Lecture 2

CME 295: Transformers & Large Language Models

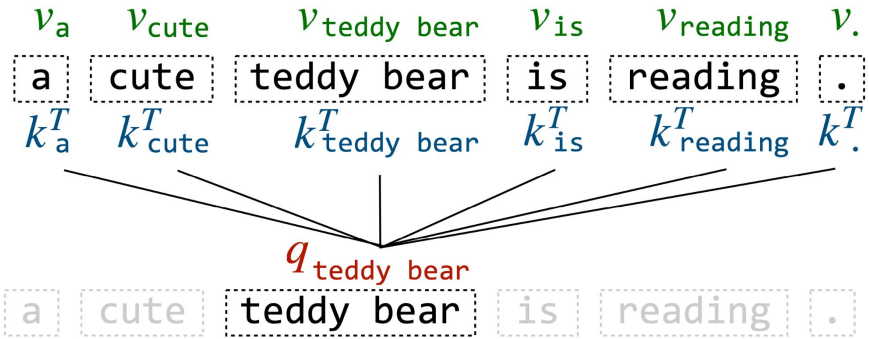
Afshine Amidi & Shervine Amidi

Stanford University

ICME

Recap of last episode…

Recap of last episode...



"*Attention Is All You Need*", Vaswani et al., 2017.

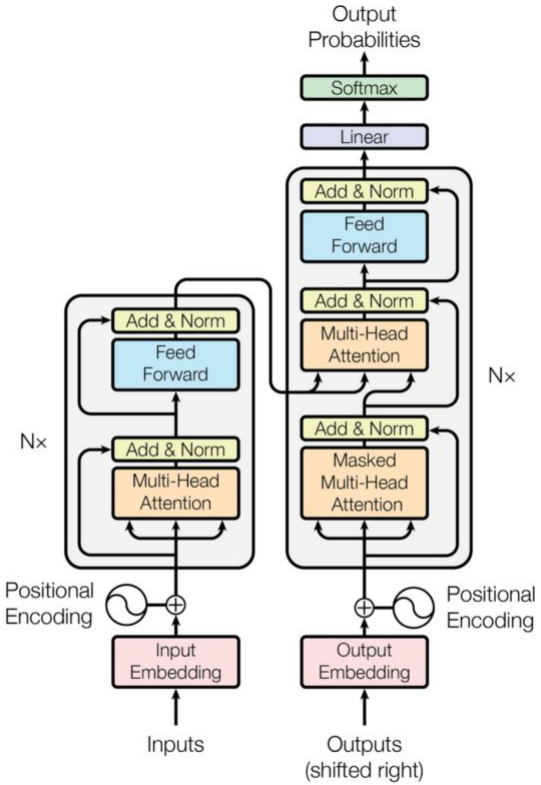
Recap of last episode...

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

"*Attention Is All You Need*", Vaswani et al., 2017.

Recap of last episode...

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



"*Attention Is All You Need*", Vaswani et al., 2017.

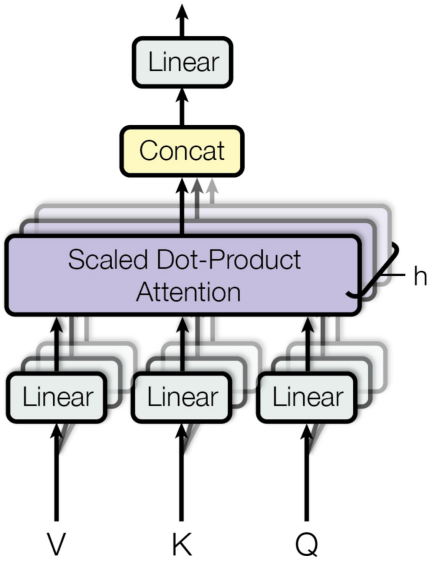
Recap of last episode...

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

"*Attention Is All You Need*", *Vaswani et al., 2017.*

Recap of last episode...

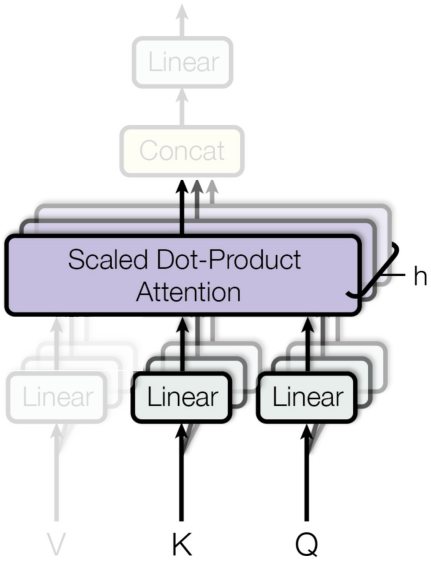
$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



"Attention Is All You Need", Vaswani et al., 2017.

Recap of last episode...

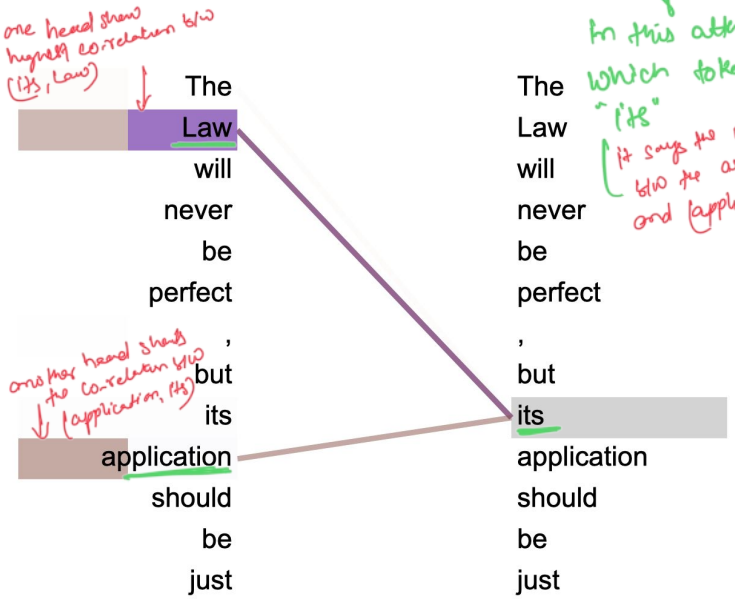
$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



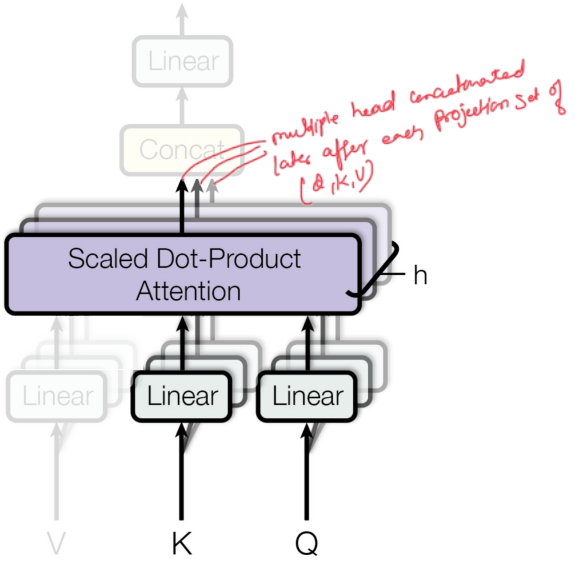
"*Attention Is All You Need*", Vaswani et al., 2017.

Recap of last episode...

Attention map



$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



"Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. [...] Isolated attentions from just the word 'its' for attention heads 5 and 6."

"Attention Is All You Need", Vaswani et al., 2017.

Suggested reading: original Transformer paper

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best

Transformers & Large Language Models

Position embeddings

Layer normalization

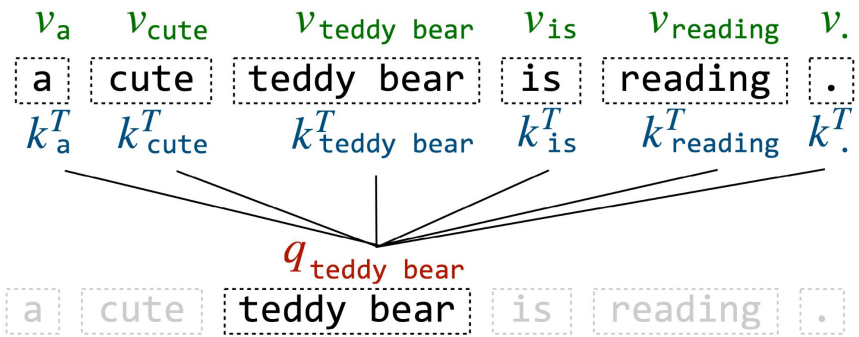
Attention approximation

Transformer-based models

BERT deep dive

Need for position information

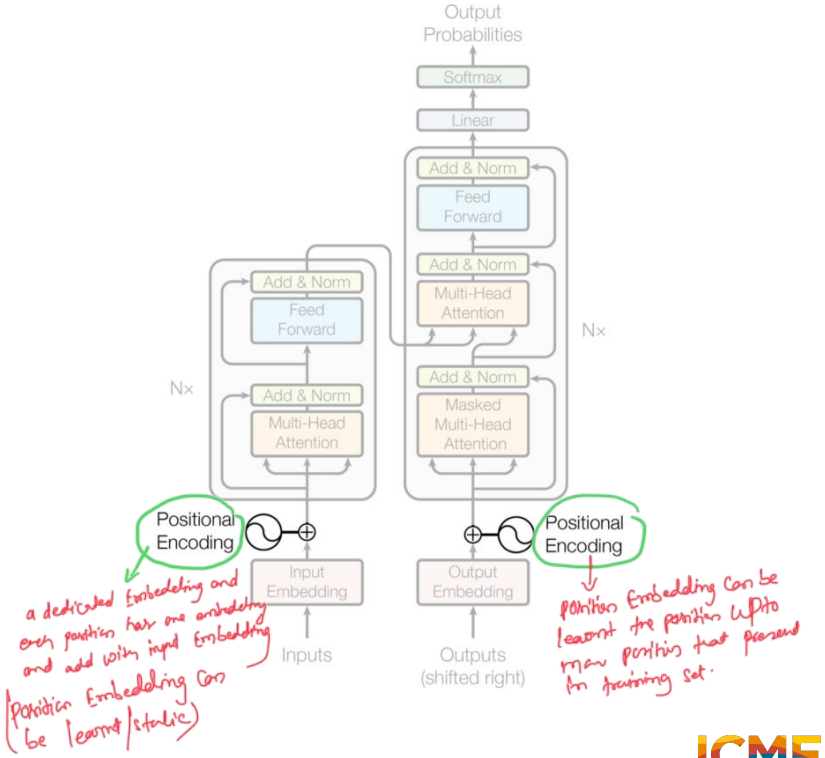
Motivation. Direct links "lose" position info



Need for position information

Motivation. Direct links "lose" position info

Idea. Add learned position-specific embedding to token vector → "one Embedding per position" one way of position Embedding



"Attention Is All You Need", Vaswani et al., 2017.

Need for position information

Motivation. Direct links "lose" position info

Idea. Add learned position-specific embedding to token vector

Limitations. Need to retrain for longer sequences

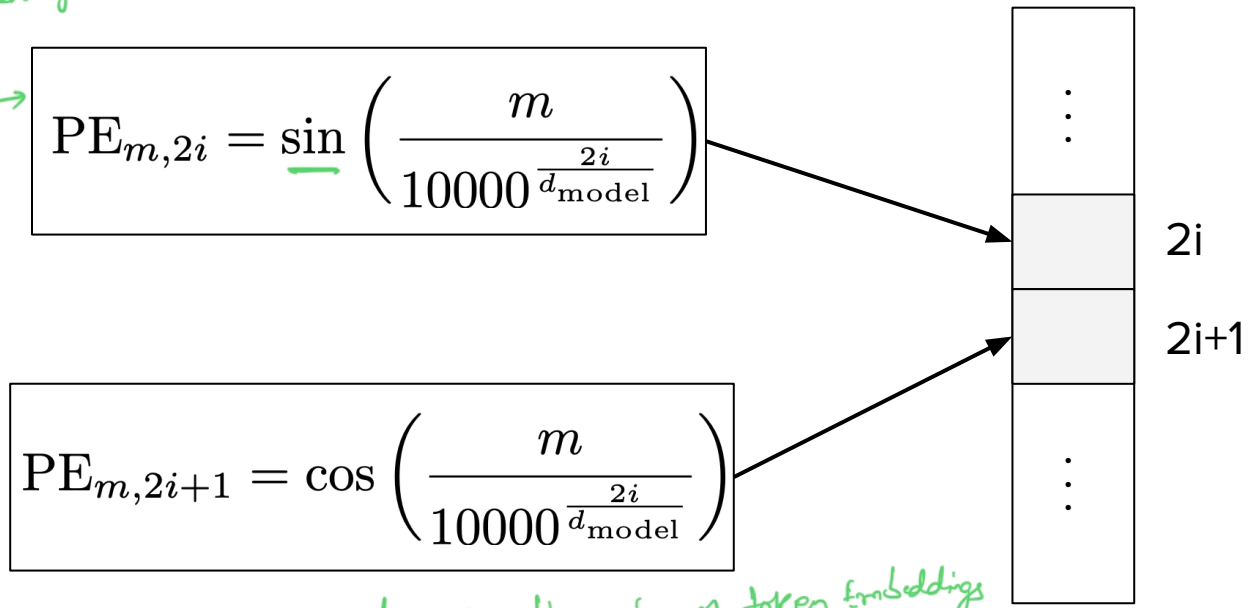
Hardcoded position embeddings

Idea. Hardcode values of position embedding → on another way of position Embedding using sin/cos

for every Embedding

$$\mathrm{PE}_{m,2i} = \sin\left(\frac{m}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

$$\mathrm{PE}_{m,2i+1} = \cos\left(\frac{m}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

