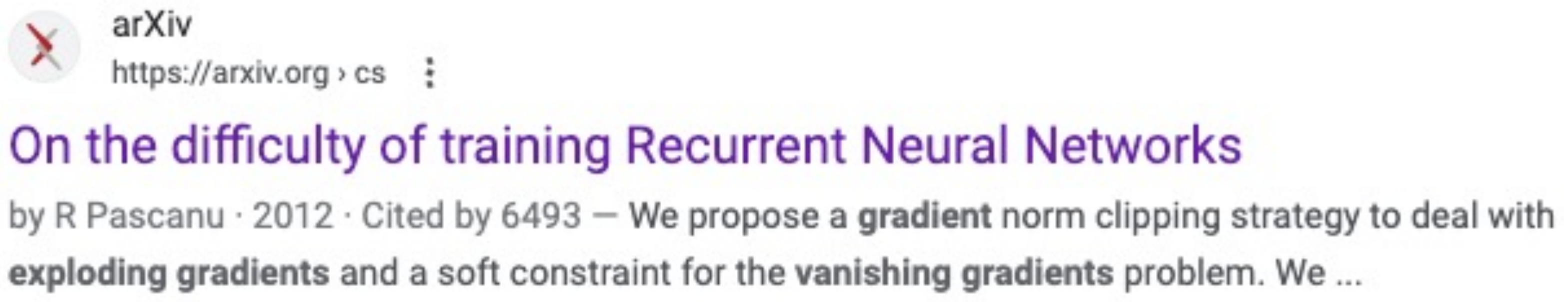


	WSJ		HKUST		LibriSpeech			
Encoder Architecture	dev	test	dev	test	dev clean	dev other	test clean	test other
"Google"-BLSTM	7.9	4.7	29.9	28.9	4.7	14.1	4.9	15.2
"Google"-LSTM	9.9	6.5	35.5	33.8	6.3	18.2	6.5	19.4

ASR results from https://www.jonathanleroux.org/pdf/Moritz2019Interspeech09.pdf

LSTM, but why?

- RNNs can be hard to train
- Gradient explosion and vanishing problem



arXiv
https://arxiv.org › cs

On the difficulty of training Recurrent Neural Networks

by R Pascanu · 2012 · Cited by 6493 — We propose a **gradient** norm clipping strategy to deal with **exploding gradients** and a soft constraint for the **vanishing gradients** problem. We ...

- In *modern words*, attention span is very short!

LSTM

- Forget gate

$$f_t = \sigma(U_f h_{t-1} + W_f x_t)$$
$$k_t = c_{t-1} \odot f_t$$

- Add gate

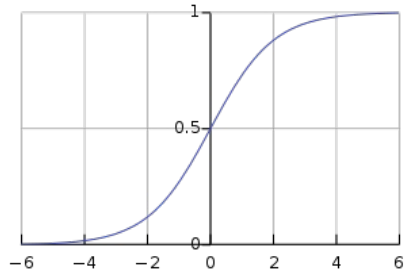
$$i_t = \sigma(U_i h_{t-1} + W_i x_t)$$
$$g_t = \tanh(U_g h_{t-1} + W_g x_t)$$
$$j_t = g_t \odot i_t$$

- Combined

$$c_t = j_t + k_t$$

- Output gate

$$o_t = \sigma(U_o h_{t-1} + W_o x_t)$$
$$h_t = o_t \odot \tanh(c_t)$$



sigmoid

LSTM

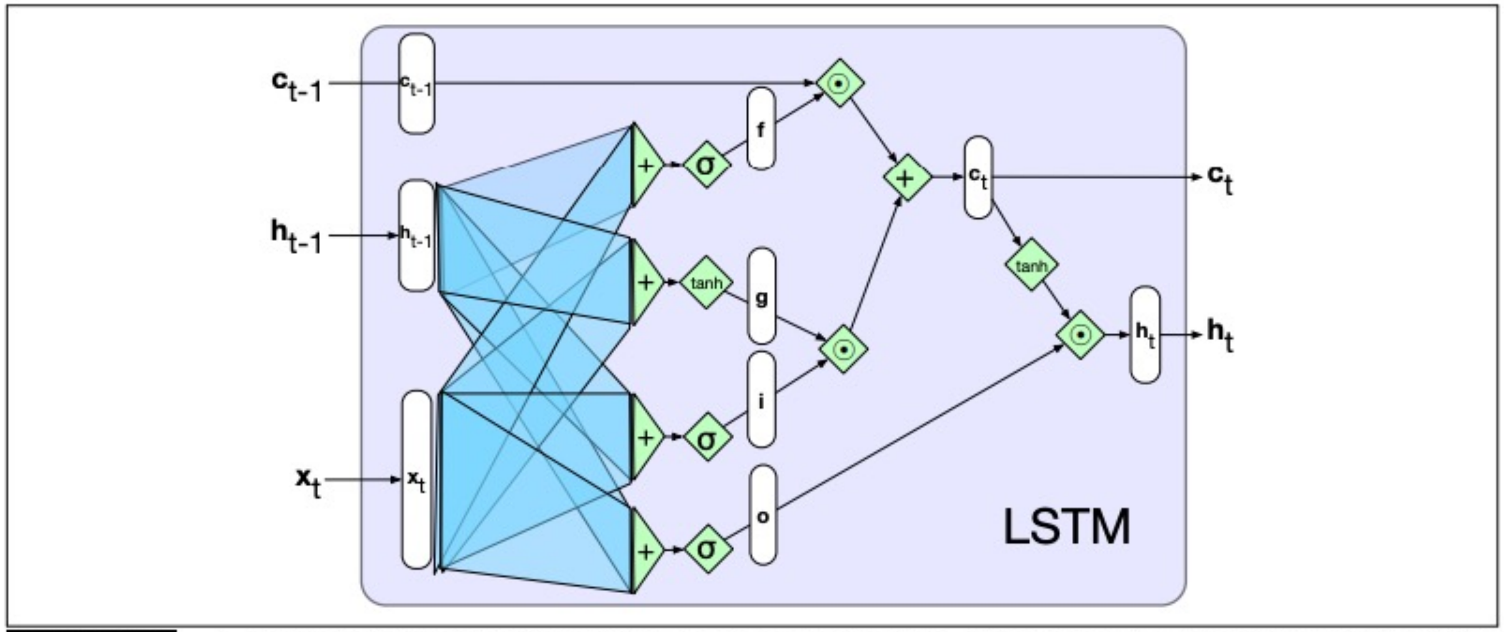
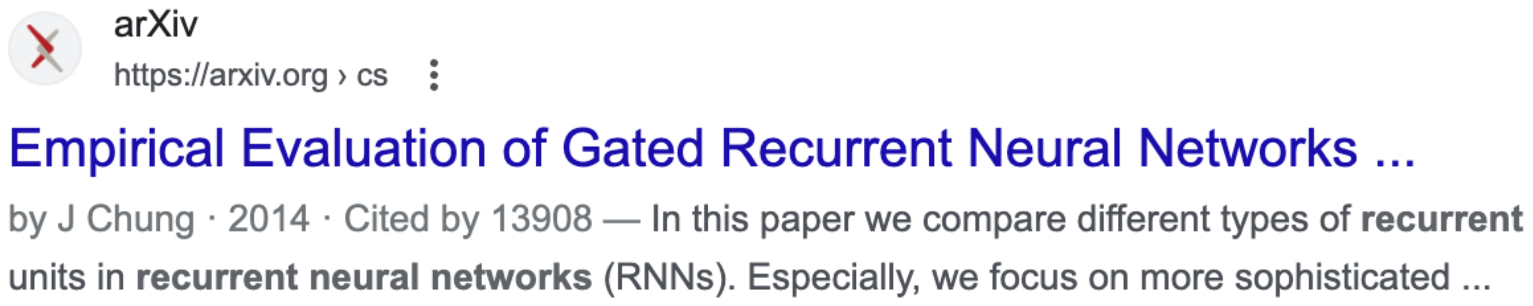


Figure 9.13 A single LSTM unit displayed as a computation graph. The inputs to each unit consists of the current input, x, the previous hidden state, h_{t-1}, and the previous context, c_{t-1}. The outputs are a new hidden state, h_t and an updated context, c_t.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

GRU



arXiv
https://arxiv.org › cs

Empirical Evaluation of Gated Recurrent Neural Networks ...

by J Chung · 2014 · Cited by 13908 — In this paper we compare different types of **recurrent** units in **recurrent neural networks** (RNNs). Especially, we focus on more sophisticated ...

