Language Model Pretraining

Yu Meng
University of Virginia
yumeng5@virginia.edu

Oct 09, 2024

Reminder

Join at
slido.com
#6434 404

- Assignment 3 due this Friday (10/11 11:59pm)!

UNIVERSITY of VIRGINIA

Join at
slido.com
#6434 404

Overview of Course Contents

- Week 1: Logistics & Overview
- Week 2: N-gram Language Models
- Week 3: Word Senses, Semantics & Classic Word Representations
- Week 4: Word Embeddings
- Week 5: Sequence Modeling and Neural Language Models
- Week 6-7: Language Modeling with Transformers (Pretraining + Fine-tuning)
- Week 8: Large Language Models (LLMs) & In-context Learning
- Week 9-10: Knowledge in LLMs and Retrieval-Augmented Generation (RAG)
- Week 11: LLM Alignment
- Week 12: Language Agents
- Week 13: Recap + Future of NLP
- Week 15 (after Thanksgiving): Project Presentations

(Recap) Position Encoding

- Motivation: inject positional information to input vectors

$$\boldsymbol{q}_i = \boldsymbol{x}_i \boldsymbol{W}^Q \quad \boldsymbol{k}_i = \boldsymbol{x}_i \boldsymbol{W}^K \quad \boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{W}^V \in \mathbb{R}^d$$

$$\boldsymbol{a}_i = \mathrm{Softmax}\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}}\right) \cdot \boldsymbol{v}_j$$

When $\boldsymbol{x}$ is word embedding, $\boldsymbol{q}$ and $\boldsymbol{k}$ do not have positional information!

- How to know the word positions in the sequence? Use position encoding!

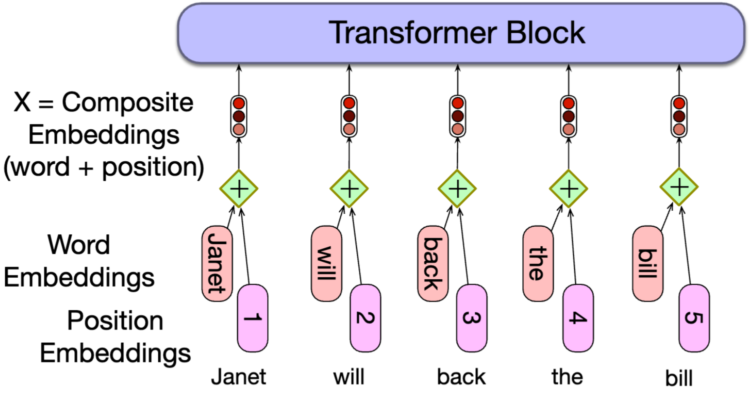


Figure source: https://web.stanford.edu/~jurafsky/slp3/9.pdf

(Recap) Subword Tokenization

- Strike a balance between character-level and word-level tokenization
  - Capture meaningful subword semantics
  - Handle out-of-vocabulary words better
  - Efficient sequence modeling
- Three common algorithms:
  - Byte-Pair Encoding (BPE): Sennrich et al. (2016)
  - WordPiece: Schuster and Nakajima (2012)
  - SentencePiece: Kudo and Richardson (2018)
- Subword tokenization usually consists of two parts:
  - A token learner that takes a raw training corpus and induces a **vocabulary** (a set of tokens)
  - A token segmenter that takes a raw sentence and **tokenizes** it according to that vocabulary

(Recap) BPE: Token Learner

Token learner of BPE

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

Figure source: https://web.stanford.edu/~jurafsky/slp3/2.pdf

(Recap) BPE: Token Segmenter

- Once we learn our vocabulary, we need a token segmenter to tokenize an unseen sentence (from test set)
- Just run (greedily based on training data frequency) on the merge rules we have learned from the training data on the test data
- Example:
  - Assume the merge rules: [(e, r), (er, _), (n, e), (ne, w), (l, o), (lo, w), (new, er_), (low, _)]
  - First merge all adjacent "er", then all adjacent "er_", then all adjacent "ne"…
  - "newer_" from the test set will be tokenized as a whole word
  - "lower_" from the test set will be tokenized as "low" + "er_"

low low low low low lowest lowest newer newer newer
newer newer newer wider wider wider new new

"lower_" is an unseen word from the training set

Agenda

- Other Transformer Modules
- Language Model Pretraining: Overview
- Pretraining for Different Transformer Architectures

Transformer Block

- Modules in Transformer layers:
  - Multi-head attention
  - Layer normalization (LayerNorm)
  - Feedforward network (FFN)
  - Residual connection

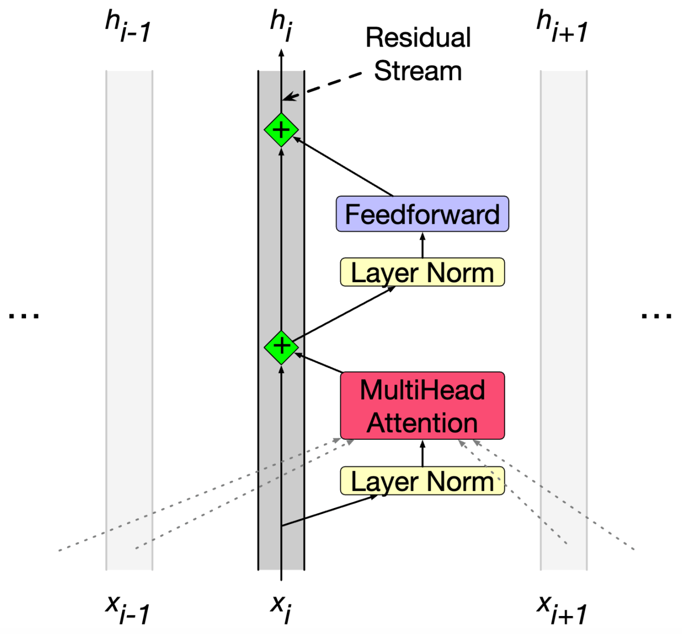


Figure source: https://web.stanford.edu/~jurafsky/slp3/9.pdf

Layer Normalization: Motivation

- Proposed in [Ba et al. (2016)]
- **"Internal covariate shift"**
  - The distribution of inputs to DNN can change during training
  - Slow down the training process: the model constantly adapts to changing distributions