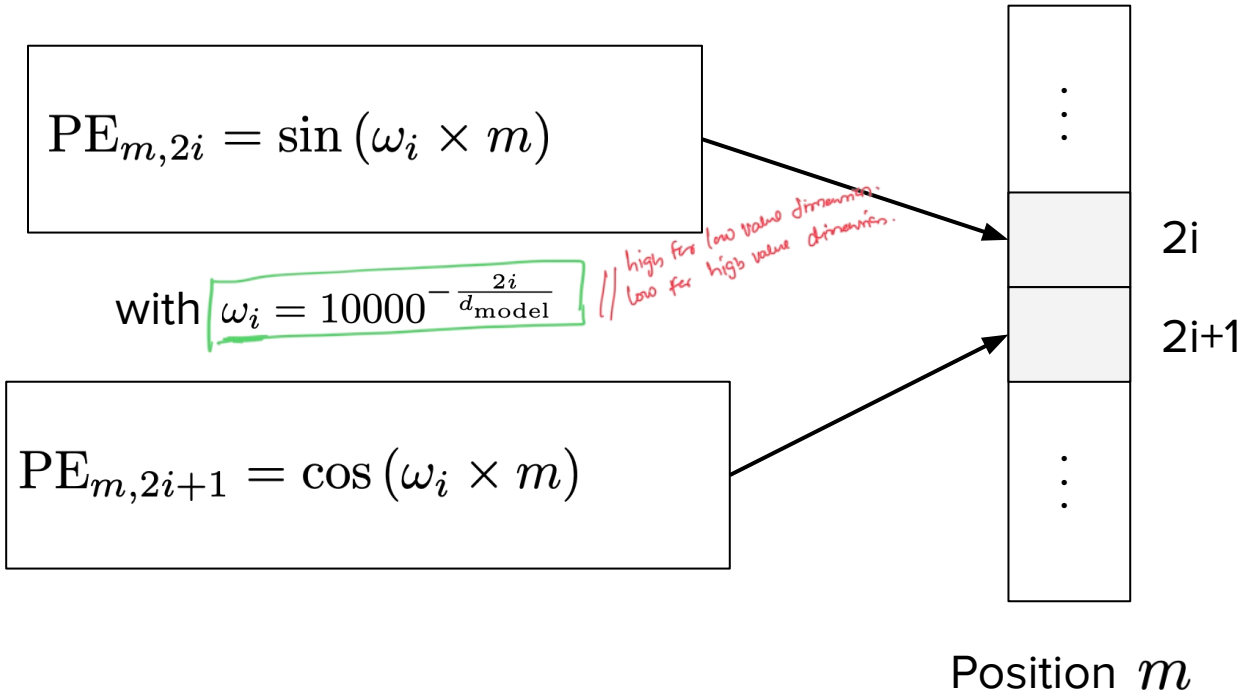


d_{model} : dimension of token embeddings

"Attention Is All You Need", Vaswani et al., 2017.

Hardcoded position embeddings

Idea. Hardcode values of position embedding



"Attention Is All You Need", Vaswani et al., 2017.

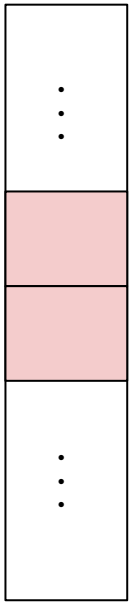
Hardcoded position embeddings

Idea. Hardcode values of position embedding

So using cosine/sine; it tries to find the relationship b/w embedding that are closer. Embedding will have close angle"
cosine - relation b/w (nth position, mth position) for embedding i

$$\mathrm{PE}_{m,2i} = \sin\left(\omega_i \times m\right)$$

$$\mathrm{PE}_{m,2i+1} = \cos\left(\omega_i \times m\right)$$



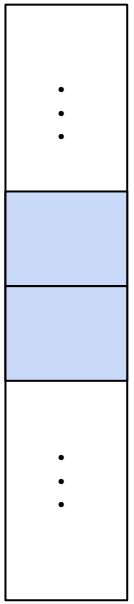
2i

2i+1

Position m

$$\mathrm{PE}_{n,2i} = \sin\left(\omega_i \times n\right)$$

$$\mathrm{PE}_{n,2i+1} = \cos\left(\omega_i \times n\right)$$



2i

2i+1

Position n

"Attention Is All You Need", Vaswani et al., 2017, "Transformer Architecture: The Positional Encoding", by Kazemnejad, 2019.

Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$$

Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$$

$$\cos(\omega_i(m-n)) = \cos(\omega_i \times m)\cos(\omega_i \times n) + \sin(\omega_i \times m)\sin(\omega_i \times n)$$

Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$$

$$\cos(\omega_i(m-n)) = \cos(\omega_i \times m)\cos(\omega_i \times n) + \sin(\omega_i \times m)\sin(\omega_i \times n)$$

$$\langle \mathrm{PE}_m, \mathrm{PE}_n \rangle = \ldots + \cos(\omega_i(m-n)) + \ldots$$

It calculates cosine similarity b/w (mth position, nth position) embedding i

Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$$

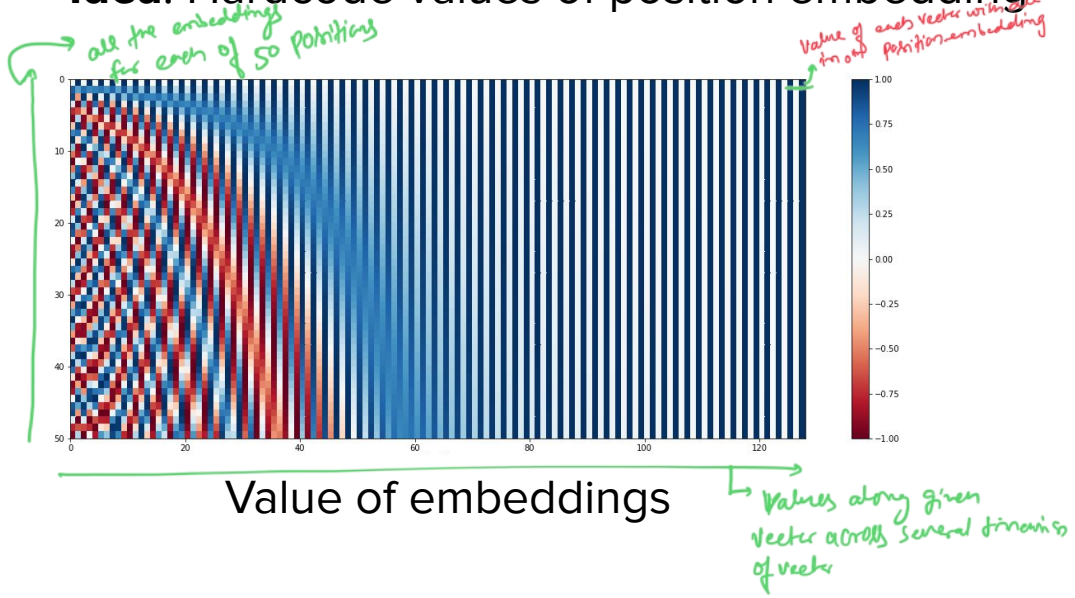
$$\cos(\omega_i(m-n)) = \cos(\omega_i \times m)\cos(\omega_i \times n) + \sin(\omega_i \times m)\sin(\omega_i \times n)$$

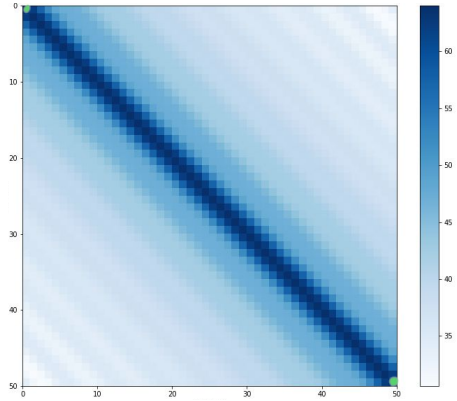
$$\langle \mathrm{PE}_m, \mathrm{PE}_n \rangle = \ldots + \cos(\omega_i(m-n)) + \ldots$$

$$\langle \mathrm{PE}_m, \mathrm{PE}_n \rangle = f(m-n)$$

Hardcoded position embeddings

Idea. Hardcode values of position embedding





Similarity between positions

"Attention Is All You Need", Vaswani et al., 2017, "Transformer Architecture: The Positional Encoding", by Kazemnejad, 2019.

Hardcoded position embeddings

Idea. Hardcode values of position embedding

Value of embeddings

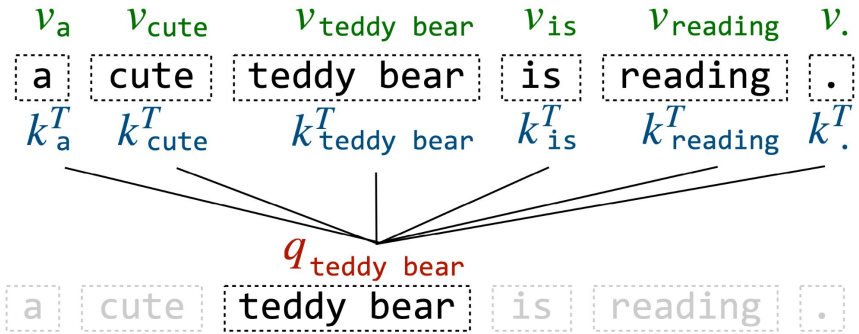
Similarity between positions

Benefits. Can extend to any sequence length

"Attention Is All You Need", Vaswani et al., 2017, "Transformer Architecture: The Positional Encoding", by Kazemnejad, 2019.

From absolute to relative position info

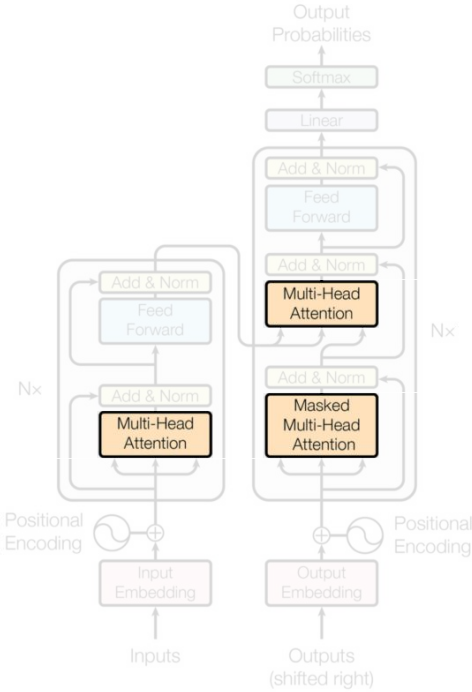
Motivation. Actually, we really care about relative position **with attention layer**



Position Embedding does focus/learn for positions but need to add with attention layer bc we care about finding the similarity in tokens in self-attention computation. Why not? integrating self-attention layer with position embeddings.

From absolute to relative position info

Motivation. Actually, we really care about relative position **with attention layer**

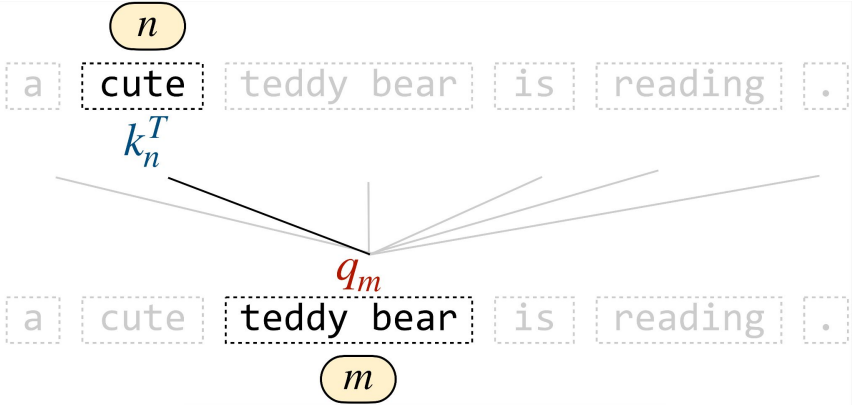


Idea. Let's change the attention layer instead

"*Attention Is All You Need*", Vaswani et al., 2017.