- A Variant of LSTM
- Fewer gates, less parameters

Agenda for Today

- Deep dive of “Nonlinear layers”
  - Recurrent Neural Network (RNN)
  - Architectures
  - Transformer
- Transformer Language Models
- Representations in Transformer

Typical Architectures

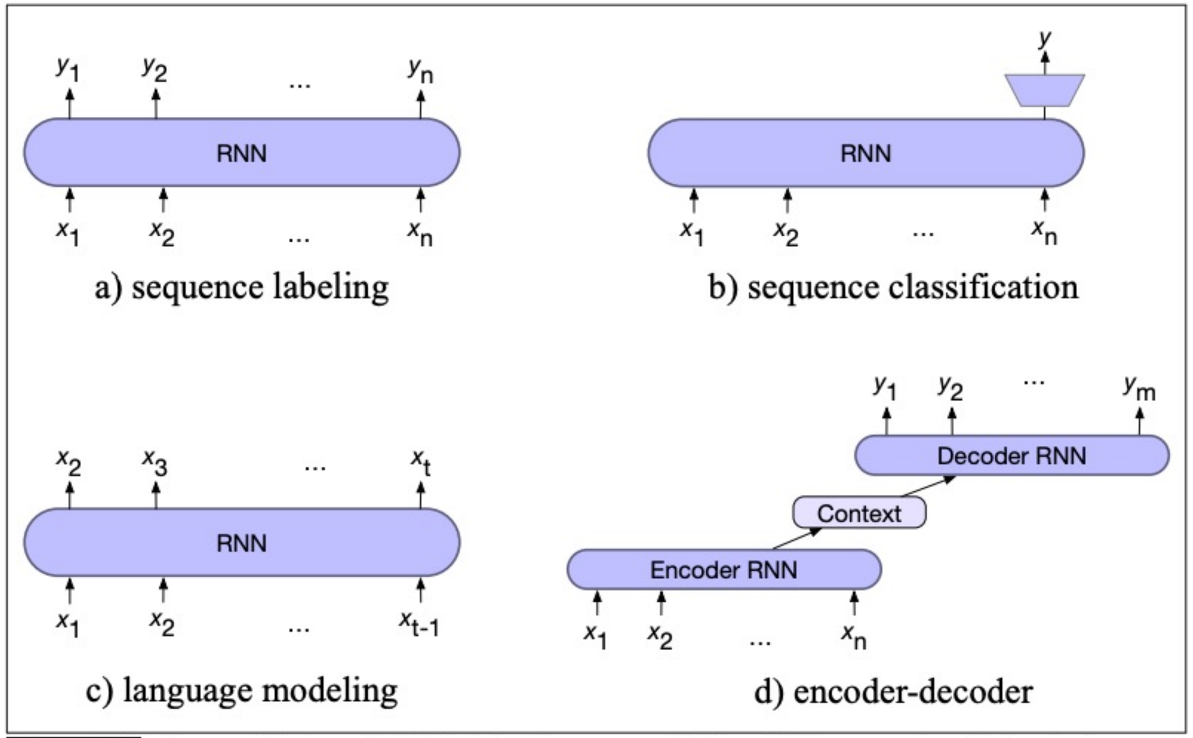


Figure 9.15 Four architectures for NLP tasks. In sequence labeling (POS or named entity tagging) we map each input token x_i to an output token y_i. In sequence classification we map the entire input sequence to a single class. In language modeling we output the next token conditioned on previous tokens. In the encoder model we have two separate RNN models, one of which maps from an input sequence **x** to an intermediate representation we call the **context**, and a second of which maps from the context to an output sequence **y**.

Note: all RNNs can be replaced by LSTMs!

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Enc-Dec with RNNs

- Applications: translation, summarization, dialog, …

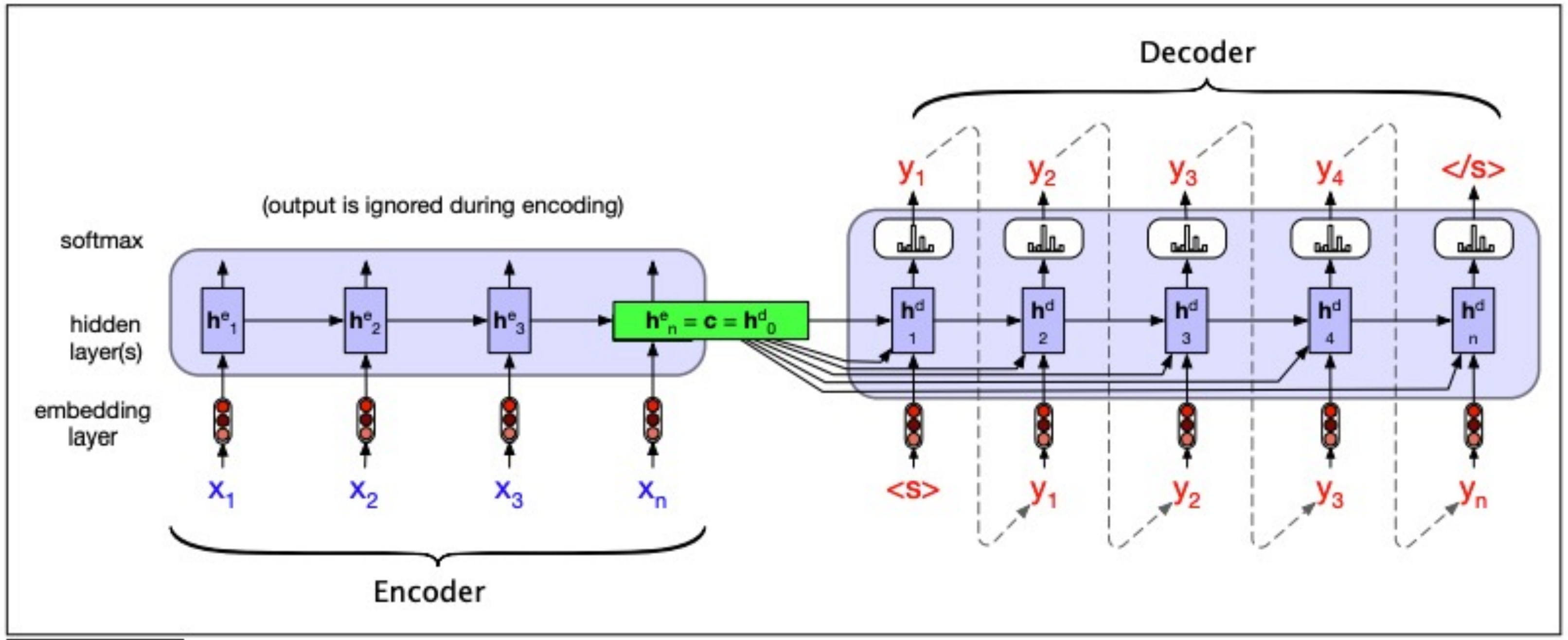


Figure 9.18 A more formal version of translating a sentence at inference time in the basic RNN-based encoder-decoder architecture. The final hidden state of the encoder RNN, h^e_n, serves as the context for the decoder in its role as h^d_0 in the decoder RNN.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Attention in Enc-Dec Architecture

- Drawback of using a single context vector

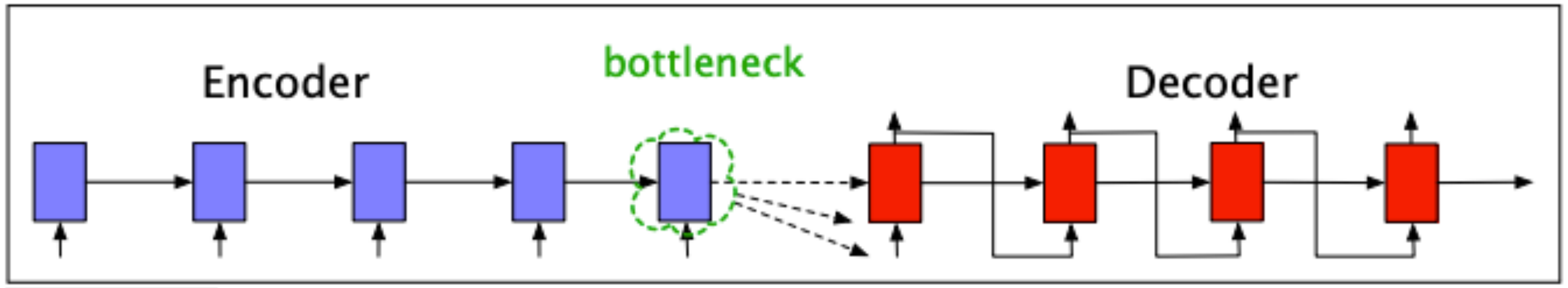


Figure 9.21 Requiring the context *c* to be only the encoder's final hidden state forces all the information from the entire source sentence to pass through this representational bottleneck.

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Attention in Enc-Dec Architecture

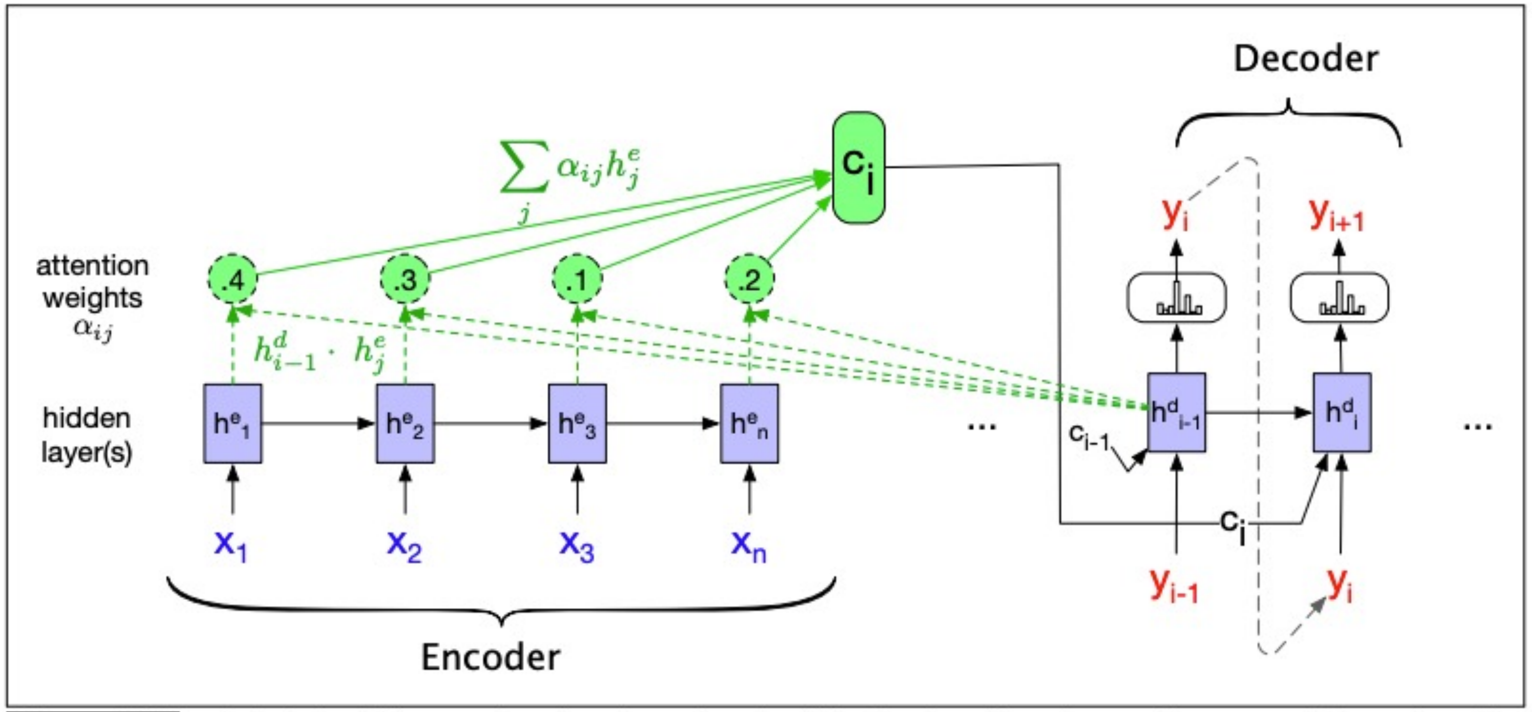


Figure 9.23 A sketch of the encoder-decoder network with attention, focusing on the computation of $\mathbf{c}_i$. The context value $\mathbf{c}_i$ is one of the inputs to the computation of $\mathbf{h}^d_i$. It is computed by taking the weighted sum of all the encoder hidden states, each weighted by their dot product with the prior decoder hidden state $\mathbf{h}^d_{i-1}$.