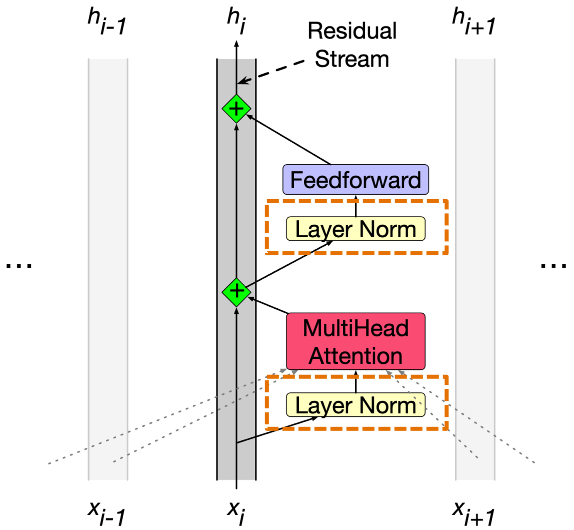


Figure source: https://web.stanford.edu/~jurafsky/slp3/9.pdf

Layer Normalization: Solution

- Normalize the input vector $\boldsymbol{x}$
  - Calculate the mean & standard deviation over the input vector dimensions

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

  - Apply normalization

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x} - \mu}{\sigma}$$

- Learn to scale and shift the normalized output with parameters

$$\mathrm{LayerNorm}(\boldsymbol{x}) = \gamma \frac{\boldsymbol{x} - \mu}{\sigma} + \beta$$

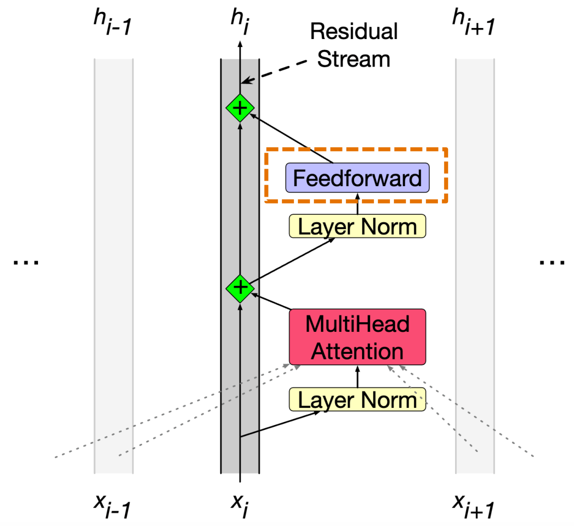
Learnable parameters

Feedforward Network (FFN)

- FFN in Transformer is a 2-layer network (one hidden layer, two weight matrices)

$$\mathrm{FFN}(\boldsymbol{x}_i) = \mathrm{ReLU}(\boldsymbol{x}_i \boldsymbol{W}_1)\boldsymbol{W}_2$$

- Apply non-linear activation after the first layer
- Same weights applied to every token
- Weights are different across different Transformer layers



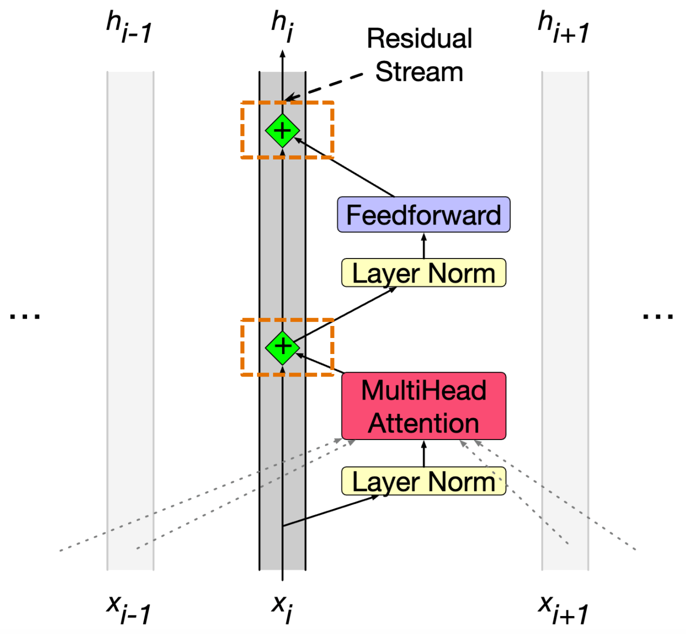
FFNs can help language models store factual knowledge!
(more on this in later lectures)

Residual Connections

- Add the original input to the output of a sublayer (e.g., attention/FFN)

$$\boldsymbol{y} = \boldsymbol{x} + f(\boldsymbol{x})$$

- Benefits
  - Address the vanishing gradient problem
  - Facilitate information flow across the network
  - Help scale up model



Language Model Head

- Language model head is added to the final layer
- Usually apply the weight tying trick (share weights between input embeddings and the output embeddings