Linear bias in attention layer

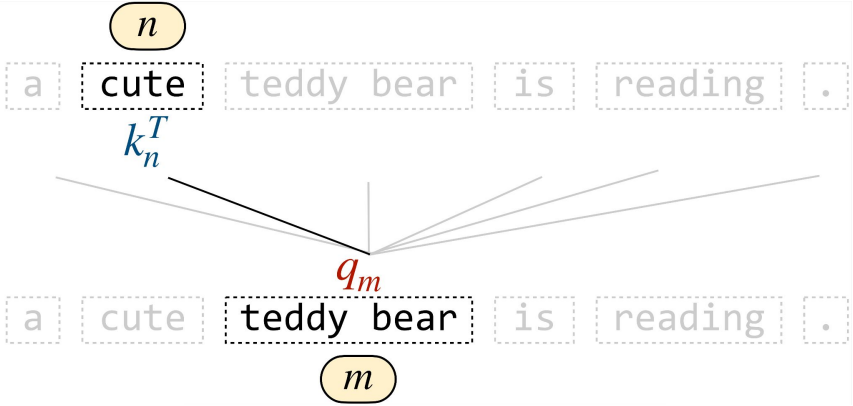
Idea. Add bias to query-key scores



$$\text{softmax}\left(\frac{\langle q_m, k_n \rangle}{\sqrt{d_k}} + \text{bias}(m, n)\right)$$

Linear bias in attention layer

Idea. Add bias to query-key scores



$$\text{softmax}\left(\frac{\langle q_m, k_n \rangle}{\sqrt{d_k}} + \text{bias}(m,n)\right)$$

it basically says that some tokens are more similar compared to other tokens.

T5 bias. Bias is **learned per head**

(1) approach

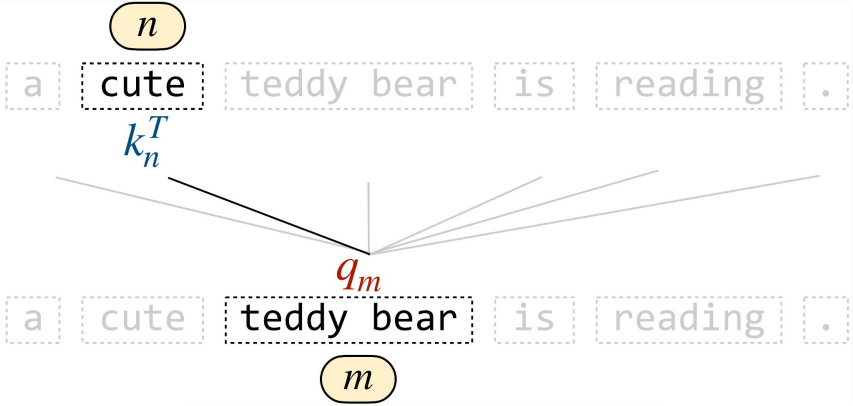
$$\text{bias}(m,n) = \beta_{\text{bucket}(m-n)}$$

bias(m,n) is getting learnt using the concept of Bucketisation of all (m,n) into some buckets and learn which bucket will be best fit for injecting into softmax.

"Super Study Guide: Transformers & Large Language Models", by Amidi, 2024.
"Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", Raffel et al., 2023.

Linear bias in attention layer

Idea. Add bias to query-key scores



$$\text{softmax}\left(\frac{\langle q_m, k_n \rangle}{\sqrt{d_k}} + \text{bias}(m,n)\right)$$

T5 bias. Bias is **learned per head**

$$\text{bias}(m,n) = \beta_{\text{bucket}(m-n)}$$

ALiBi. Bias is **linear**, **deterministic**, not bounded

② approach

$$\text{bias}(m,n) = \mu \times (n-m)$$

attention with linear biases

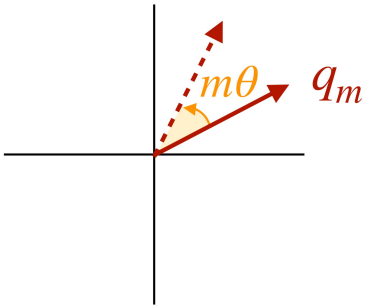
↳ instead of learning these biases, let's have deterministic formula.

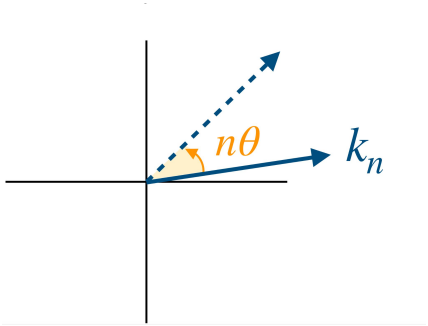
"Super Study Guide: Transformers & Large Language Models", by Amidi, 2024.
"Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation", Press et al., 2021.

Default choice nowadays: rotations in attention layer

important | default choice for attention layer

Idea. Rotate query and key vectors with rotation matrix





"RoFormer: Enhanced Transformer with Rotary Position Embeddings", Su et al., 2021.
Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Default choice nowadays: rotations in attention layer

Rotary Position Embeddings.

Idea. Rotate query and key vectors with rotation matrix

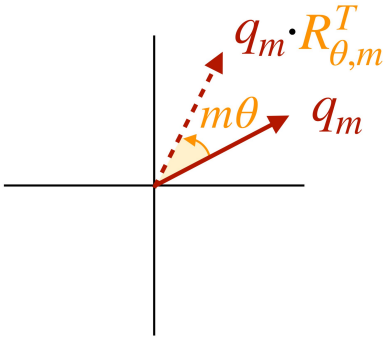
Formula. Use rotation matrix: $R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$

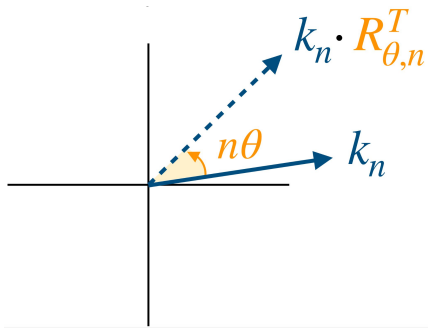
"RoFormer: Enhanced Transformer with Rotary Position Embeddings", Su et al., 2021.
Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Default choice nowadays: rotations in attention layer

an ideal position embeddings ← [**RoPE** = **Ro**tary **P**osition **E**mbeddings]

Idea. Rotate query and key vectors with rotation matrix — find up with quantity that will be function will be relative distance b/w two.





Formula. Use rotation matrix: $R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$

"RoFormer: Enhanced Transformer with Rotary Position Embeddings", Su et al., 2021.
Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Default choice nowadays: rotations in attention layer

Extension to dimension d > 2. Rotate every block of dimension 2

$$R_{\theta,m} = \begin{pmatrix} \text{block}_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \text{block}_{\frac{d_k}{2}} \end{pmatrix} \qquad \text{block}_i = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix}$$

"*RoFormer: Enhanced Transformer with Rotary Position Embeddings*", Su et al., 2021.

Default choice nowadays: rotations in attention layer

Extension to dimension d > 2. Rotate every block of dimension 2

$$R_{\theta,m} = \begin{pmatrix} \text{block}_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \text{block}_{\frac{d_k}{2}} \end{pmatrix} \qquad \text{block}_i = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix}$$

Benefits. Relative distance nicely captured:

$$q_m k_n^T = x_m W_q R_{\theta,n-m} W_k^T x_n^T$$

rotating key by angle n

rotation matrix

rotating query by angle m

Default choice nowadays: rotations in attention layer

Extension to dimension d > 2. Rotate every block of dimension 2

$$R_{\theta,m} = \begin{pmatrix} \text{block}_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \text{block}_{\frac{d_k}{2}} \end{pmatrix} \qquad \text{block}_i = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix}$$

Benefits. Relative distance nicely captured:

$$q_m k_n^T = x_m W_q R_{\theta,n-m} W_k^T x_n^T$$

Attention weight has a long-term decay

Relative upper bound of $A(q_m k_n^T)$

attention weight b/w query in Boyle m and key in angle n

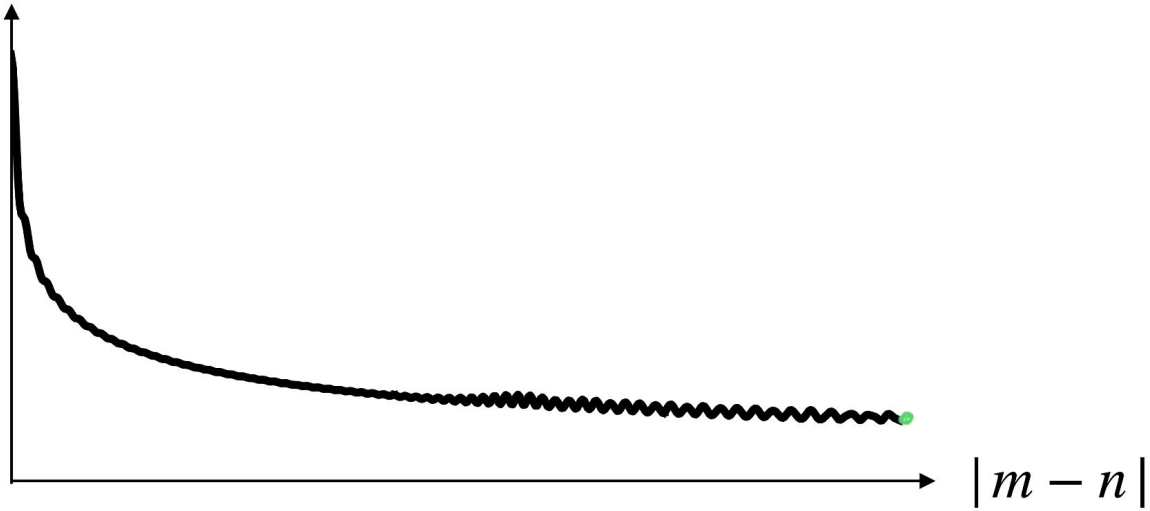


 Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Transformers & Large Language Models

Position embeddings

Layer normalization

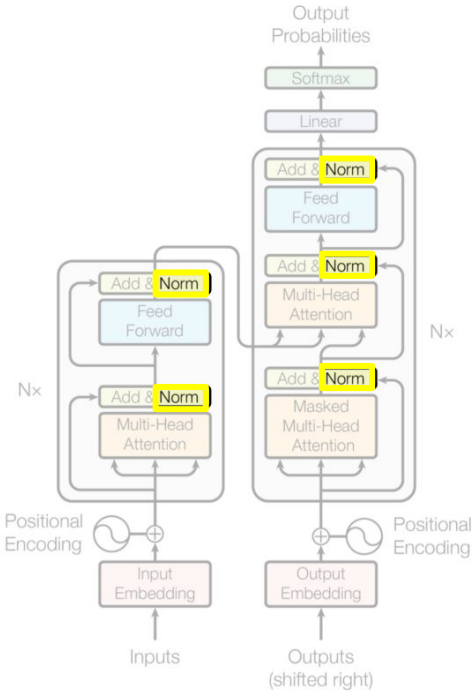
used for normalizing components of vector.

Attention approximation

Transformer-based models

BERT deep dive

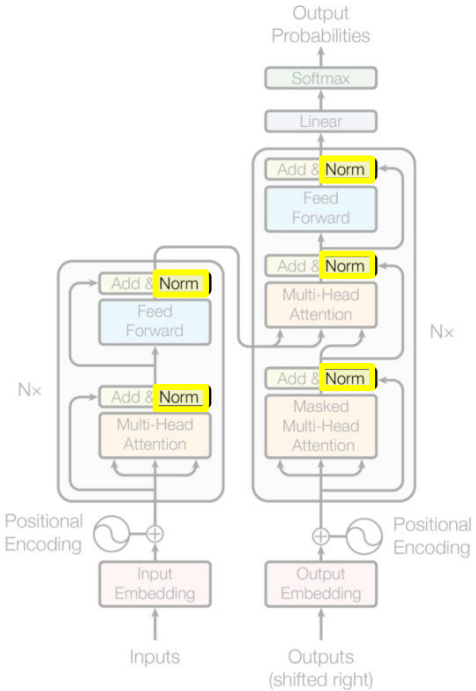
Layer normalization



"*Attention Is All You Need*", *Vaswani et al., 2017.*

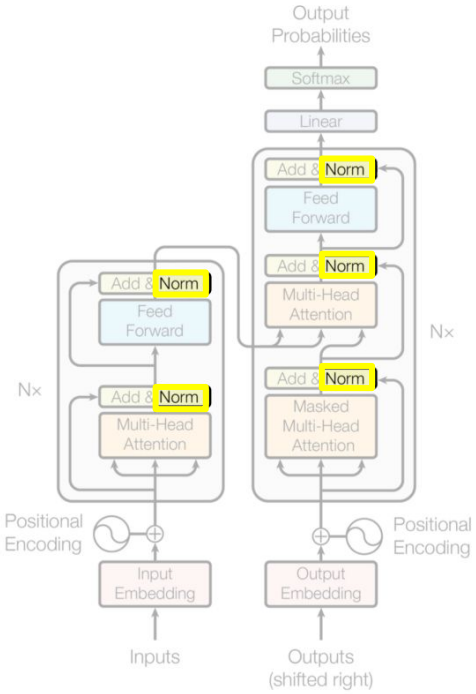
Layer normalization

LN = **L**ayer **N**ormalization



"*Attention Is All You Need*", *Vaswani et al., 2017.*

Layer normalization



LN = **L**ayer **N**ormalization

helps training stability and convergence time

Learnable parameters.