$$\alpha_{ij} = h^d_i \cdot h^e_j$$

Illustration from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Agenda for Today

- Deep dive of “Nonlinear layers”
  - Recurrent Neural Network (RNN)
  - Architectures
  - Transformer
- Transformer Language Models
- Representations in Transformer

More Parallelizable than RNN

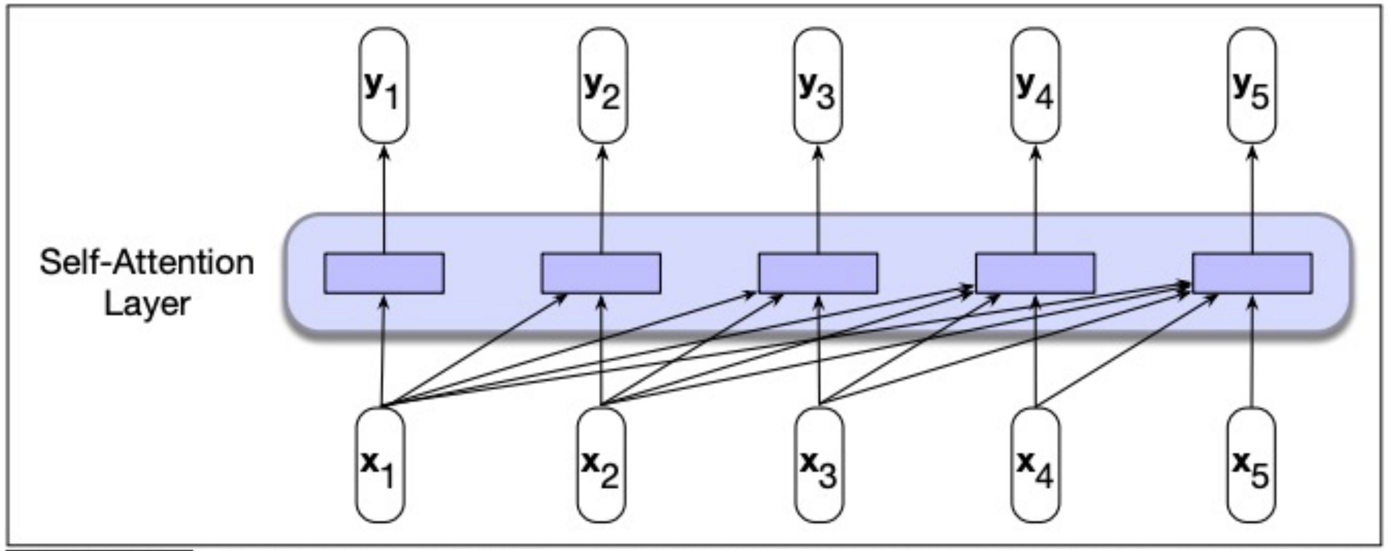


Figure 10.1 Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be performed in parallel.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

A Transformer Block

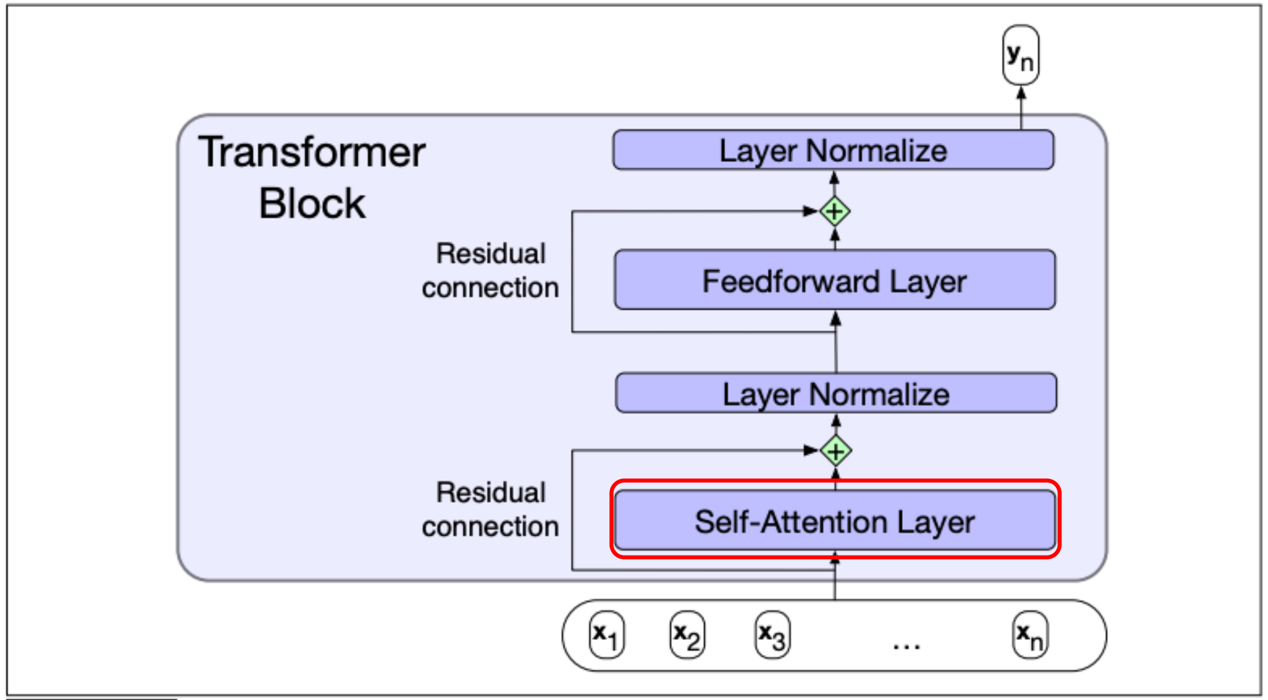


Figure 10.4 A transformer block showing all the layers.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Math of Self-attention

- Map hidden states to keys, queries and values

$$q_i = W^q x_i, k_i = W^k x_i, v_i = W^v x_i$$

- Raw score

$$s_{i,j} = q_i \cdot k_j / \sqrt{d}$$

- Attention weights

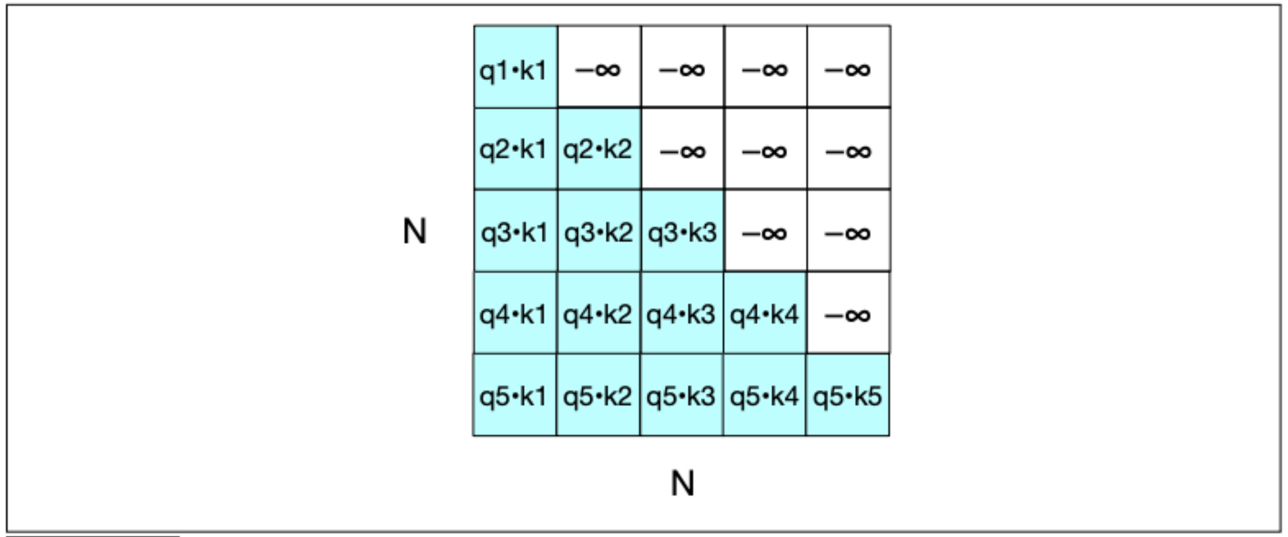
$$\alpha_{i,j} = \frac{e^{s_{i,j}}}{\sum_l e^{s_{i,l}}}$$

- Output Hidden states

$$y_i = \sum_j \alpha_{i,j} v_j$$

Attention Mask

- $\alpha_{i,j} = \frac{e^{s_{i,j}}}{\sum_l e^{s_{i,l}}}, y_i = \sum_j \alpha_{i,j} v_j$
- The range of j (or l) for causal language model: $j \leq i$



q1•k1	−∞	−∞	−∞	−∞
q2•k1	q2•k2	−∞	−∞	−∞
q3•k1	q3•k2	q3•k3	−∞	−∞
q4•k1	q4•k2	q4•k3	q4•k4	−∞
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

Figure 10.3 The $N \times N$ $\mathbf{QK}^\intercal$ matrix showing the $q_i \cdot k_j$ values, with the upper-triangle portion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax will turn to zero).