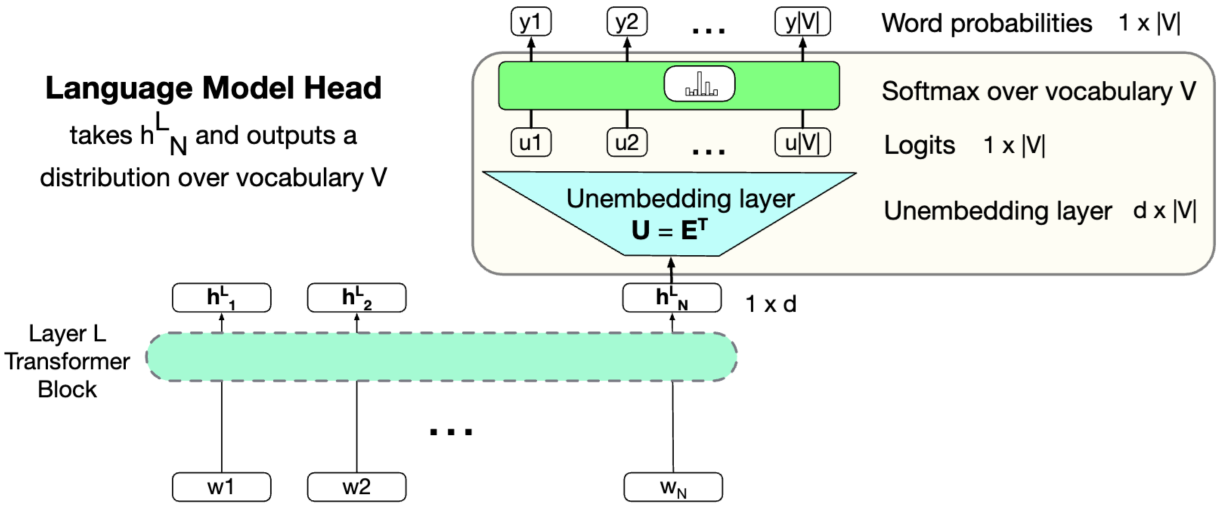


Figure source: https://web.stanford.edu/~jurafsky/slp3/9.pdf

Join at
slido.com
#6434 404

Transformer Language Model: Overview

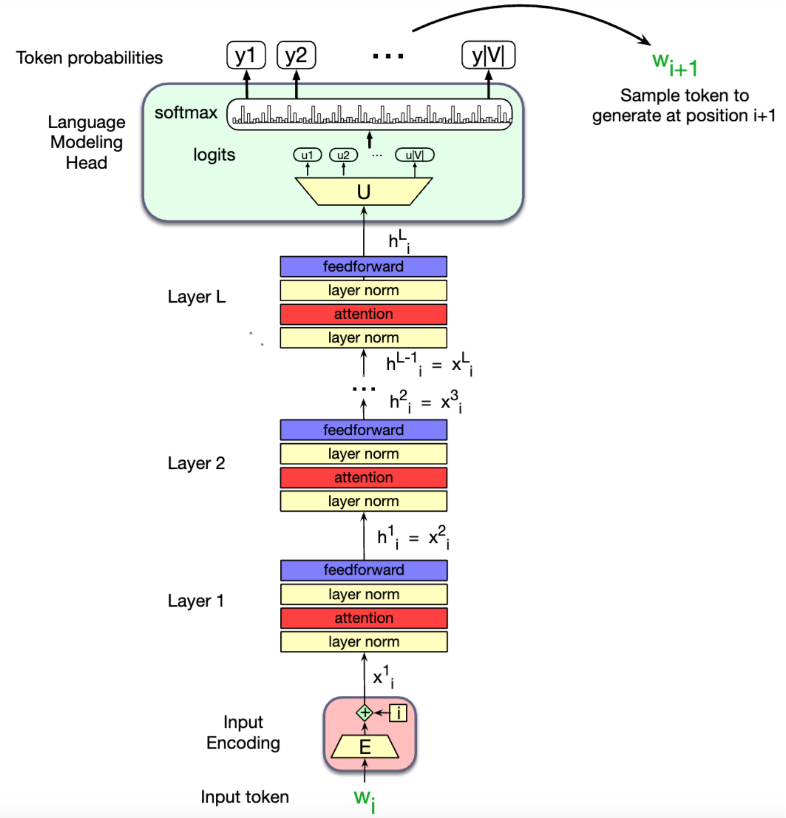


Figure source: https://web.stanford.edu/~jurafsky/slp3/9.pdf

Transformer Language Model Training

Use cross-entropy loss to train Transformers for language modeling (like RNN LMs)

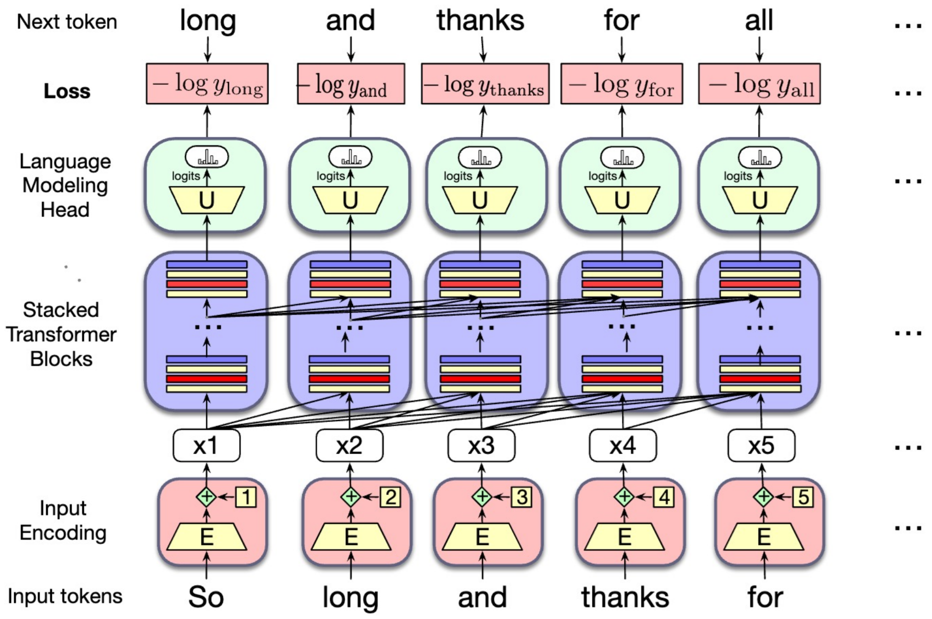


Figure source: https://web.stanford.edu/~jurafsky/slp3/10.pdf

Agenda

- Other Transformer Modules
- Language Model Pretraining: Overview
- Pretraining for Different Transformer Architectures