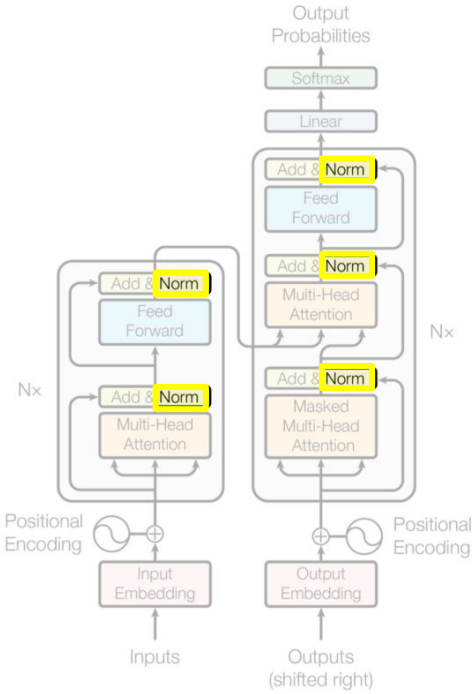
$$\boxed{\mathrm{LN}(x) = \gamma\hat{x} + \beta} \quad \text{with} \quad \hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

where $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$ and $\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$

"Attention Is All You Need", Vaswani et al., 2017.

Layer normalization

LN = **L**ayer **N**ormalization



$$\boxed{\mathrm{LN}(x) = \gamma\hat{x} + \beta} \quad \text{with} \quad \hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

where $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$ and $\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$

Benefits. Helps with training stability and convergence.

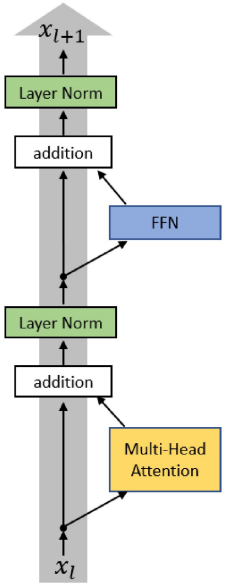
"Attention Is All You Need", Vaswani et al., 2017. "Layer Normalization", Ba et al., 2016.

Different kinds of layer normalization

Post-Norm

$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$

SubLayer(x) can be attention layer / feed forward nn layer.



After

residual connection

"On Layer Normalization in the Transformer Architecture", Xiong et al., 2020.

Different kinds of layer normalization

Post-Norm

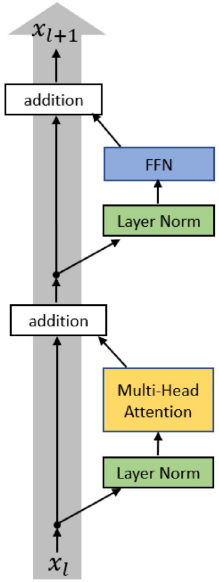
$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$

After

residual connection

Pre-Norm

$$\text{Output} = x + \text{SubLayer}(\text{LayerNorm}(x))$$



Before

residual connection

"On Layer Normalization in the Transformer Architecture", Xiong et al., 2020.

Different kinds of layer normalization

Nowadays:

Pre-Norm

"On Layer Normalization in the Transformer Architecture", Xiong et al., 2020.

Different kinds of layer normalization

Nowadays:

Pre-Norm +

RMSNorm

$$\gamma\hat{x} + \beta \longrightarrow \gamma \frac{x}{\mathrm{RMS}(x)}$$

instead of using layernorm; it is also useful to use for benefits of fewer parameters to learn; in layer norm, we learn γ, β but here we have γ to learn

"On Layer Normalization in the Transformer Architecture", Xiong et al., 2020. "Root Mean Square Layer Normalization", Zhang et al., 2019.

Transformers & Large Language Models

Position embeddings

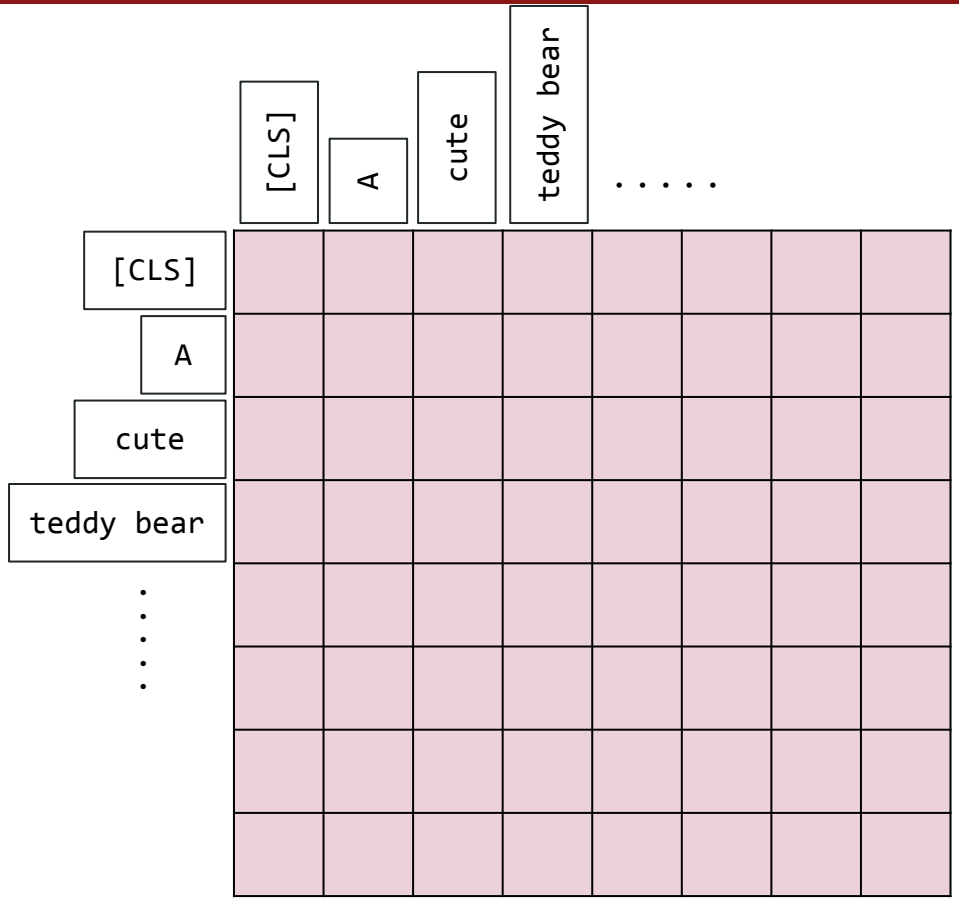
Layer normalization

Attention approximation

Transformer-based models

BERT deep dive

Sparse attention: Longformer



if when each embedding will attend each embedding for attention mechanism calculation then basically it becomes n^2 calculations so, to get rid of this mucn of calculation, the attention can be approximated. using Longformer

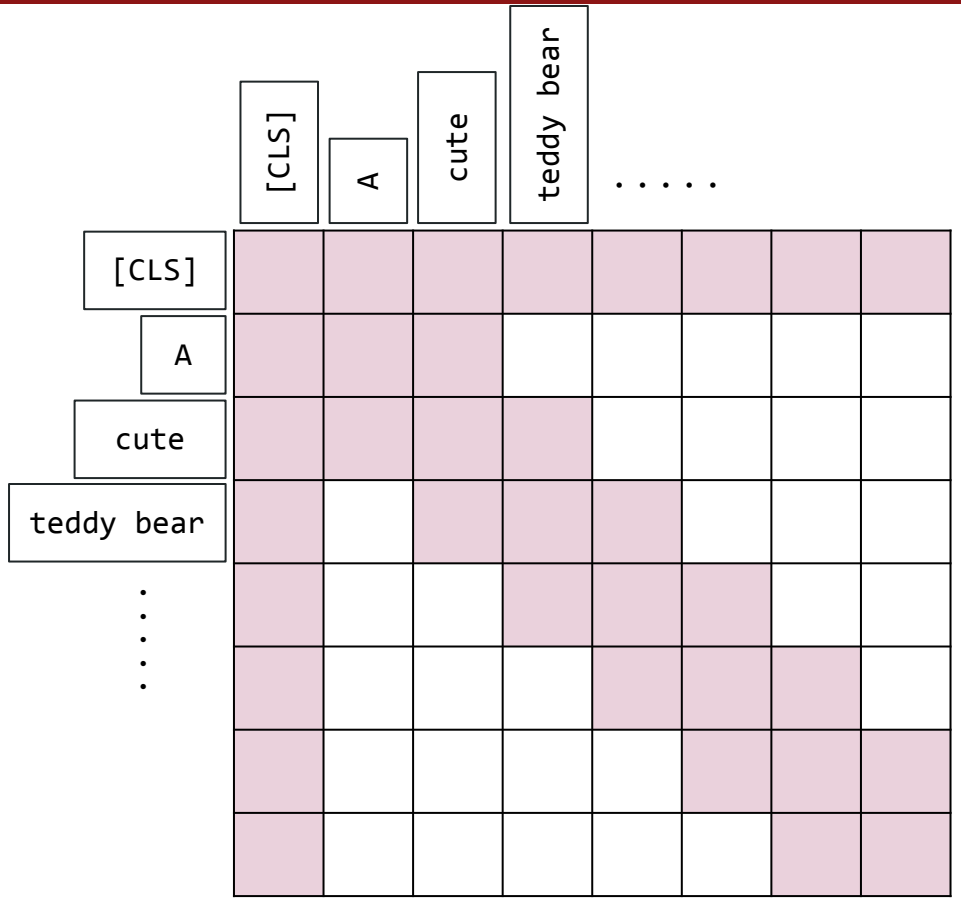
it restricts the window of interest it only allows neighbour embedding to include into attention mechanism and remaining approximated using sliding window Attention mechanism.

"Longformer: The Long-document Transformer", Beltagy et al., 2020.

Sparse attention: Longformer

So basically sliding window

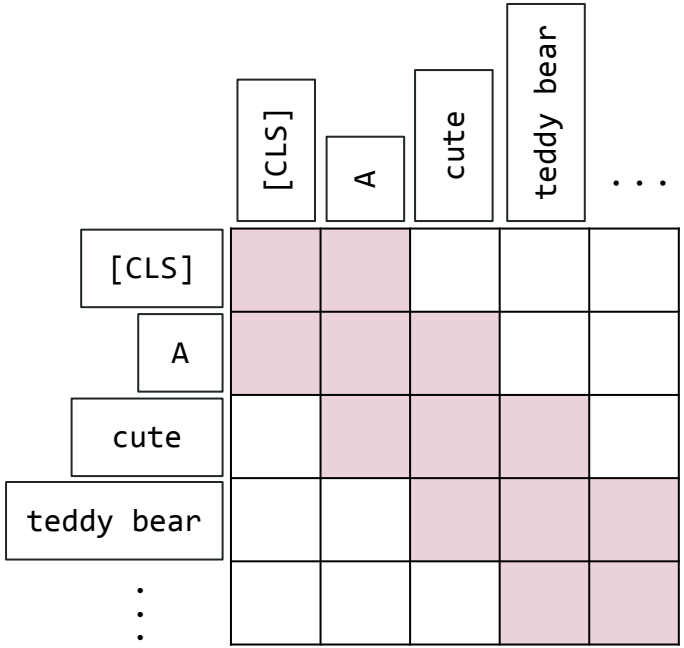
attention mechanism gives the window size of embeddings to be attended. (like receptive field in computer vision)



"Longformer: The Long-document Transformer", Beltagy et al., 2020.

Leveraging local and global attention

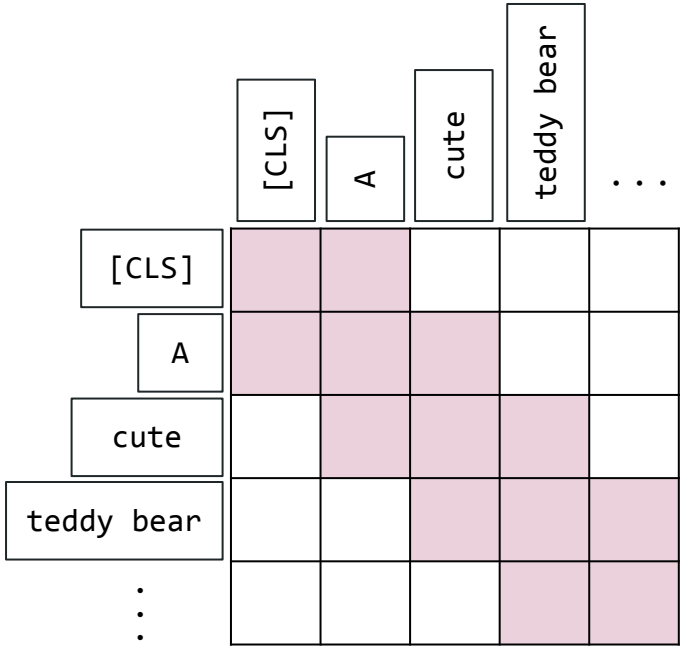
SWA = **S**liding **W**indow **A**ttention

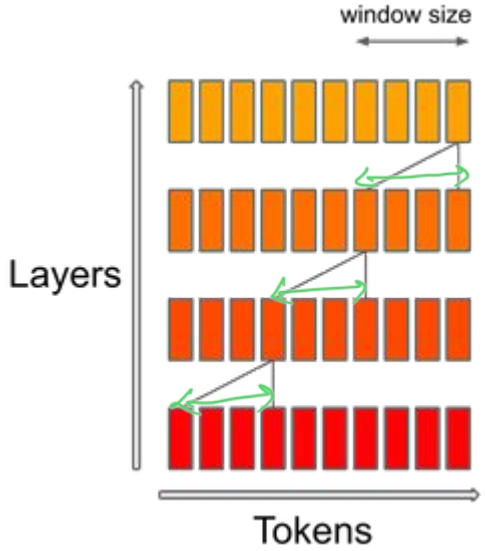


Variations include **interleaving local** and **global attention layers**

Leveraging local and global attention

SWA = **S**liding **W**indow **A**ttention



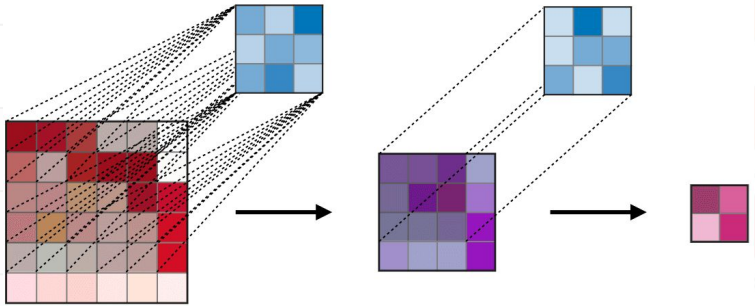


Mistral 7B

Right figure from Mistral 7B announcement, 2023.