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Multi-head Attention

- Multiple sets of W^q, W^k and W^v
- concatenate output from each head
- Project with W^o

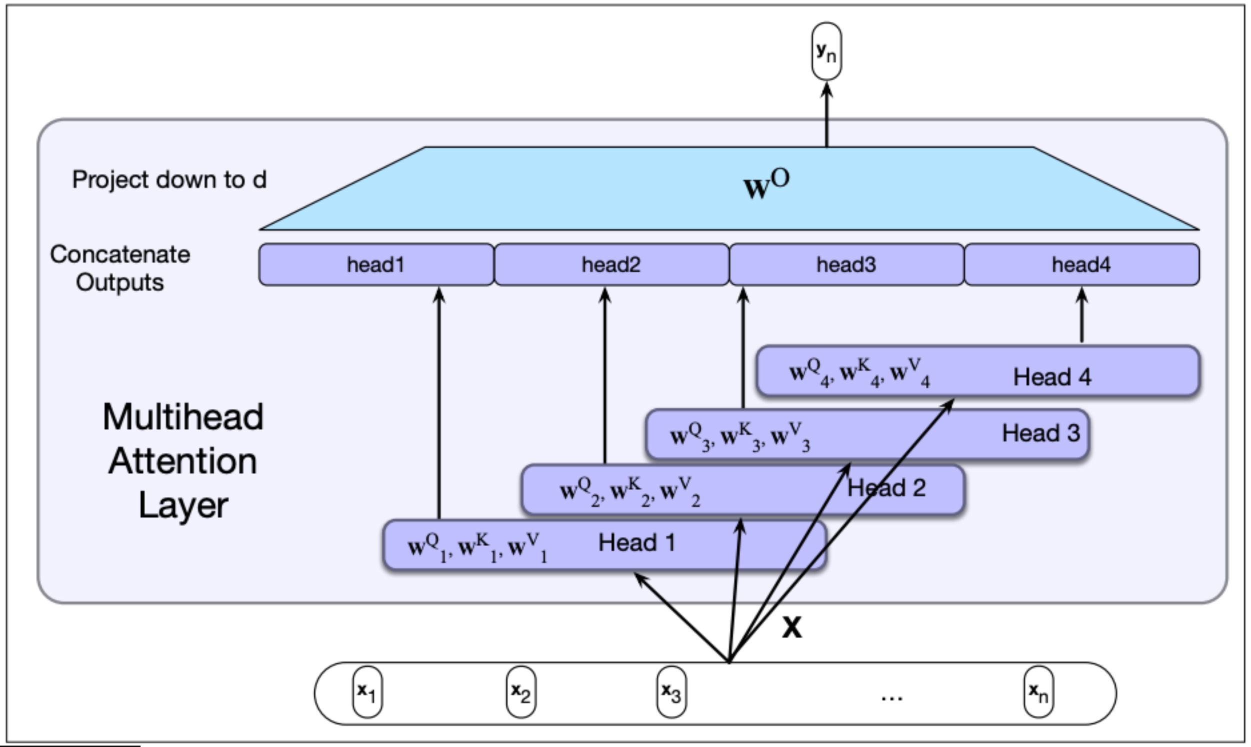
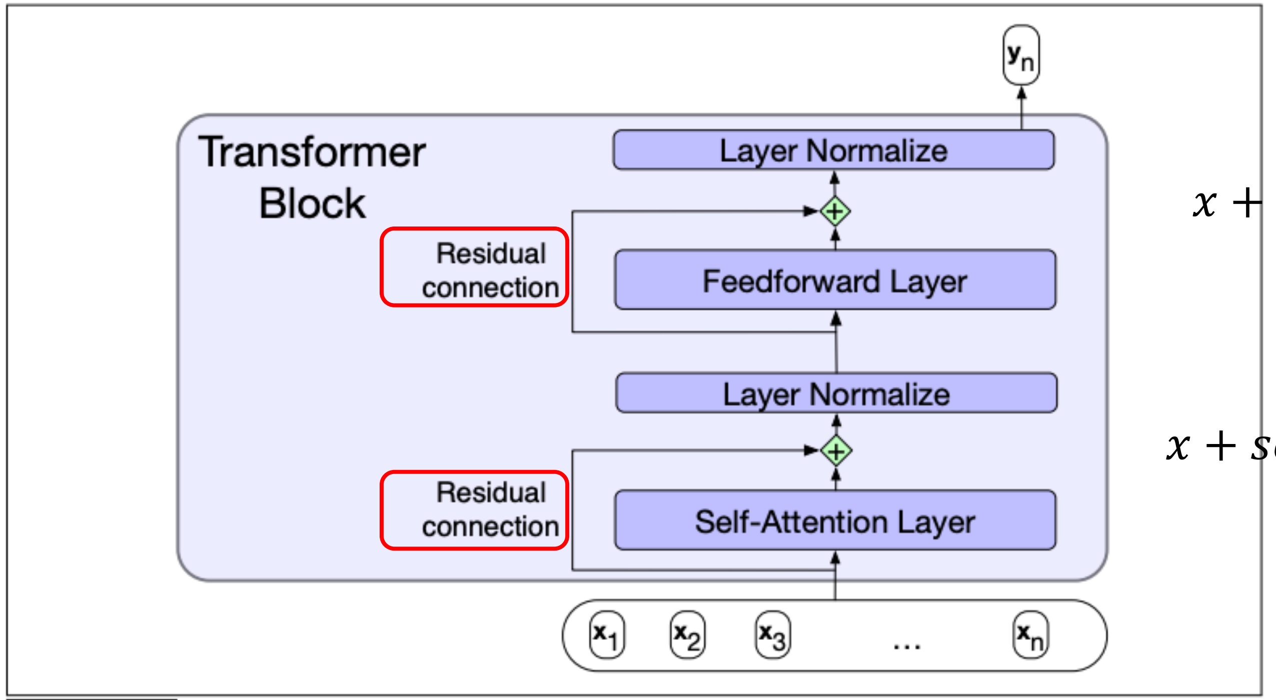


Figure 10.5 Multihead self-attention: Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices. The outputs from each of the layers are concatenated and then projected down to *d*, thus producing an output of the same size as the input so layers can be stacked.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Residual Connection



$x + FF(x)$

$x + selfAttn(x)$

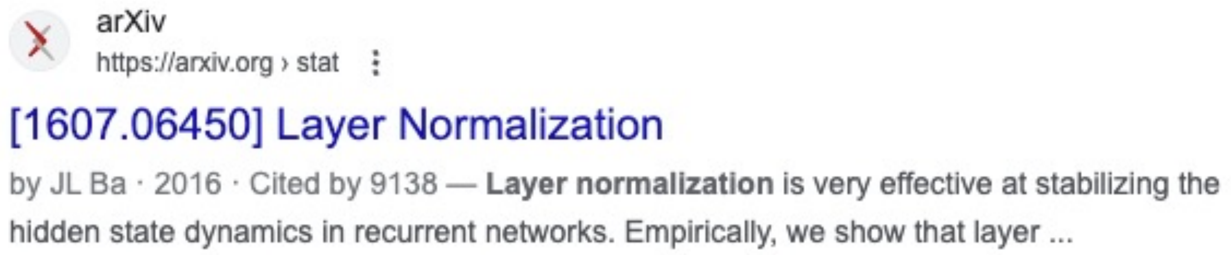
Figure 10.4 A transformer block showing all the layers.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Why Residual Connection

- Consider a system with residual connection
$$y = Ax(\theta) + x(\theta)$$
- We use gradient descent to train θ
$$\frac{\partial y}{\partial \theta} = (A + I)\frac{\partial x}{\partial \theta}$$
- $A + I$ is better “conditioned” than A
- Called diagonal loading sometimes

Layer Normalization


arXiv
https://arxiv.org › stat

[1607.06450] Layer Normalization

by JL Ba · 2016 · Cited by 9138 — **Layer normalization** is very effective at stabilizing the hidden state dynamics in recurrent networks. Empirically, we show that layer ...