Pretraining: Motivation

- Before pretraining became prevalent in NLP, most NLP models were trained from scratch on downstream task data
- **Data scarcity**: many NLP tasks do not have large labeled datasets available (costly to obtain)
- **Poor generalization**: models trained from scratch on specific tasks do not generalize well to unseen data or other tasks
- **Sensitivity to noise and randomness**: models are more likely to learn spurious correlations or be affected by annotation errors/randomness in training

Join at
slido.com
#6434 404

Pretraining: Motivation

- There are abundant text data on the web, with rich information of linguistic features and knowledge about the world
- Learning from these easy-to-obtain data greatly benefits various downstream tasks

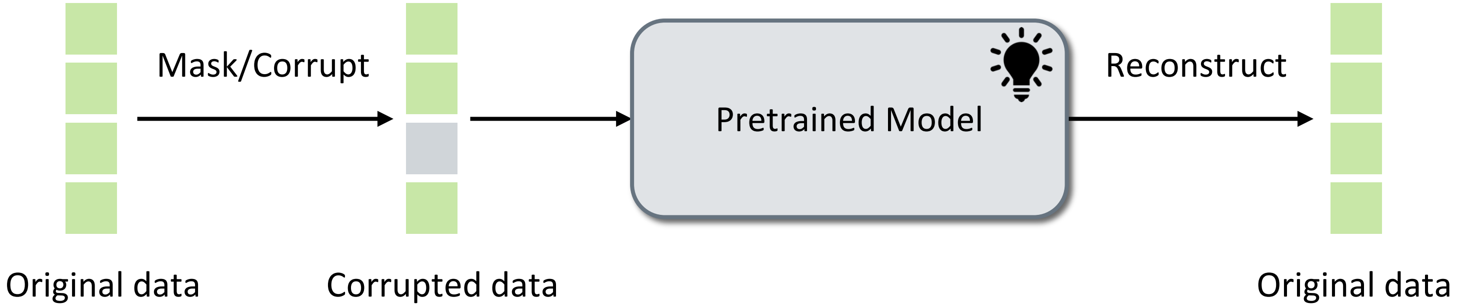
Pretraining: Multi-Task Learning

- In my free time, I like to **{run, banana}** (*Grammar*)
- I went to the zoo to see giraffes, lions, and **{zebras, spoon}** (*Lexical semantics*)
- The capital of Denmark is **{Copenhagen, London}** (*World knowledge*)
- I was engaged and on the edge of my seat the whole time. The movie was **{good, bad}** (*Sentiment analysis*)
- The word for "pretty" in Spanish is **{bonita, hola}** (*Translation*)
- 3 + 8 + 4 = **{15, 11}** (*Math*)
- …

Examples from: https://docs.google.com/presentation/d/1hQUd3pF8_2Gr2Obc89LKjmHL0DlH-uof9M0yFVd3FA4/edit#slide=id.g28e2e9aa709_0_1

Pretraining: Self-Supervised Learning

- Pretraining is a form of **self-supervised** learning
- Make a part of the input unknown to the model
- Use other parts of the input to reconstruct/predict the unknown part



No Human Supervision Needed!

Pretraining + Fine-Tuning

- Pretraining: trained with pretext tasks on large-scale text corpora
- Fine-tuning (continue training): adjust the pretrained model's parameters with fine-tuning data
- Fine-tuning data can have different forms:
  - Task-specific labeled data (e.g., sentiment classification, named entity recognition)
  - (Multi-turn) dialogue data (i.e., instruction tuning)

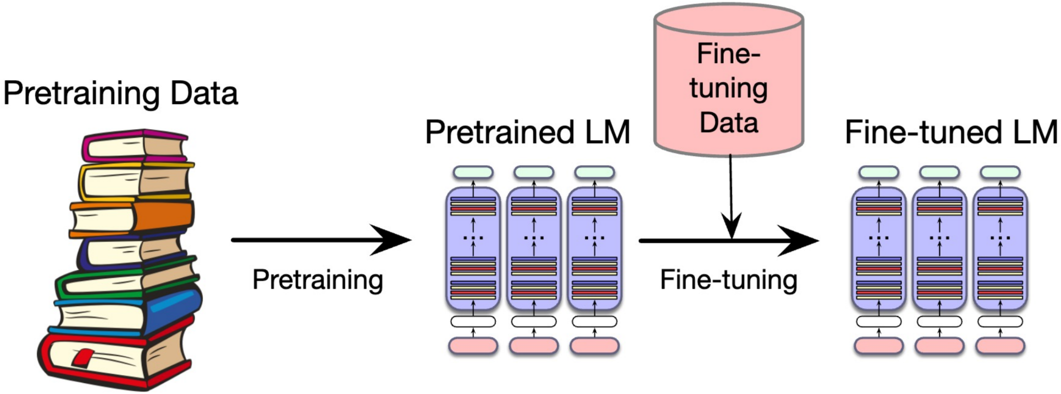


Figure source: https://web.stanford.edu/~jurafsky/slp3/10.pdf

Transformer for Pretraining

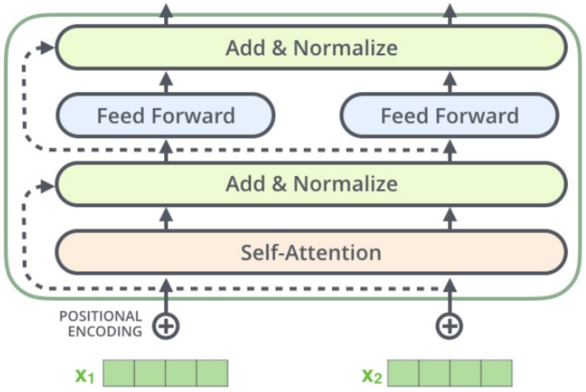
- Transformer is the common backbone architecture for language model pretraining
- **Efficiency**: Transformer processes all tokens in a sequence simultaneously – fast and efficient to train, especially on large datasets
- **Scalability**: Transformer architectures have shown impressive scaling properties, with performance improving as model size and training data increase (more on this later!)
- **Versatility**: Transformer can be adapted for various tasks and modalities beyond just text, including vision, audio, and other multimodal applications