Leveraging local and global attention

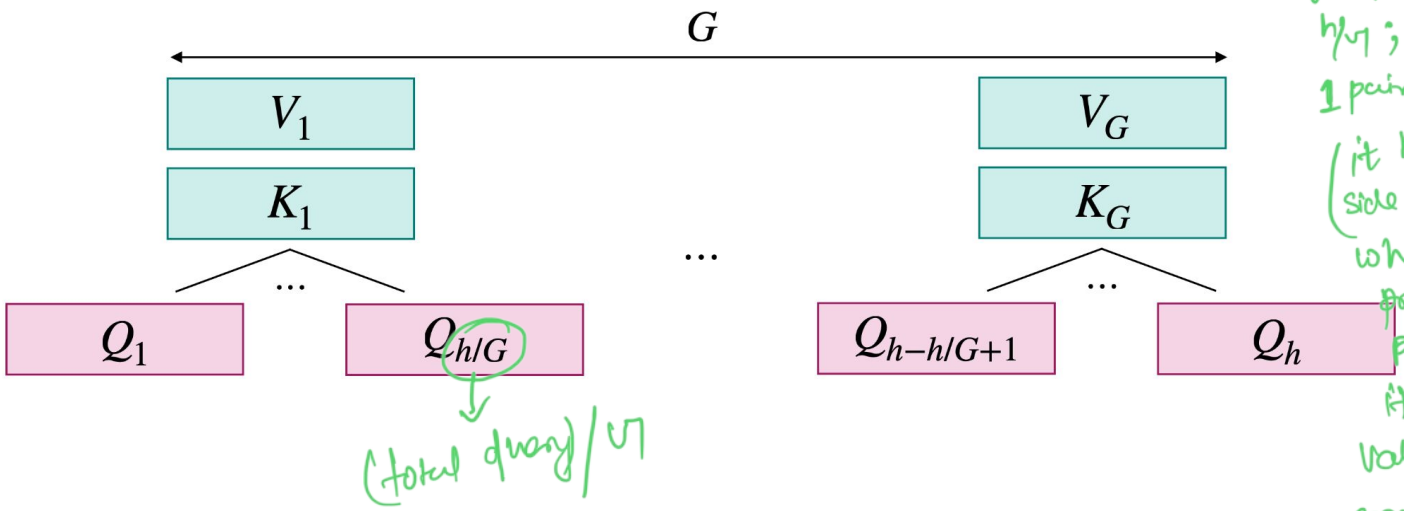
Similar to "**receptive field**" in computer vision



"VIP Cheatsheet: Convolutional Neural Networks", Amidi, 2018.

Sharing attention heads

Idea. Share key/value attention heads within groups of queries



having h heads but grouped into G; which h/G; and it share with 1 pair of value and key. (it helps in decoding side of architecture, when next word is predicted based on previous word; and it repeats so, some value/key pair repeats and for getting rid of repeating some k/v it share with specific set

(total query)/G

Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

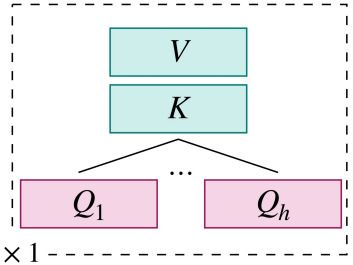
Sharing attention heads

of query.

$G = 1$

Multi-**Q**uery **A**ttention (MQA)

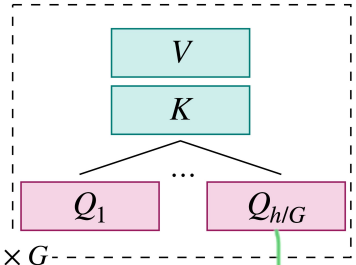
All h head share all the same projection matrix for V, K



multi query projection share the same V, K projection matrix

Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Sharing attention heads

$G = 1$	**M**ulti-**Q**uery **A**ttention (MQA)	
$1 < G < h$	**G**roup-**Q**uery **A**ttention (GQA)	

group of projection of query share V, K

Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Sharing attention heads

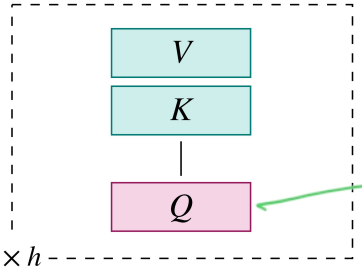
$G = 1$	**M**ulti-**Q**uery **A**ttention (MQA)	
$1 < G < h$	**G**roup-**Q**uery **A**ttention (GQA)	
$G = h$	**M**ulti-**H**ead **A**ttention (MHA)	

Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Transformers & Large Language Models

Attention approximation

Position embeddings

Layer normalization

Transformer-based models

BERT deep dive

Overview of Transformer-based models

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5

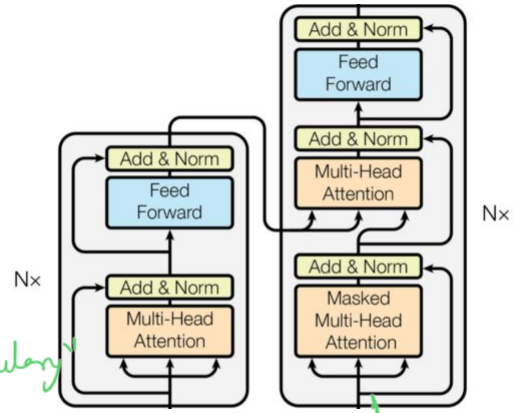


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Overview of Transformer-based models

it decodes the masking tasks

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa

classificatn task i.e. it uses only Encoder parts for transformers

only Encoder parts is being used for classificatn task

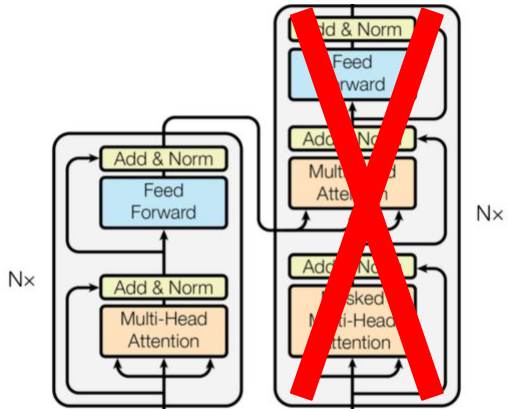


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Overview of Transformer-based models

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series

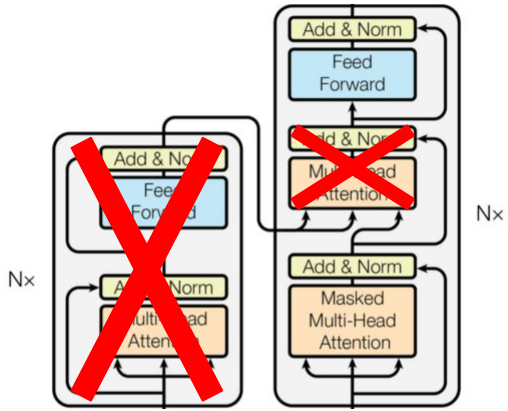
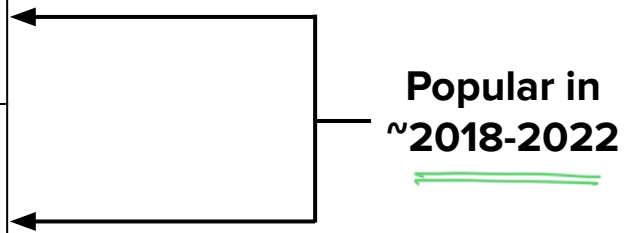


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Overview of Transformer-based models

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series



Overview of Transformer-based models

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series

Popular now!

Transformers & Large Language Models

Attention approximation

Position embeddings

Layer normalization

Transformer-based models

BERT deep dive

Why "BERT"?

BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Encoder only architecture used for classification task

"*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*", Devlin et al., 2018.

Why "BERT"?

BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Why "BERT"?

BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

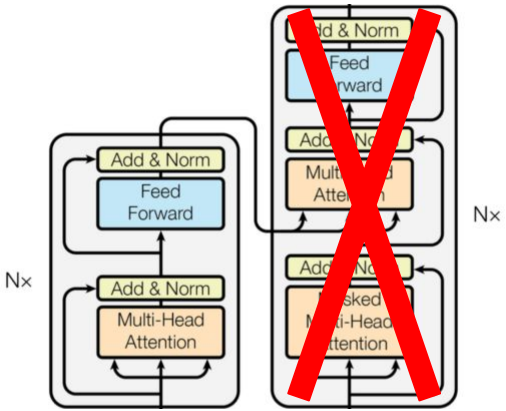


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Why "BERT"?

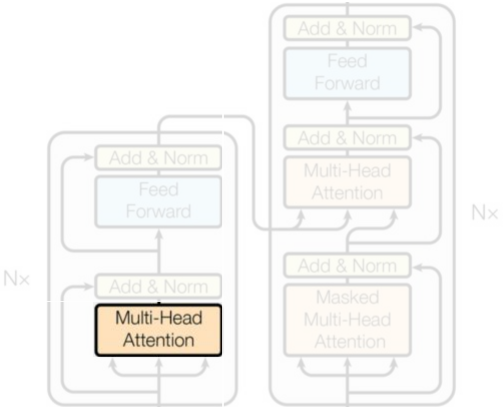
BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

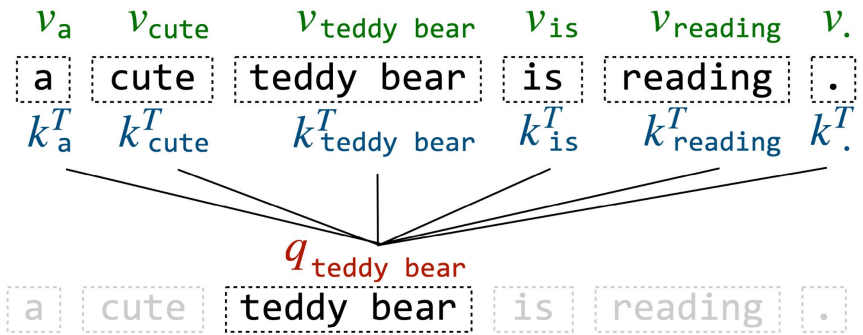
Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Why "BERT"?

BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Bidirectional meaning in BERT architecture is because all token attend all tokens to the Representations



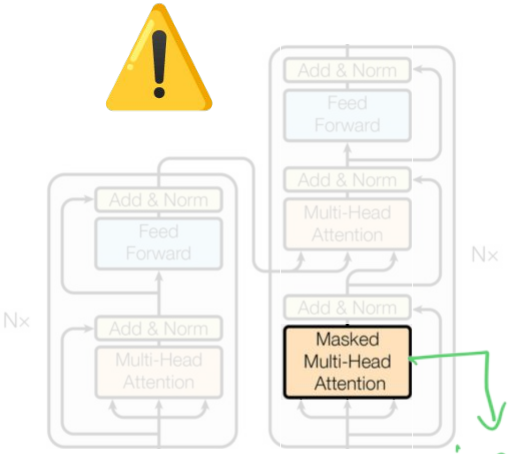


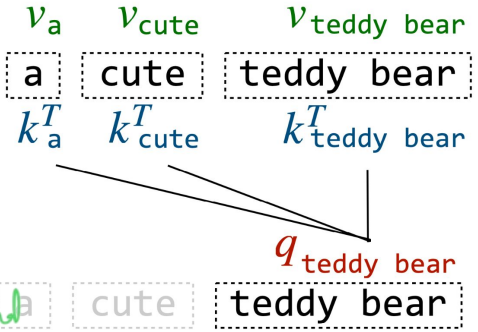
Bidirectional

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Why "BERT"?

BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers



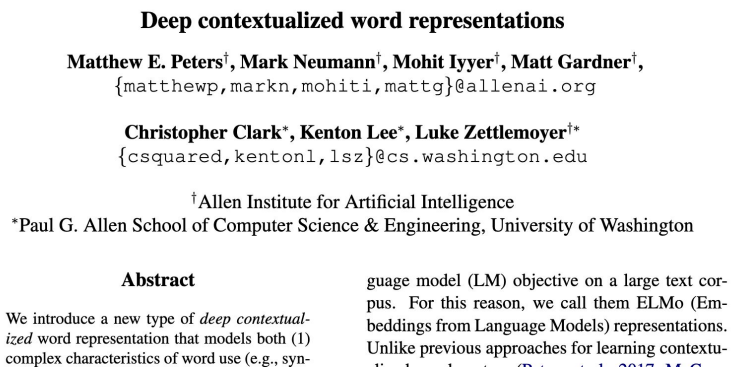


Not bidirectional

is not bidirectional (bcz it is masked attention)
↳ it is basically used for preventing the linking b/w tokens
so, if link is being broken that's why it's not bidirectional.

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Why "BERT"?



Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

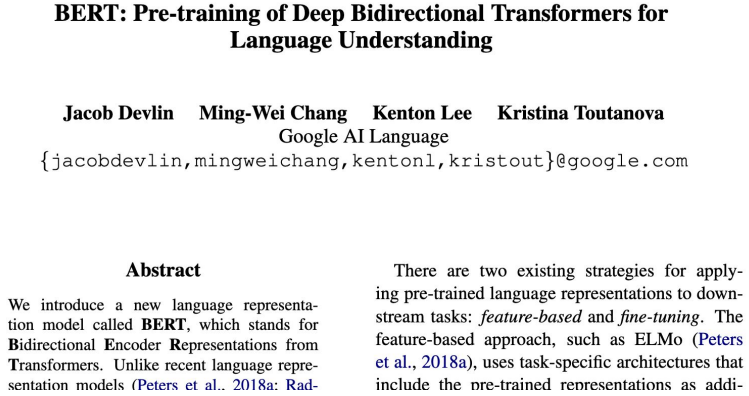
[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syn-

guage model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextu-

ELMo paper
[submitted in Feb 2018]



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language repre-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as addi-

BERT paper
[submitted in Oct 2018]

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Devlin et al., 2018. [BERT paper]
"Deep contextualized word representations", Peters et al., 2018 [ELMo paper]

Why "BERT"?

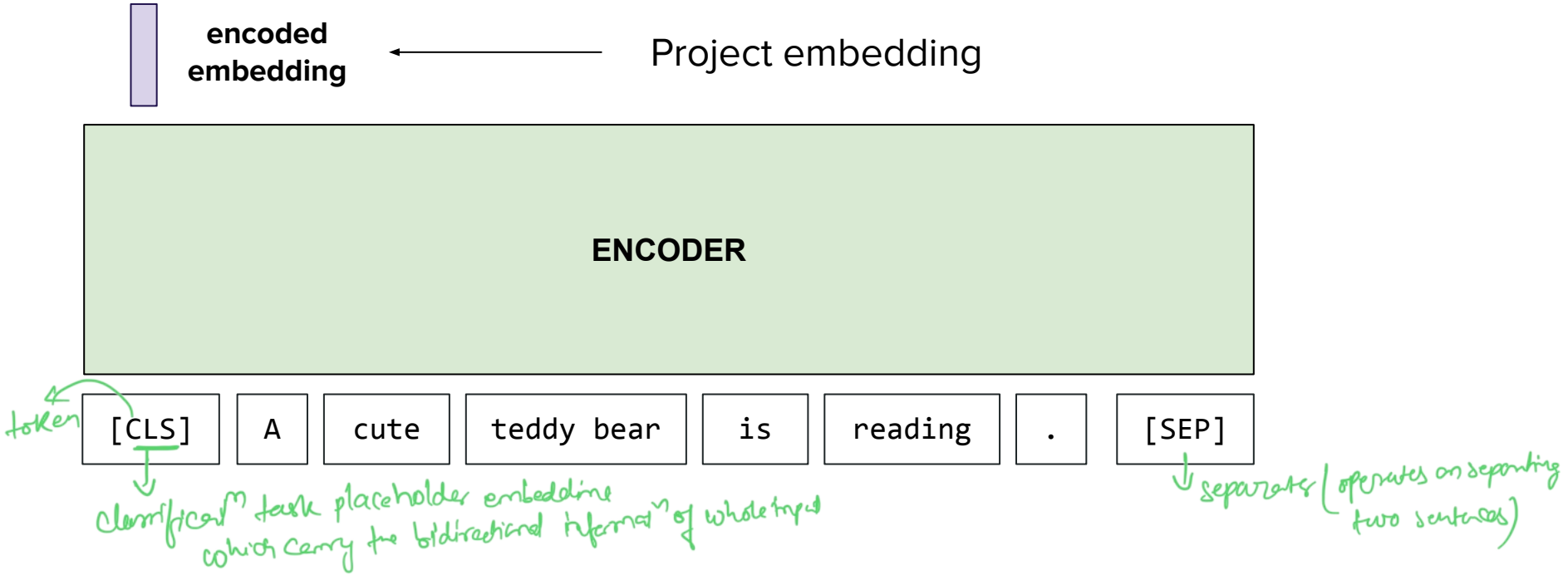
Elmo

Bert

For more background: Sesame Street, Wikipedia.

BERT and encoder-only models

Idea. Keep only **encoders** + leverage **encoded embeddings** to make predictions.



"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Devlin et al., 2018.

BERT and encoder-only models

Strategy.

- **Step 1**: Pretraining with proxy tasks (MLM and NSP)
- **Step 2**: Finetuning for given end task

"*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*", Devlin et al., 2018.

BERT and encoder-only models

Strategy.

- **Step 1**: Pretraining with proxy tasks (MLM and NSP)
- **Step 2**: Finetuning for given end task

Discussion.

Pros	Cons
• Finetuning does not need a lot of data • Good performance	• Not suited for a range of tasks (e.g. text generation) • Finetuning is a required step

BERT and encoder-only models

Strategy.

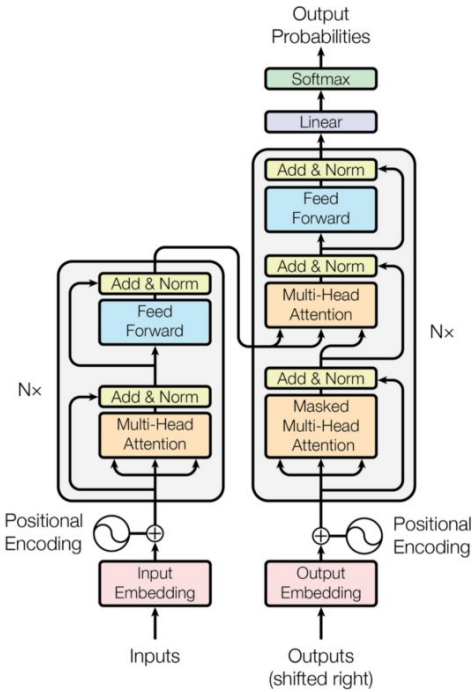
- **Step 1**: Pretraining with proxy tasks (MLM and NSP)
- **Step 2**: Finetuning for given end task

Discussion.

Pros	Cons
• Finetuning does not need a lot of data • Good performance	• Not suited for a range of tasks (e.g. text generation) • Finetuning is a required step

Variants. RoBERTa, DistilBERT, ALBERT

BERT architecture

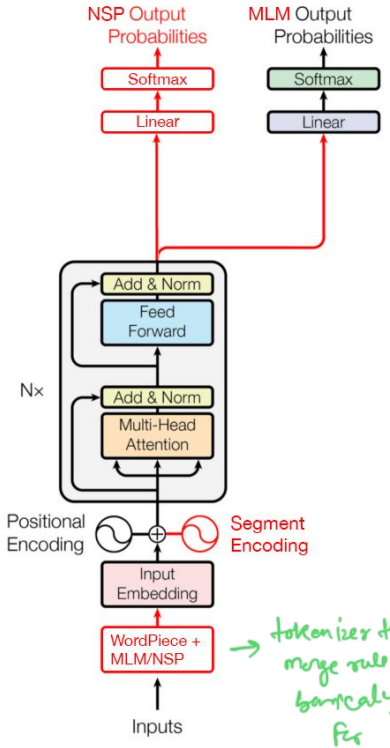


Original transformer (2017)

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

BERT architecture

Original transformer (2017)

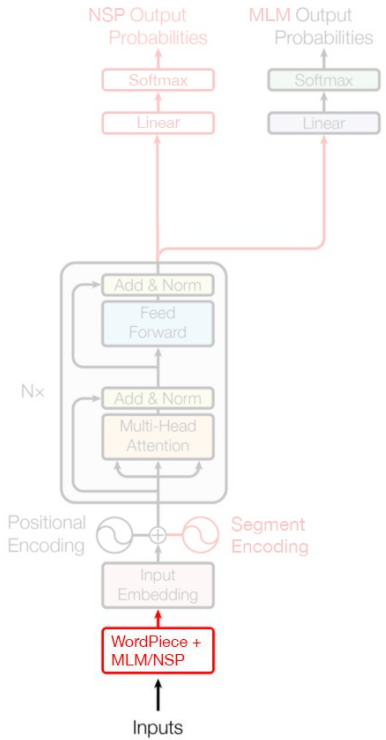


BERT (2018)

tokenizer that learns on training set based on merge rule that maximize the likelihood; basically merging most/multiple possible tokens for improving training performance

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Input processing



WordPiece algorithm

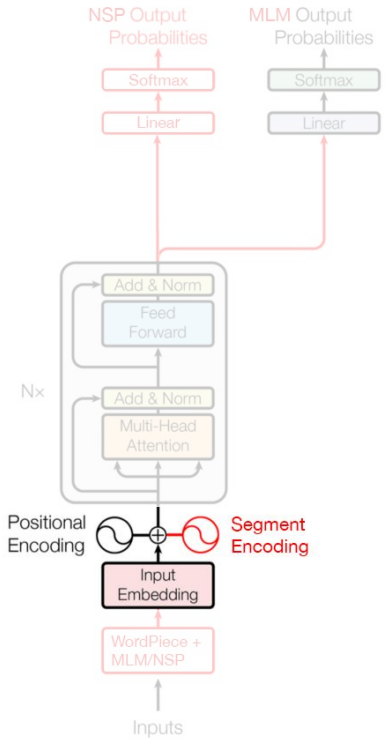
- Tokenizer trained on a training set beforehand
- Vocabulary size: ~30,000
- Great at detecting common particles

NSP / MLM task processing

- Add [CLS] token at the beginning of the input
- Separate consecutive segments with the [SEP] token and put another one at the end
- Use [MASK] to mask inputs

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Input embedding



Input embeddings

- Gigantic lookup table
- Learns an embedding for each word of the vocabulary

Positional encoding

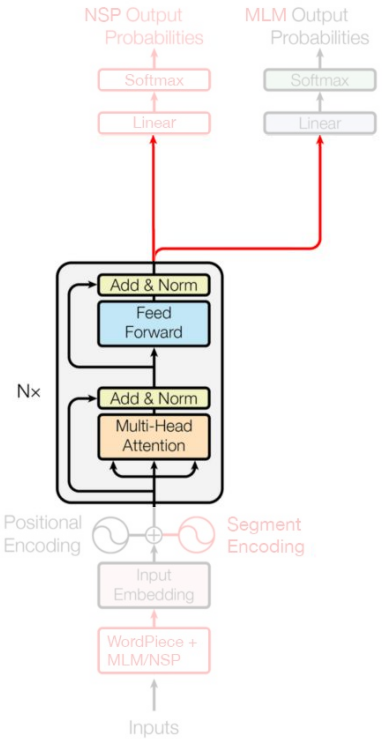
- Helps the network associate tokens with a position
- Encoding either learnt or fixed with cosines and sines

(new!) Segment encoding

- Shared embedding for a segment

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Encoder-only model



Model

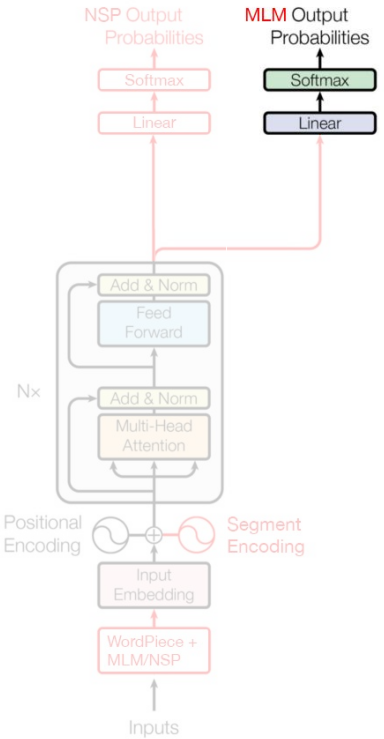
Encoder part of the original transformers paper

Goal

- Represent input data with features (hopefully) needed for NLP tasks
- Leverage the Transformer's self-attention mechanism
- Use learned embedding towards classification-oriented tasks

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Proxy task: Masked Language Modeling



Idea. 15% of input tokens are set up for prediction where:

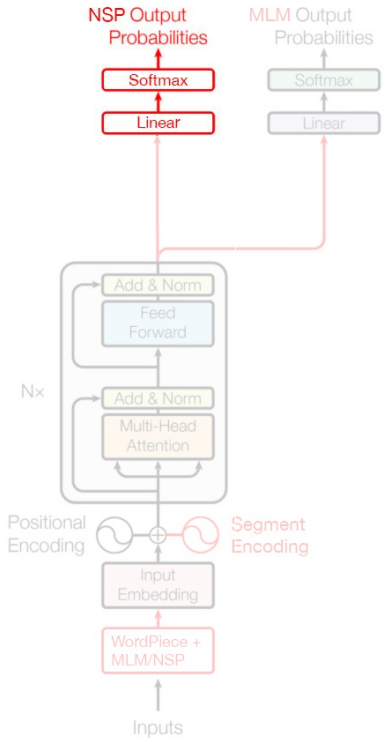
- 80% are masked
- 10% are changed to a random word
- 10% are unchanged

Benefits

- Network learns language modeling based on contextual information
- Regularization reflects probabilistic nature of language

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Proxy task: Next Sentence Prediction



Idea. Pick two sentences from the corpus, where:

- 50% of the time, they follow each other
- 50% of the time, they **do not** follow each other

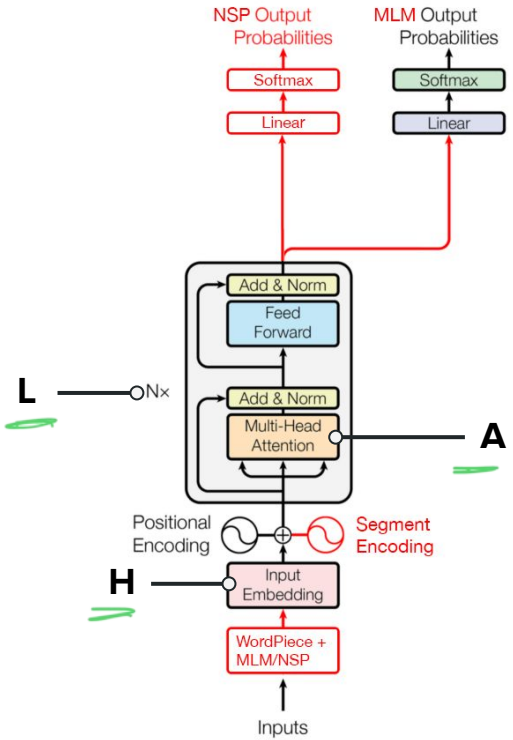
Task. Predict if they actually follow each other.

Benefits

- Network implicitly learns to detect useful contextual information
- Easy classification task that does not require any labels

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Hyperparameters



Model.

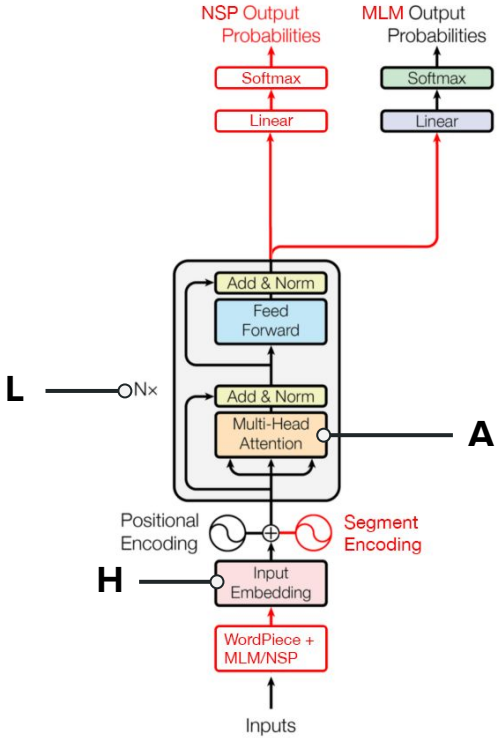
- **L** Layers.
- **H** Hidden layer size aka dimension of embeddings.
- **A** Attention heads operating in parallel.

Data.

- Language-specific / multilingual. Languages on which the model has been trained on.
- Cased / uncased. Whether inputs are converted to lowercase or not.

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Some numbers



	L	H	A	Parameters
BERT-Tiny	2	128	2	4M
BERT-Mini	4	256	4	11M
BERT-Small	4	512	8	30M
BERT-Medium	8	512	8	42M
BERT-Base	12	768	12	110M
BERT-Large	24	1024	16	340M

Left figure adapted from "Attention Is All You Need", Vaswani et al., 2017. Parameters in the right table were computed in this paper.

Overview of BERT finetuning

Goal. Leverage embeddings learned by BERT for a “sister” task

Overview of BERT finetuning

Goal. Leverage embeddings learned by BERT for a “sister” task

Tricks

- Use weights from **already massively pretrained** model
- **Freezing** early layers: sometimes better trade-off complexity/performance
- Great results possible with **minimal labeled data** (depending on complexity/proximity to pretrained data distribution + objectives)

Overview of BERT finetuning

Goal. Leverage embeddings learned by BERT for a "sister" task

Tricks

- Use weights from **already massively pretrained** model
- **Freezing** early layers: sometimes better trade-off complexity/performance
- Great results possible with **minimal labeled data** (depending on complexity/proximity to pretrained data distribution + objectives)

Use cases

- Sequence classification: e.g. sentiment extraction
- Token classification: e.g. question answering

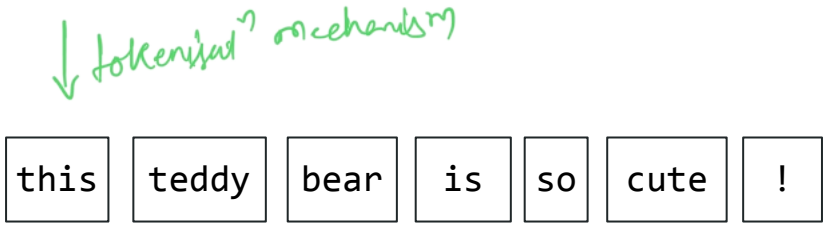
Finetuning: sentiment extraction

```
This teddy bear is SO CUTE!
```

Finetuning: sentiment extraction

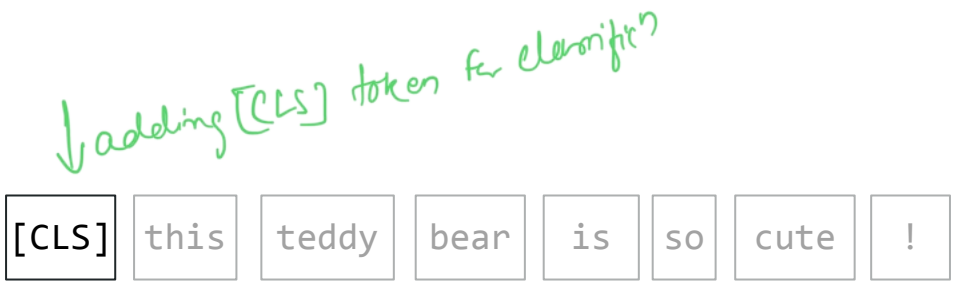
this teddy bear is so cute!

Finetuning: sentiment extraction

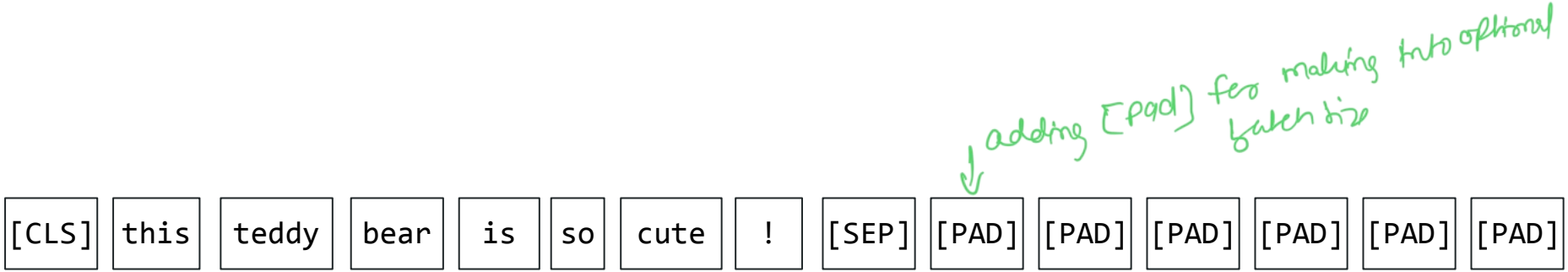


Finetuning: sentiment extraction

Add [CLS] token as a placeholder for sentiment.



Finetuning: sentiment extraction



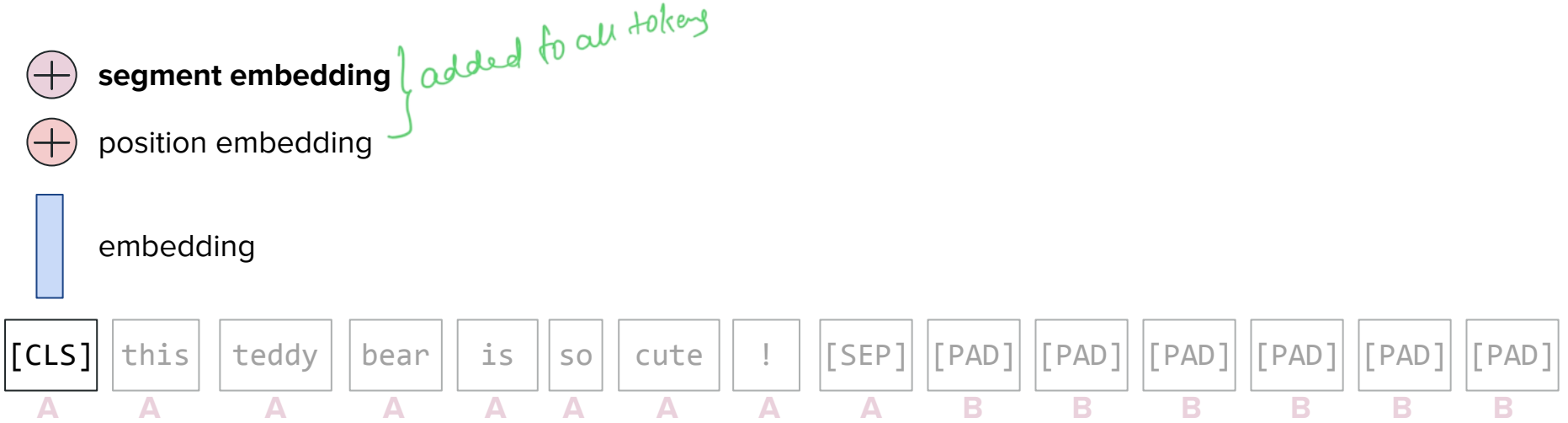
Finetuning: sentiment extraction

embedding

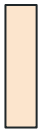
[CLS]	this	teddy	bear	is	so	cute	!	[SEP]	[PAD]	[PAD]	[PAD]	[PAD]	[PAD]	[PAD]

Finetuning: sentiment extraction

Finetuning: sentiment extraction



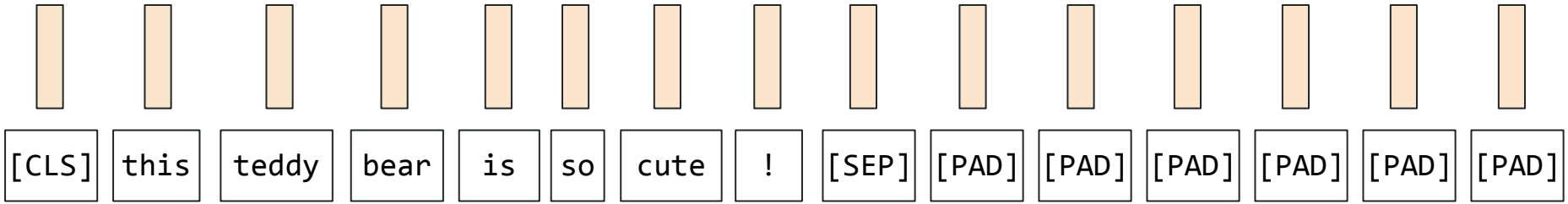
Finetuning: sentiment extraction



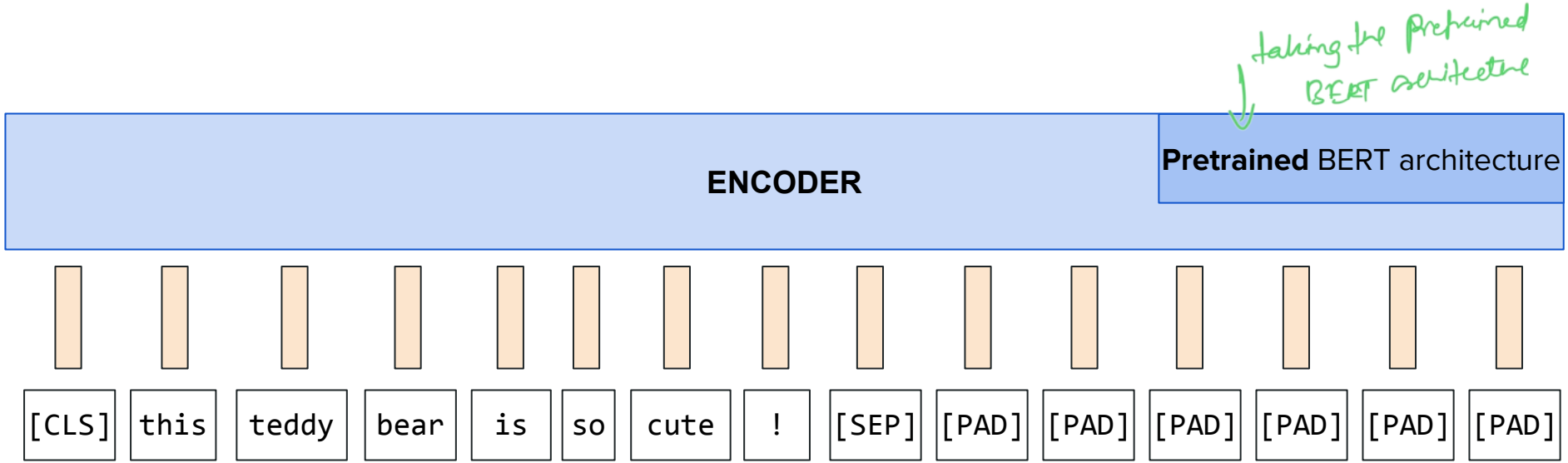
position- and segment-aware embedding

[CLS]	this	teddy	bear	is	so	cute	!	[SEP]	[PAD]	[PAD]	[PAD]	[PAD]	[PAD]	[PAD]