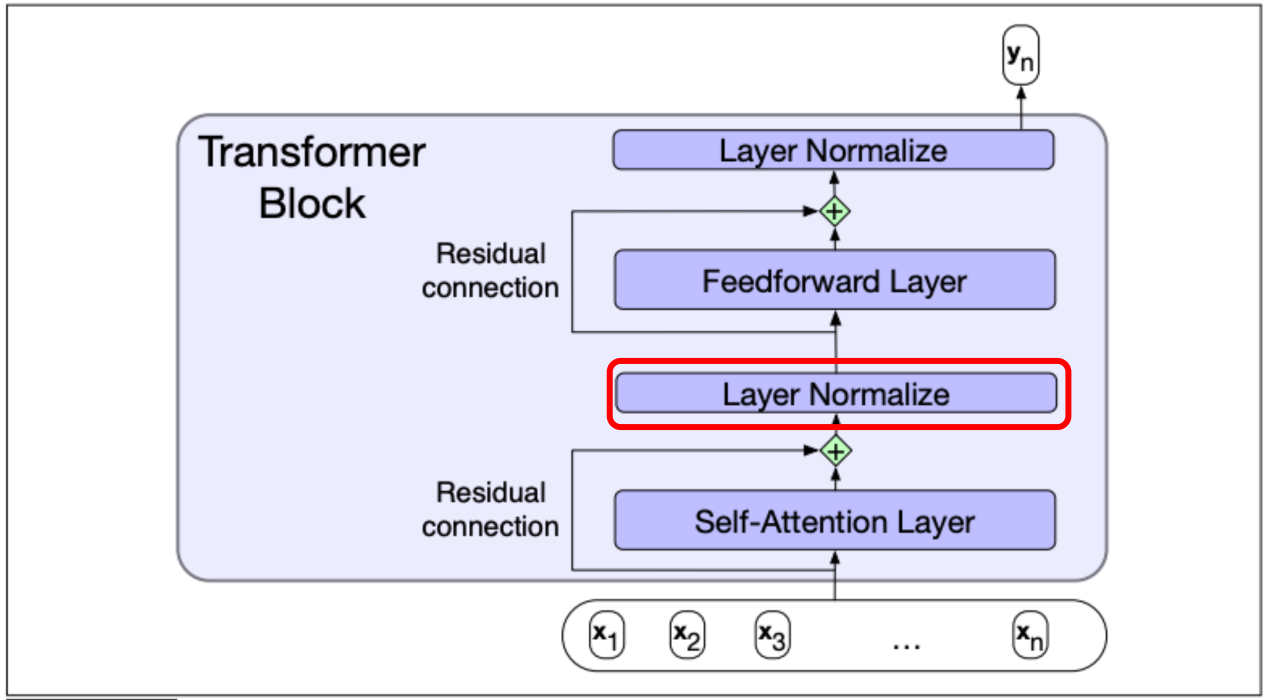


Figure 10.4 A transformer block showing all the layers.

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i-\mu)^2}$$

$$\hat{\mathbf{x}} = \frac{(\mathbf{x}-\mu)}{\sigma}$$

$$\text{LayerNorm} = \gamma\hat{\mathbf{x}}+\beta$$

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Feedforward Layer

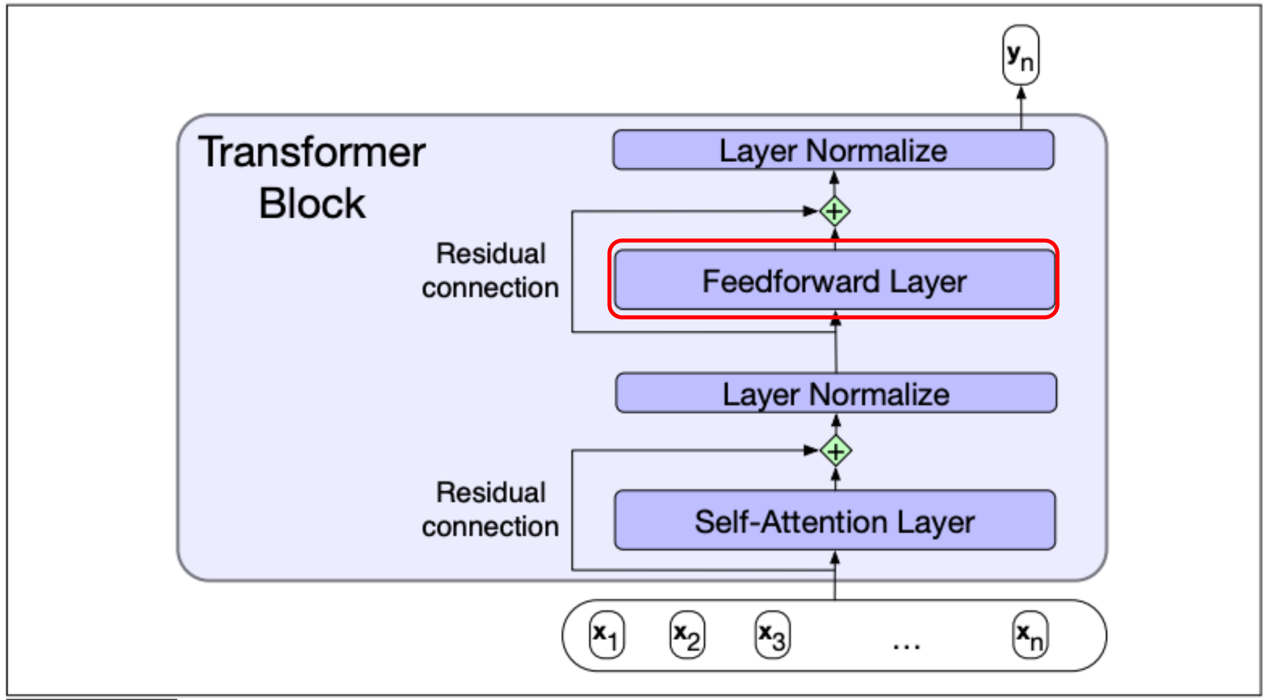


Figure 10.4 A transformer block showing all the layers.

$$y = W_2 \sigma(W_1 x)$$

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Brief Summary

- What are the parameters to be trained?
- Which part has most parameters?

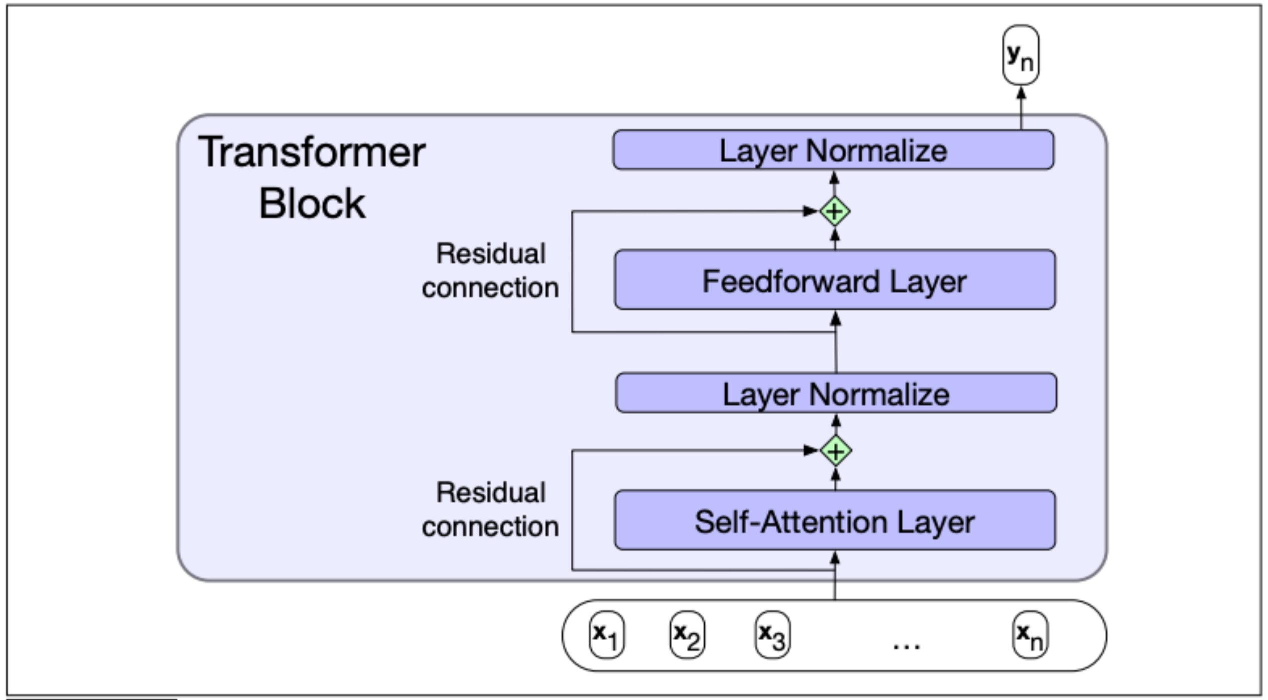


Figure 10.4 A transformer block showing all the layers.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Word Order Information

- Encode following two sentences

  “It is good, isn’t it ?” v.s. “it isn’t good, is it ?”
- Take the hidden state at last time step as sentence embedding
- The embeddings are the same!
- How to break the symmetry?