Agenda

- Other Transformer Modules
- Language Model Pretraining: Overview
- Pretraining for Different Transformer Architectures

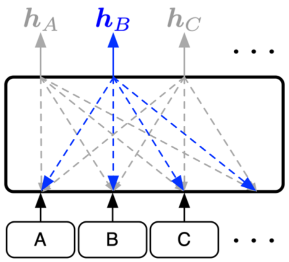
Transformer Architectures

- Based on the type of self-attention, Transformer can be instantiated as
  - Encoder: Bidirectional self-attention
  - Decoder: Unidirectional self-attention
  - Encoder-decoder: Use both encoder and decoder



Encoder

Decoder



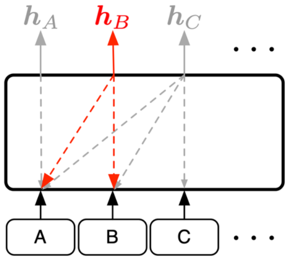
every token attends to all tokens

Bidirectional Self-Attention

N

q1·k1	q1·k2	q1·k3	q1·k4
q2·k1	q2·k2	q2·k3	q2·k4
q3·k1	q3·k2	q3·k3	q3·k4
q4·k1	q4·k2	q4·k3	q4·k4

N



every token attends to its previous tokens

Unidirectional Self-Attention

N

q1·k1	$-\infty$	$-\infty$	$-\infty$
q2·k1	q2·k2	$-\infty$	$-\infty$
q3·k1	q3·k2	q3·k3	$-\infty$
q4·k1	q4·k2	q4·k3	q4·k4

N

Applications of Transformer Architectures

- Encoder (e.g., BERT):
  - Capture bidirectional context to learn each token representations
  - Suitable for natural language understanding (NLU) tasks
- Decoder (modern large language models, e.g., GPT):
  - Use prior context to predict the next token (conventional language modeling)
  - Suitable for natural language generation (NLG) tasks
  - Can also be used for NLU tasks by generating the class labels as tokens
- Encoder-decoder (e.g., BART, T5):
  - Use the encoder to process input, and use the decoder to generate outputs
  - Can conduct all tasks that encoders/decoders can do

NLU:
Text classification
Named entity recognition
Relation extraction
Sentiment analysis
…

NLG:
Text summarization
Machine translation
Dialogue system
Question answering
…

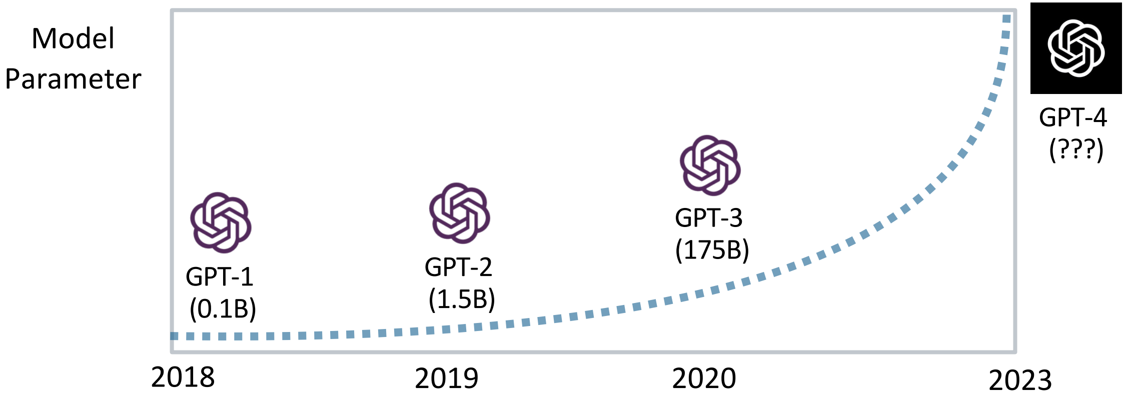
Decoder Pretraining

- Decoder architecture is the prominent choice in large language models
- Pretraining decoders is first introduced in GPT (generative pretraining) models
- Follow the standard language modeling (cross-entropy) objective

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\log p_{\boldsymbol{\theta}}(x_i|x_1, x_2, \ldots, x_{i-1})$$

GPT Series

- GPT-1 (2018): 12 layers, 117M parameters, trained in ~1 week
- GPT-2 (2019): 48 layers, 1.5B parameters, trained in ~1 month
- GPT-3 (2020): 96 layers, 175B parameters, trained in several months



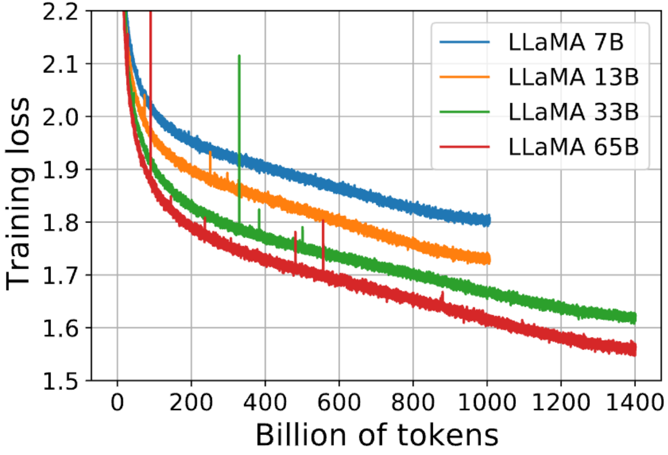
Papers: (GPT-1) https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
(GPT-2) https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
(GPT-3) https://arxiv.org/pdf/2005.14165.pdf