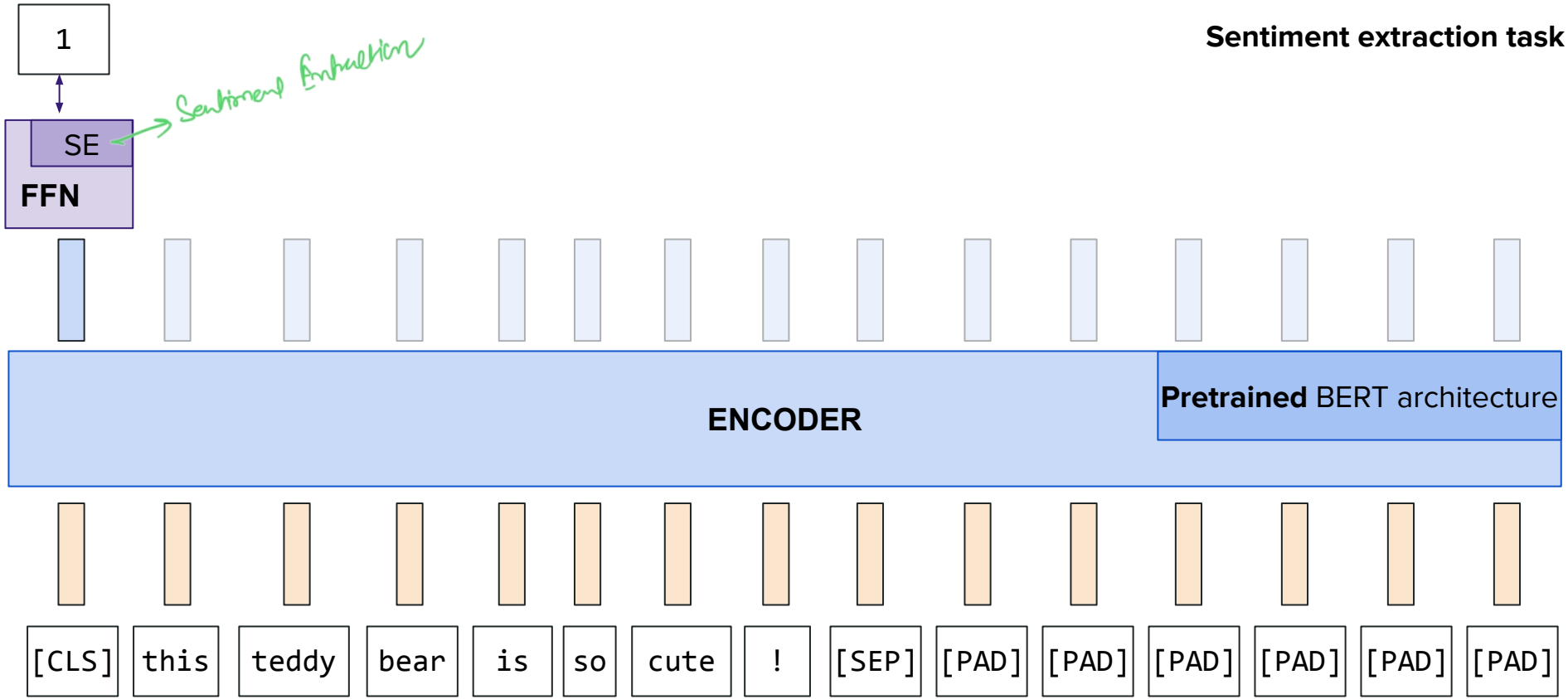
Finetuning: sentiment extraction



Finetuning: sentiment extraction



Finetuning: sentiment extraction



Takeaways and shortcomings

Benefits.

- State of the art results
- ~True contextual representation of words
- Adaptable to many classification tasks

Takeaways and shortcomings

Benefits.

- State of the art results
- ~True contextual representation of words
- Adaptable to many classification tasks

Applications. Widely used in the industry for anything related to encoding

Takeaways and shortcomings

Benefits.

- State of the art results
- ~True contextual representation of words
- Adaptable to many classification tasks

Applications. Widely used in the industry for anything related to encoding

Limitations.

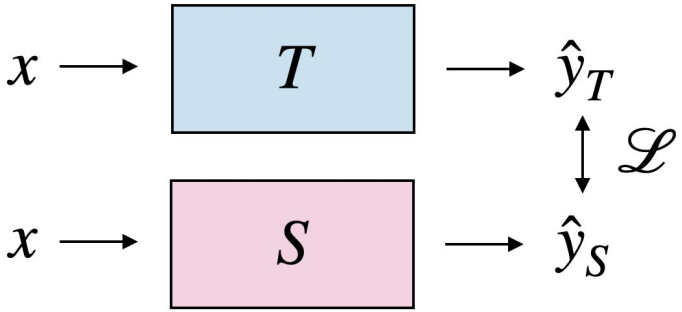
- Context window size is limited
- Computationally expensive: hard sell for low-latency/cost-sensitive applications
- Training paradigm is complex: MLM/NSP + finetuning

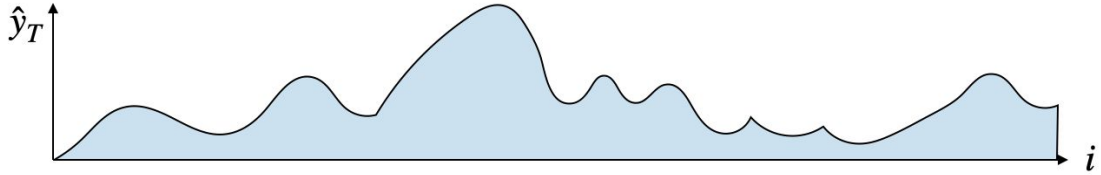
Trick: distillation

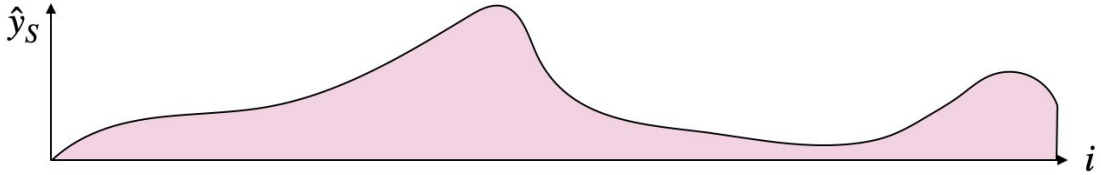
"The **soft targets** contain almost all the **knowledge**."

"Dark knowledge", Hinton et al., 2014.

Trick: distillation

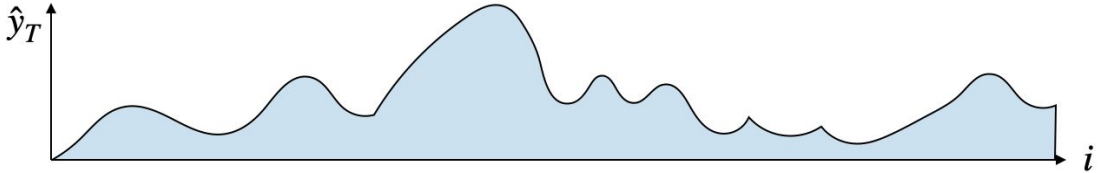




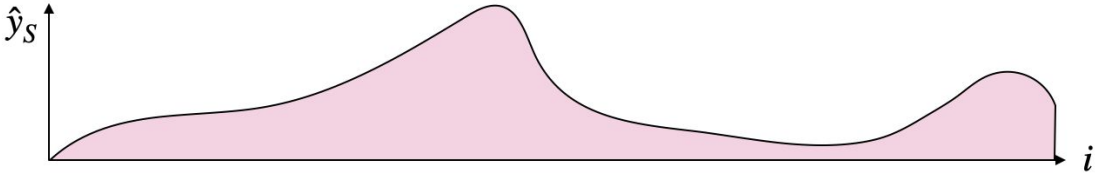


Figures from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Trick: distillation

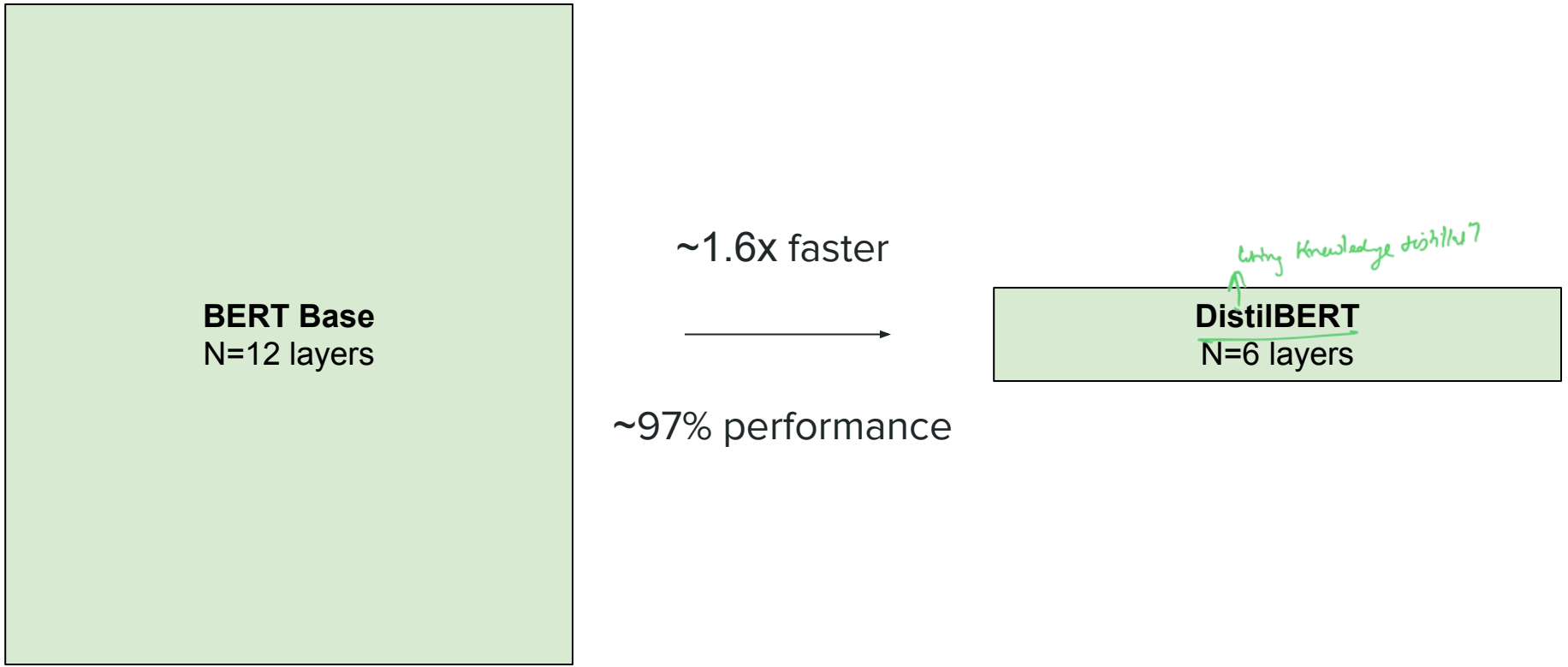


$$\mathrm{KL}(\widehat{y}_T||\widehat{y}_S) = \sum_i \widehat{y}_T^{(i)} \log\left(\frac{\widehat{y}_T^{(i)}}{\widehat{y}_S^{(i)}}\right)$$



Figures from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Variant for efficiency: DistilBERT



"DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter", Sanh et al., 2019.

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Modeling

- Removing NSP/segment encodings: ~no effect!*
- Static → dynamic masking across epochs // dynamic masking dynamically

*DistilBERT had also removed it

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Modeling

- Removing NSP/segment encodings: ~no effect!
- Static → dynamic masking across epochs

Data

- Richer: 16 GB → 160 GB of pretraining corpus size
- For longer: 1M steps of batch size 256 vs. 500k of batch size 8k

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Modeling

- Removing NSP/segment encodings: ~no effect!
- Static → dynamic masking across epochs

Data

- Richer: 16 GB → 160 GB of pretraining corpus size
- For longer: 1M steps of batch size 256 vs. 500k of batch size 8k

Result. +4% across benchmarks with same architecture

Thank you for your attention!