Positional Encoding

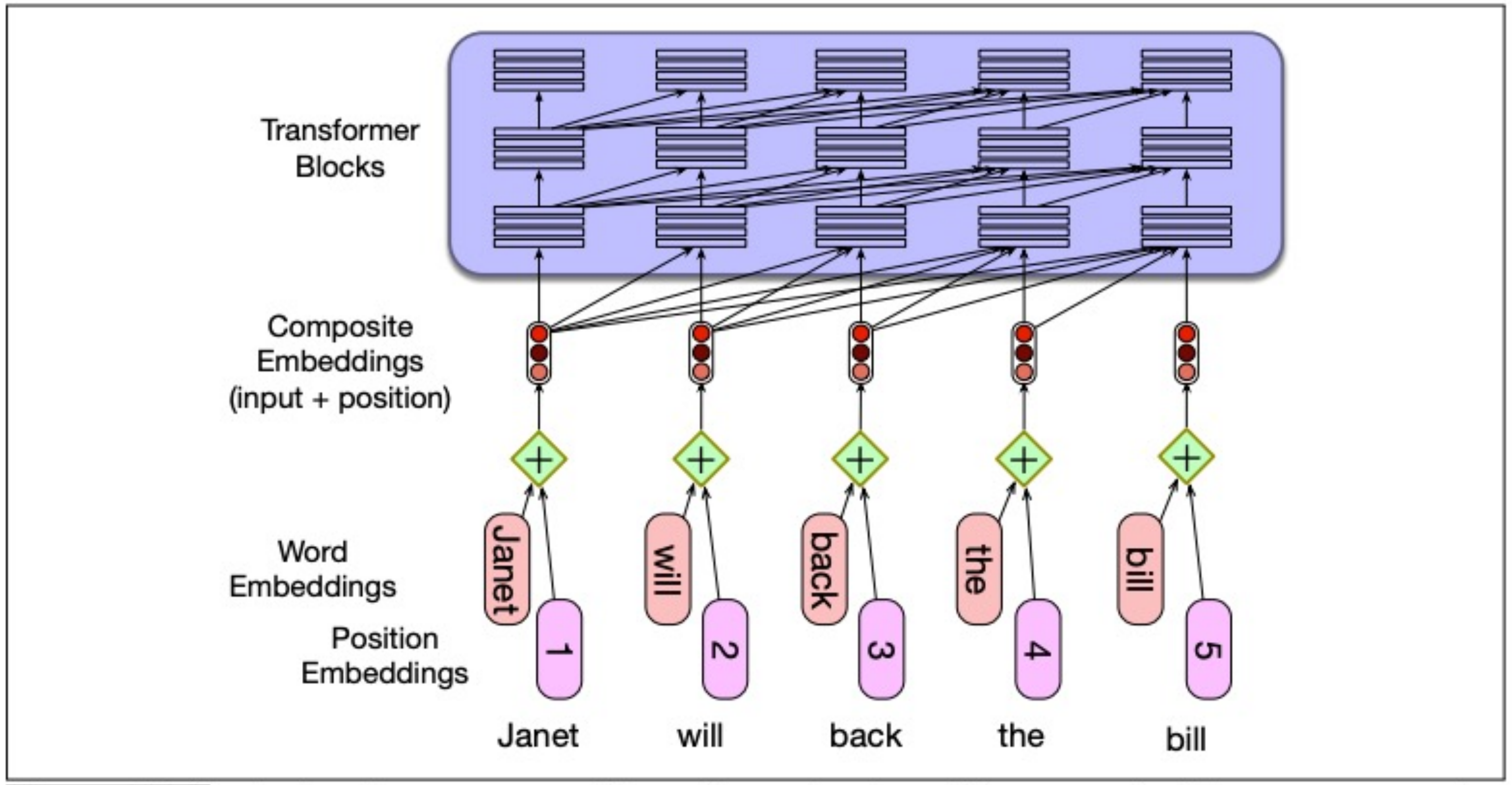


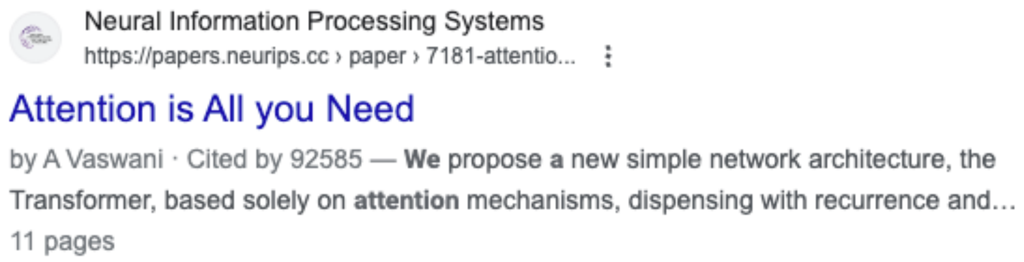
Figure 10.6 A simple way to model position: simply adding an embedding representation of the absolute position to the input word embedding to produce a new embedding of the same dimenionality.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Absolute Position Encoding

- Each position with a encoding vector
- E.g, sinusoidal

$$\vec{p}_i = \left[\sin \omega_1 i\,, \cos \omega_1\, i, \dots, \sin \omega_{d/2} i, \cos \omega_{d/2}\, i\right]$$



Neural Information Processing Systems
https://papers.neurips.cc › paper › 7181-attentio...

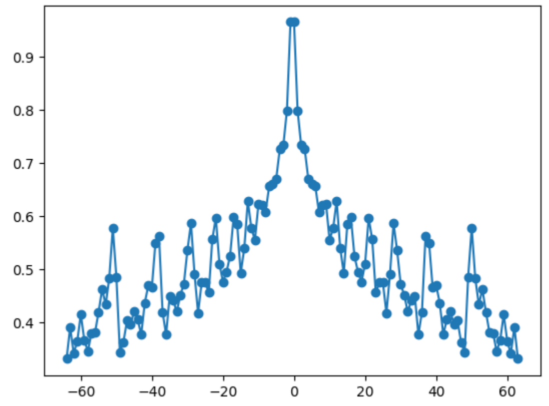
Attention is All you Need

by A Vaswani · Cited by 92585 — **We propose a new simple network architecture, the** Transformer, based solely on **attention** mechanisms, dispensing with recurrence and…

11 pages

- Why it helps

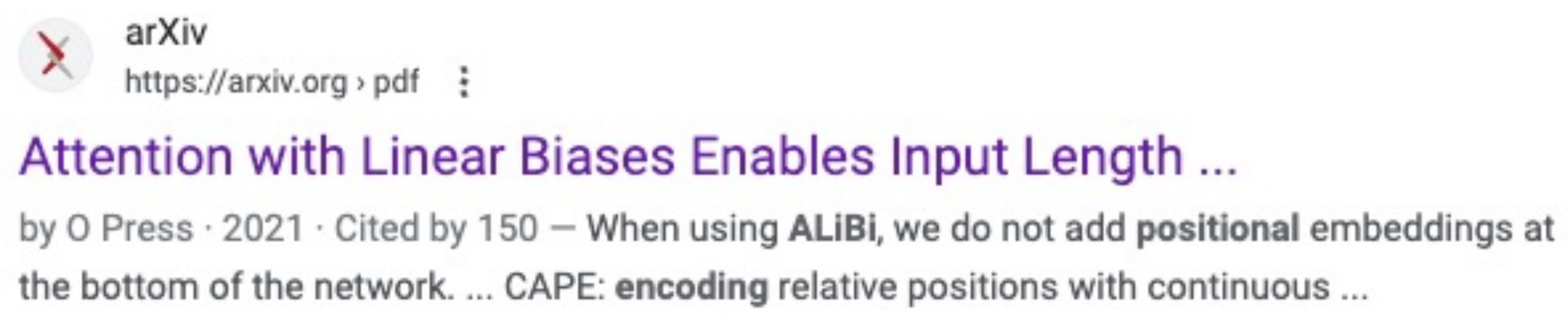
$$\vec{p}_i \cdot \vec{p}_j = \sum_{s=1}^{d/2} \cos \omega_s (i - j)$$



Relative Position Encoding

- Some function of $i - j$
- E.g., ALiBi's raw score

$$s_{i,j} = \frac{q_i \cdot k_j}{\sqrt{d}} - c|i - j|, c > 0$$



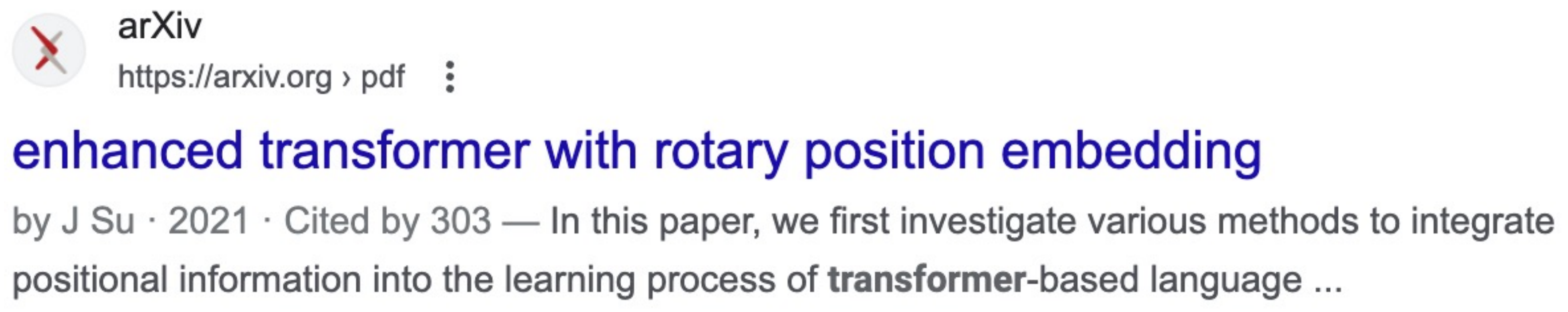
arXiv
https://arxiv.org › pdf

Attention with Linear Biases Enables Input Length ...

by O Press · 2021 · Cited by 150 — When using **ALiBi**, we do not add **positional** embeddings at the bottom of the network. ... CAPE: **encoding** relative positions with continuous ...

- Encourage to pay more attention to nearby tokens

RoPE: Another Relative Position Encoding


arXiv
https://arxiv.org › pdf

enhanced transformer with rotary position embedding

by J Su · 2021 · Cited by 303 — In this paper, we first investigate various methods to integrate positional information into the learning process of **transformer**-based language ...