UNIVERSITY of VIRGINIA

Join at
slido.com
#6434 404

Llama Series

- Llama-1 (2023/02): 7B/13B/33B/65B
- Llama-2 (2023/07): 7B/13B/70B
- Llama-3 (3.1 & 3.2) (2024/07): 1B/3B/8B/70B/405B w/ multi-modality



Larger models learn pretraining data better

Papers: (Llama-1) https://arxiv.org/pdf/2302.13971
(Llama-2) https://arxiv.org/pdf/2307.09288
(Llama-3) https://arxiv.org/pdf/2407.21783

UNIVERSITY of VIRGINIA

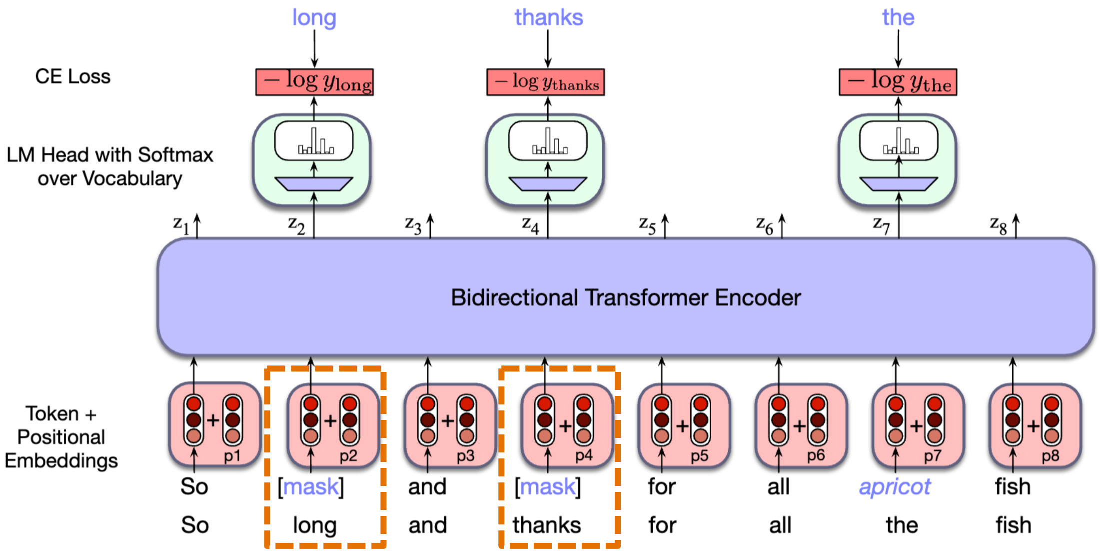
Join at
slido.com
#6434 404

Further Reading on Decoder LMs

- Mistral 7B [Jiang et al., 2023]
- Qwen Technical Report [Bai et al., 2023]
- GPT-4 Technical Report [OpenAI, 2023]
- Gemma: Open Models Based on Gemini Research and Technology [Gemma Team, 2024]

Encoder Pretraining: BERT

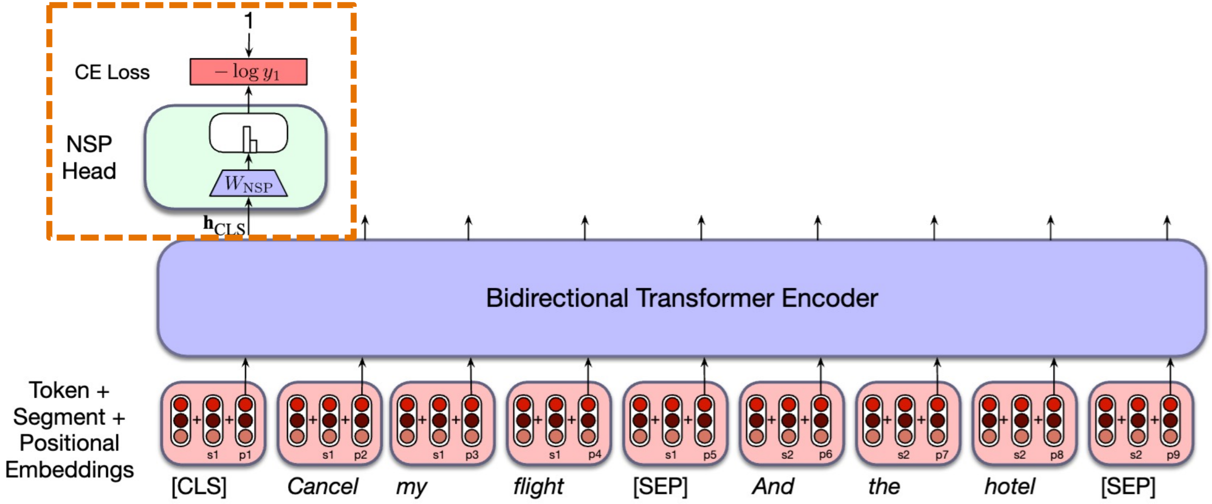
- BERT pretrains encoder models with bidirectionality
- **Masked language modeling** (MLM): With 15% words randomly masked, the model learns bidirectional contextual information to predict the masked words



BERT: https://arxiv.org/pdf/1810.04805.pdf Figure source: https://web.stanford.edu/~jurafsky/slp3/11.pdf

Encoder Pretraining: BERT

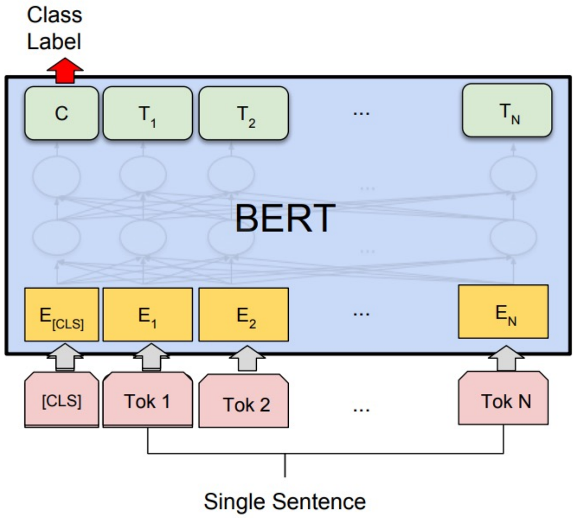
- **Next sentence prediction** (NSP): the model is presented with pairs of sentences
- The model is trained to predict whether each pair consists of an actual pair of adjacent sentences from the training corpus or a pair of unrelated sentence



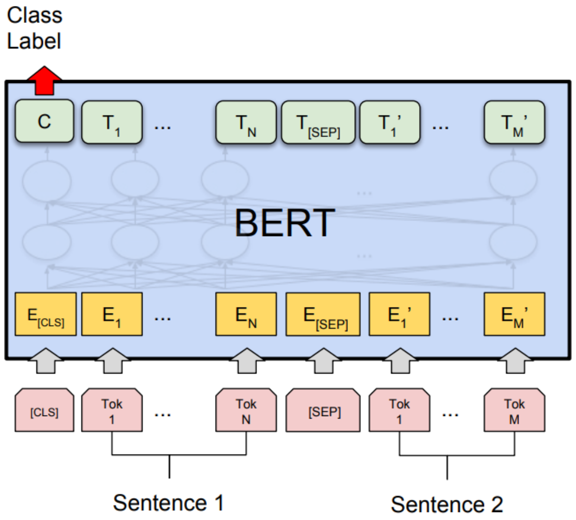
BERT: https://arxiv.org/pdf/1810.04805.pdf Figure source: https://web.stanford.edu/~jurafsky/slp3/11.pdf

BERT Fine-Tuning

- Fine-tuning pretrained BERT models takes different forms depending on task types
- Usually replace the LM head with a linear layer fine-tuned on task-specific data



Single sequence classification



Sequence-pair classification

BERT vs. GPT on NLU tasks

- BERT outperforms GPT-1 on a set of NLU tasks
- Encoders capture **bidirectional** contexts – build a richer understanding of the text by looking at both preceding and following words
- Are encoder models still better than state-of-the-art (large) decoder models?
  - LLMs can be as good as (if not better than) encoders model on NLU: [Can ChatGPT Understand Too?]
  - The sheer model size + massive amount of pretraining data compensate for LLMs' unidirectional processing

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

BERT Variant I: RoBERTa

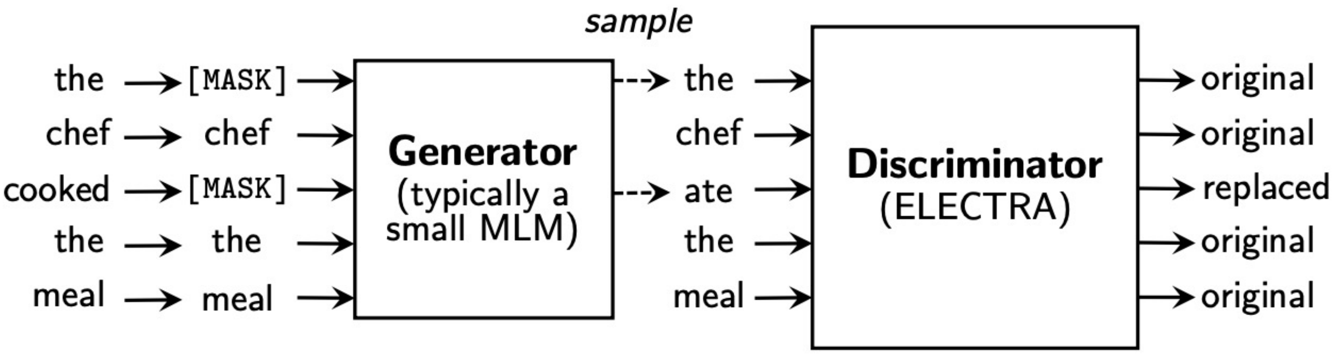
- Pretrain the model for longer, with bigger batches over more data
- Pretrain on longer sequences
- Dynamically change the masking patterns applied to the training data in each epoch

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

RoBERTa: https://arxiv.org/pdf/1907.11692.pdf

BERT Variant II: ELECTRA

- Use a small MLM model as an auxiliary generator (discarded after pretraining)
- Pretrain the main model as a discriminator
- The small auxiliary MLM and the main discriminator are jointly trained
- The main model's pretraining task becomes more and more challenging in pretraining
- Major benefits: sample efficiency + learning curriculum



ELECTRA: https://arxiv.org/pdf/2003.10555.pdf

ELECTRA Performance

- ELECTRA pretraining incurs lower computation costs compared to MLM
- Better downstream task performance

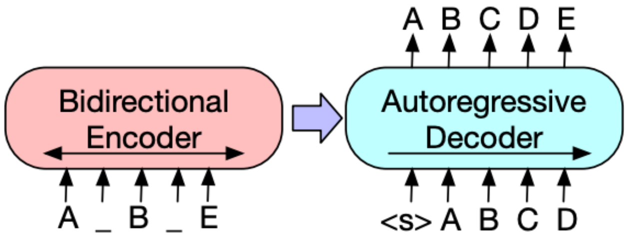
Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