- Rotate q_i by $i\times\theta$, k_j by $j\times\theta$
- So if $|i - j|$ big, their inner product is small

Agenda for Today

- Deep dive of “Nonlinear layers”
  - Recurrent Neural Network (RNN)
  - Architectures
  - Transformer
- Transformer Language Models
- Representations in Transformer

Masked Language Model

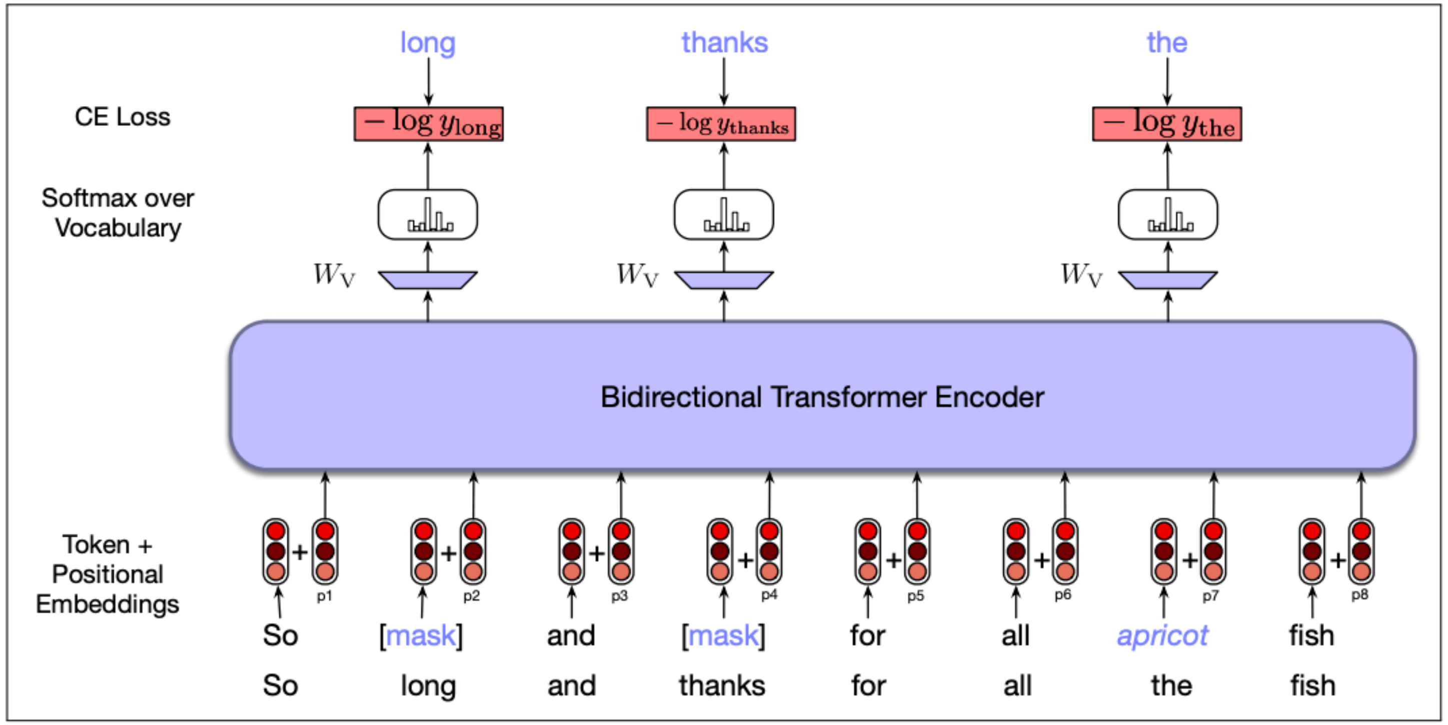


Figure 11.5 Masked language model training. In this example, three of the input tokens are selected, two of which are masked and the third is replaced with an unrelated word. The probabilities assigned by the model to these three items are used as the training loss. (In this and subsequent figures we display the input as words rather than subword tokens; the reader should keep in mind that BERT and similar models actually use subword tokens instead.)

- Sees left and right contexts
- no attention mask

Illustration from https://web.stanford.edu/~jurafsky/slp3/11.pdf

Masked Language Model

- Aka encoder model
- E.g. BERT
- Application: language understanding tasks

Causal Language Model

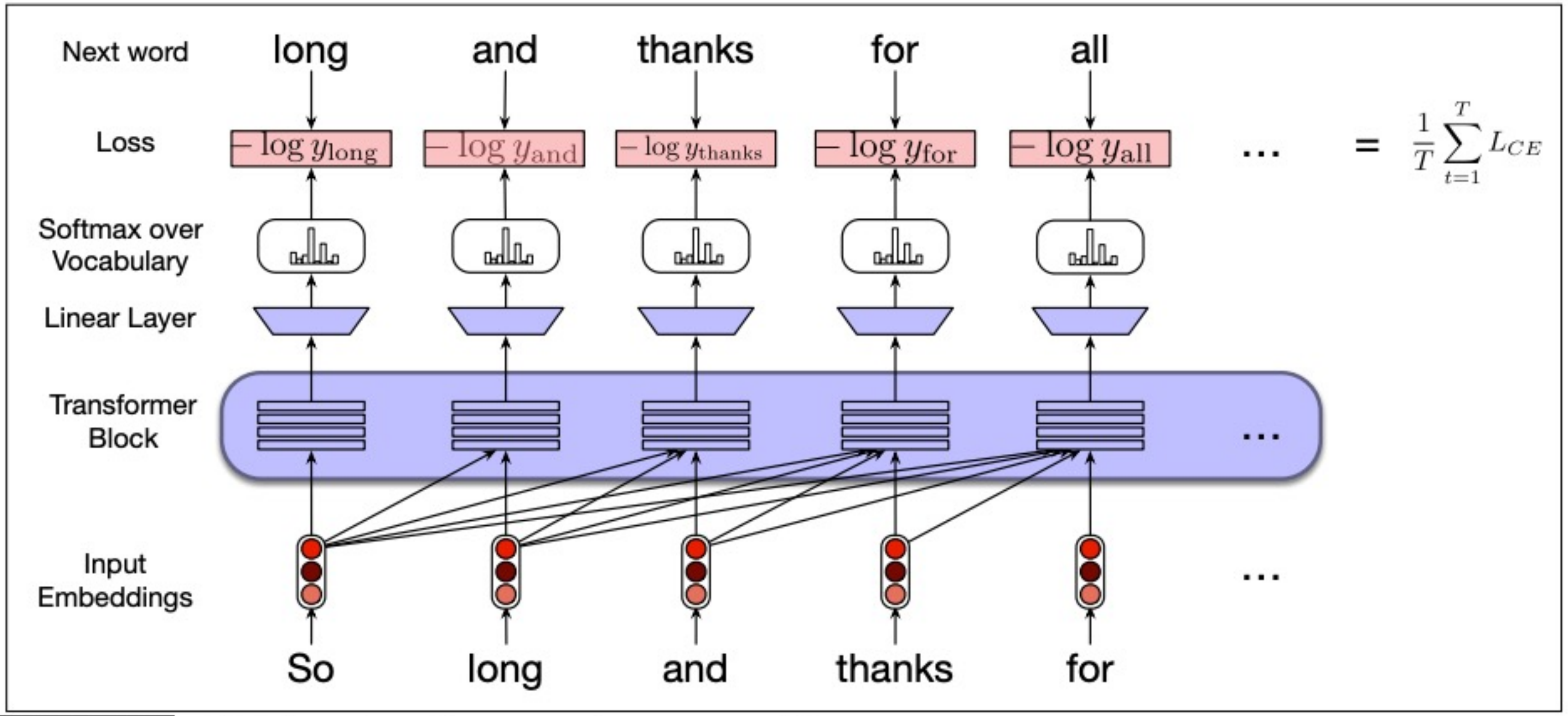


Figure 10.7 Training a transformer as a language model.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Causal Language Model

- Aka decoder model
- E.g., GPT-x, Llama
- Applications: language generation tasks

Language Generation

- How to sample the most probable path from softmax probabilities?
- Greedy
- Beam search

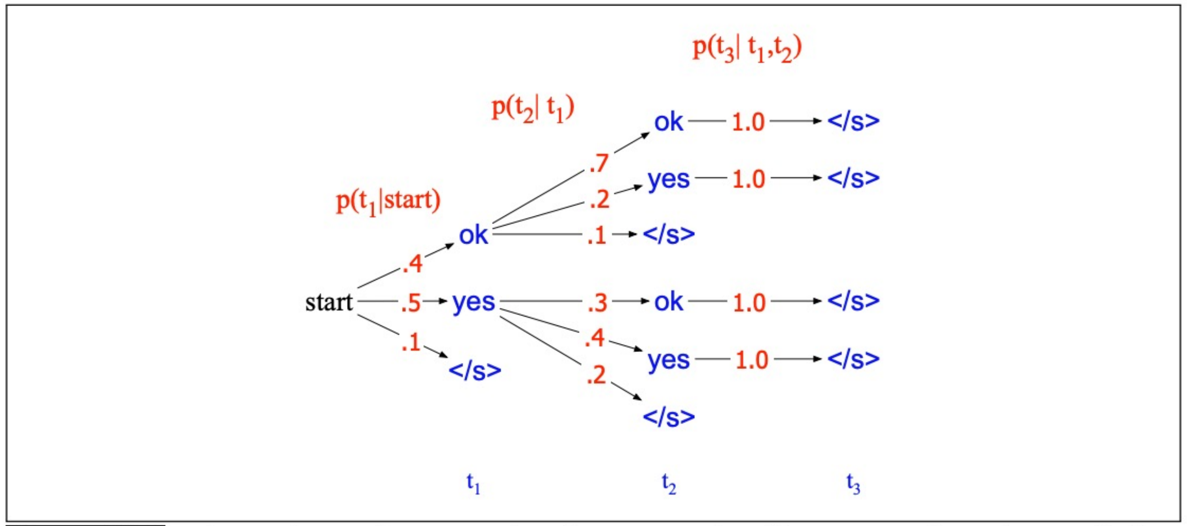


Figure 10.8 A search tree for generating the target string $T = t_1, t_2, \ldots$ from the vocabulary $V = \{\text{yes}, \text{ok}, \text{<s>}\}$, showing the probability of generating each token from that state. Greedy search would choose *yes* at the first time step followed by *yes*, instead of the globally most probable sequence *ok ok*.