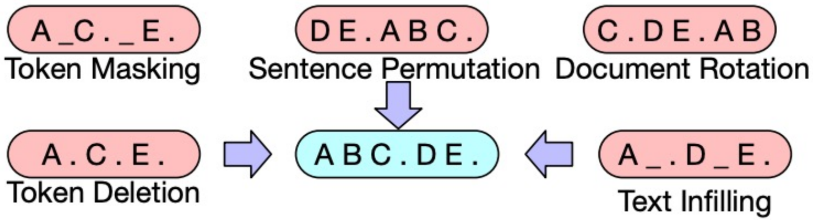
Further Reading on Encoder LMs

- XLNet: Generalized Autoregressive Pretraining for Language Understanding [Yang et al., 2019]
- ALBERT: A Lite BERT for Self-supervised Learning of Language Representations [Lan et al., 2020]
- DeBERTa: Decoding-enhanced BERT with Disentangled Attention [He et al., 2020]
- COCO-LM: Correcting and Contrasting Text Sequences for Language Model Pretraining [Meng et al. 2021]

Encoder-Decoder Pretraining: BART

- Pretraining: Apply a series of noising schemes (e.g., masks, deletions, permutations…) to input sequences and train the model to recover the original sequences
- Fine-Tuning:
  - For NLU tasks: Feed the same input into the encoder and decoder, and use the final decoder token for classification
  - For NLG tasks: The encoder takes the input sequence, and the decoder generates outputs autoregressively





BART: https://arxiv.org/pdf/1910.13461.pdf

UNIVERSITY of VIRGINIA

Join at
slido.com
#6434 404

BART Performance

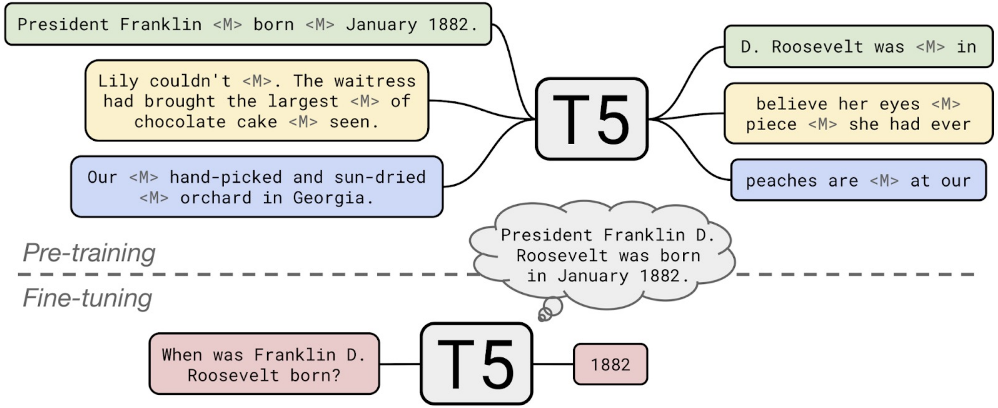
- Comparable to encoders on NLU tasks
- Good performance on NLG tasks

	SQuAD 1.1 EM/F1	SQuAD 2.0 EM/F1	MNLI m/mm	SST Acc	QQP Acc	QNLI Acc	STS-B Acc	RTE Acc	MRPC Acc	CoLA Mcc
BERT	84.1/90.9	79.0/81.8	86.6/-	93.2	91.3	92.3	90.0	70.4	88.0	60.6
UniLM	-/-	80.5/83.4	87.0/85.9	94.5	-	92.7	-	70.9	-	61.1
XLNet	**89.0**/94.5	86.1/88.8	89.8/-	95.6	91.8	93.9	91.8	83.8	89.2	63.6
RoBERTa	88.9/**94.6**	**86.5/89.4**	**90.2/90.2**	96.4	92.2	94.7	**92.4**	86.6	**90.9**	**68.0**
BART	88.8/**94.6**	86.1/89.2	89.9/90.1	**96.6**	**92.5**	**94.9**	91.2	**87.0**	90.4	62.8

	CNN/DailyMail			XSum		
	R1	R2	RL	R1	R2	RL
Lead-3	40.42	17.62	36.67	16.30	1.60	11.95
PTGEN (See et al., 2017)	36.44	15.66	33.42	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38	28.10	8.02	21.72
UniLM	43.33	20.21	40.51	-	-	-
BERTSUMABS (Liu & Lapata, 2019)	41.72	19.39	38.76	38.76	16.33	31.15
BERTSUMEXTABS (Liu & Lapata, 2019)	42.13	19.60	39.18	38.81	16.50	31.27
BART	**44.16**	**21.28**	**40.90**	**45.14**	**22.27**	**37.25**

Encoder-Decoder Pretraining: T5

- T5: **T**ext-**to**-**T**ext **T**ransfer **T**ransformer
- Pretraining: Mask out spans of texts; generate the original spans
- Fine-Tuning: Convert every task into a sequence-to-sequence generation problem
- We'll see this model again in the instruction tuning lectures



T5: https://arxiv.org/pdf/1910.10683

T5 Performance

- Good performance across various tasks
- T5 vs. BART performance: unclear comparison due to difference in model sizes & training setups

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	**75.1**	90.6	**92.2**	**91.9**	96.9	**92.8**	**94.5**

Encoder-Decoder vs. Decoder-Only

- Modern LLMs are mostly based on the decoder-only Transformer architecture
- Simplicity:
  - Decoder-only models are simpler in structure (one Transformer model)
  - Encoder-decoder models require two Transformer models
- Efficiency:
  - Decoder-only models are more parameter-efficient for text generation
  - Encoder-decoder models' encoder part does not contribute to generation
- Scalability:
  - Decoder-only models scale very well with increased model size and data
  - Encoder-decoder models do not outperform decoder-only models at large model sizes

Thank You!

Yu Meng
University of Virginia
yumeng5@virginia.edu