Illustration from https://web.stanford.edu/~jurafsky/slp3/10.pdf

Language Generation

- Most probable path is too deterministic
- Top-K sampling
- Nucleus sampling

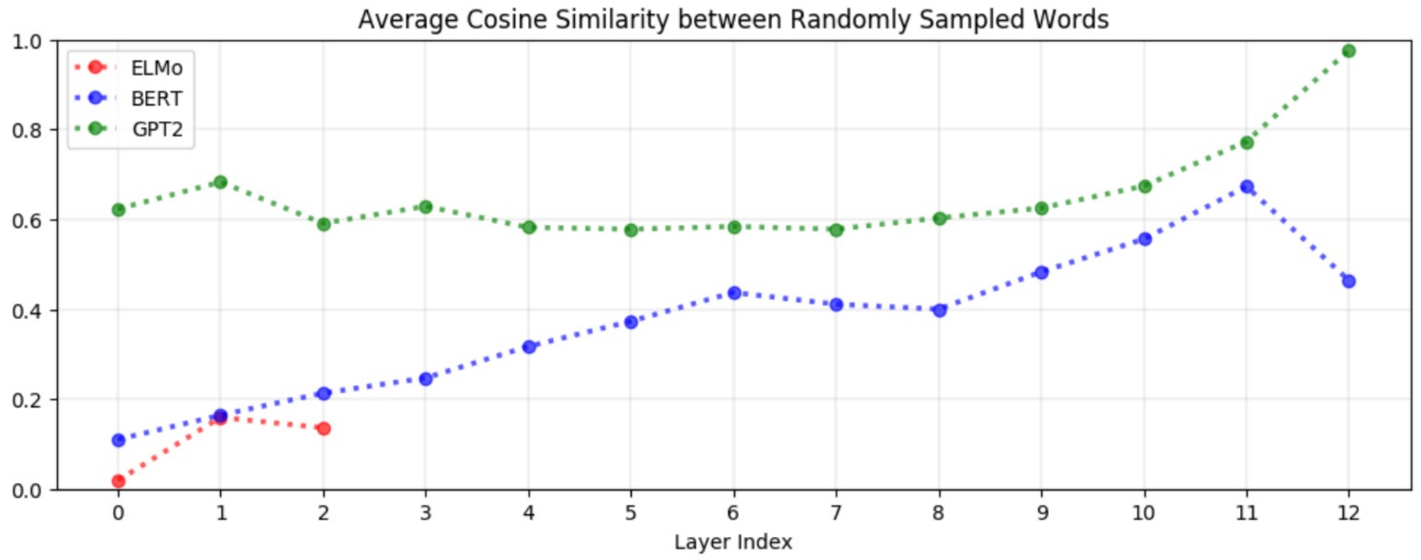
Agenda for Today

- Deep dive of “Nonlinear layers”
  - Recurrent Neural Network (RNN)
  - Architectures
  - Transformer
- Transformer Language Models
- Representations in Transformer

arXiv
https://arxiv.org › cs

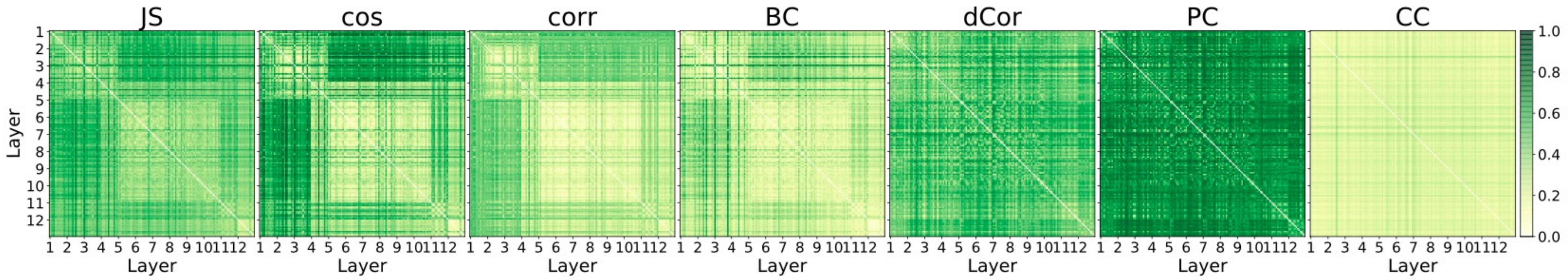
How Contextual are Contextualized Word Representations ...

by K Ethayarajh · 2019 · Cited by 659 — Title:**How Contextual are Contextualized Word Representations**? **Comparing the Geometry of BERT**, ELMo, and GPT-2 Embeddings.



- The representations occupy a small cone

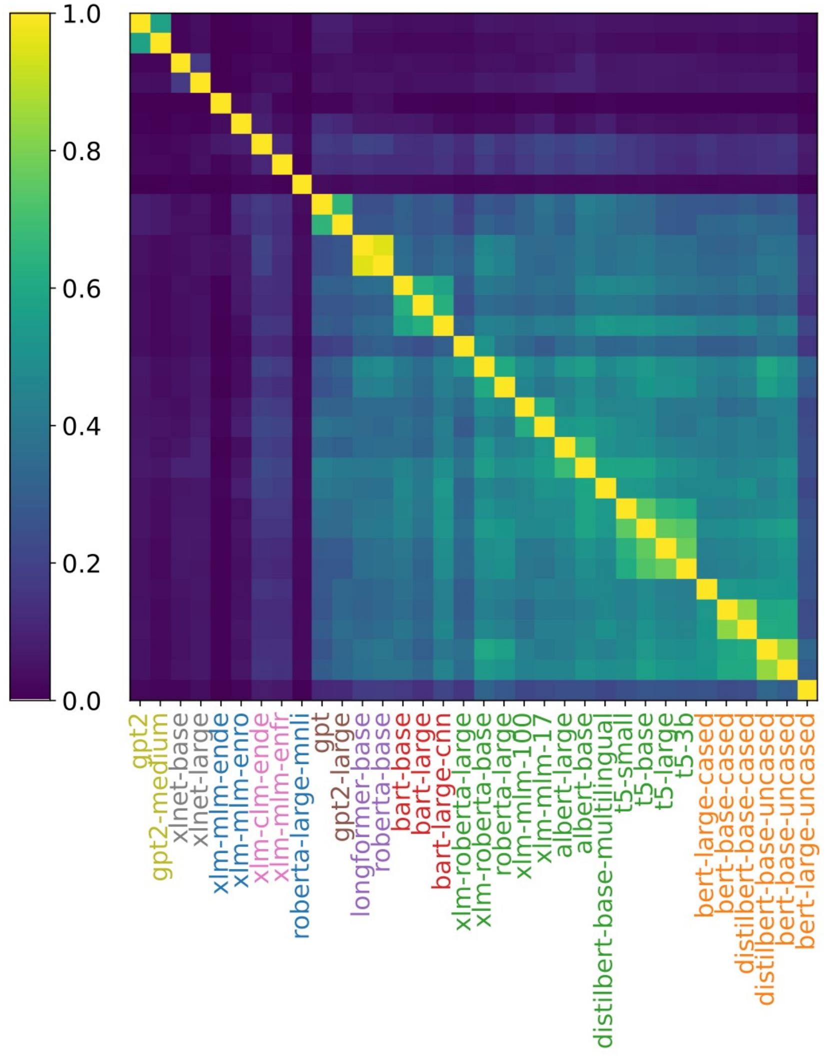
Similarity between Attention Heads



- Consecutive layers are similar
- Some heads are similar
- Motivates pruning of attention heads

Image from https://aclanthology.org/2021.naacl-main.72.pdf

Similarities between Published Models



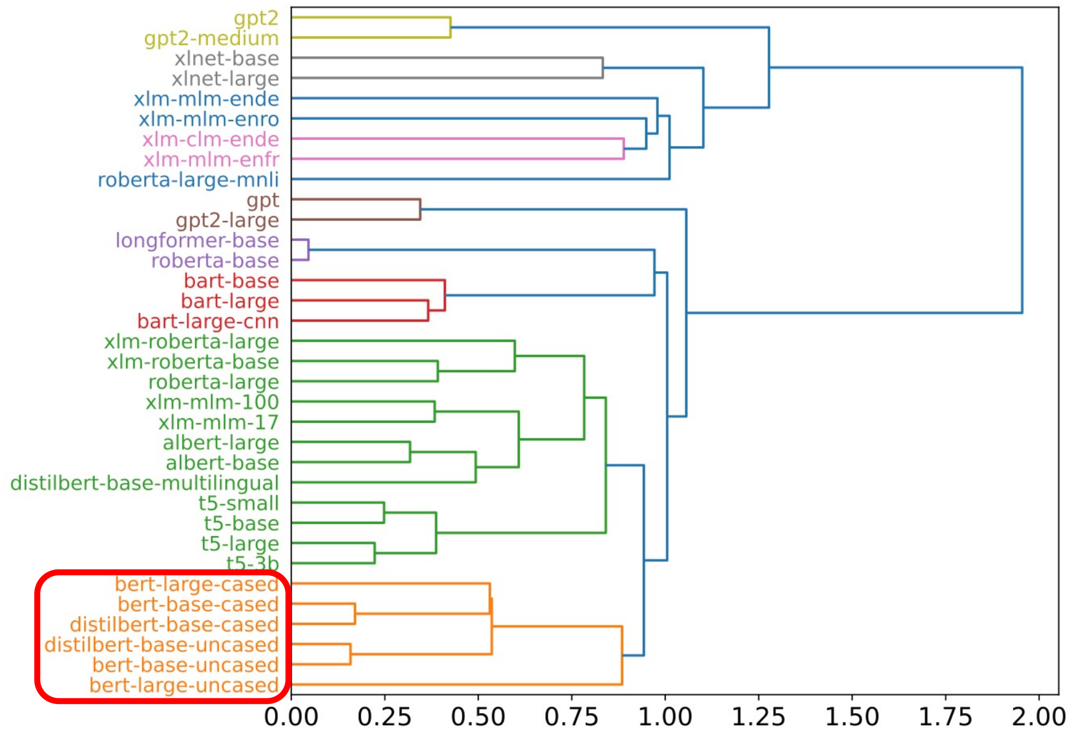


Image from https://proceedings.neurips.cc/paper_files/paper/2021/file/a1c3ae6c49a89d92aef2d423dadb477f-Paper.pdf

Bilingual Lexicon Induction Revisited

- Source embedding $X \in \mathbb{R}^{n\times d}$, target embedding Y $\in \mathbb{R}^{n\times d}$
- Learn a rotation matrix $R \in \mathbb{R}^{d\times d}, RR^T = I$
- Procrustes problem

$$\min_{R:RR^T=I} \|XR - Y\|^2$$

- We can show it's equivalent to solving

$$\max_{R:RR^T=I} \langle R, Y^T X\rangle$$

- Let the SVD of $Y^T X = U\Lambda V^T$, then optimum $R^* = UV^T$