CS 6501 Natural Language Processing (Spring 2024)

Yu Meng
University of Virginia
yumeng5@virginia.edu

Jan 22, 2024

Course Format & Grading (Recap & Updates)

- Course Website: https://yumeng5.github.io/teaching/2024-spring-cs6501
- **Paper Presentation (30%)**
  - Starting from the next lecture, each lecture will be presented by a group of 2 or 3 students
  - Signup sheet released: https://docs.google.com/spreadsheets/d/1-QqSvqdLg6ejfeS8jscHHaFcEUXDBK7sgBKXnM7U5vU/edit?usp=drive_link
  - **Presentation duration**: strictly limited to 60 minutes, followed by a 10-minute question-and-answer session with the audience
  - **Deadline**: Email your slides to the instructor and TAs at least 48 hours before your presentation (e.g., if presenting on Monday, slides should be emailed by Saturday 3:30 pm)
  - Assessment: Clarity, Completeness, Teamwork, Question answering

UNIVERSITY of VIRGINIA

Course Format & Grading (Recap & Updates)

- Course Website: https://yumeng5.github.io/teaching/2024-spring-cs6501
- **Paper Presentation (30%)**
- Tips
  - No need to cover every detail of the papers; focus on conveying general ideas and insights
  - For theoretical papers, don’t go over each proof in detail, but explain the major conclusions/insights of the theories
  - For empirical papers, don’t present every piece of experiment results, but explain how the empirical findings support the major claims
- A good presentation should highlight
  - The major contributions of the paper
  - Why these contributions are deemed important (e.g., did they reveal any previously unknown facts or change people’s opinions on a widely acknowledged phenomenon?)
  - The most important technical details (e.g., the motivation & implementation of a new training objective/model architecture design)
  - The limitations of the work and how they might be addressed in the future

Course Format & Grading (Recap & Updates)

- Course Website: https://yumeng5.github.io/teaching/2024-spring-cs6501
- **Participation (20%):**
  - Starting from the next lecture, everyone is required to complete two mini-assignments
  - **Pre-lecture question**: read the 4 papers to be introduced in the lecture, and submit a question you have when you read them
  - **Post-lecture feedback**: provide feedback to the presenters after the lecture
  - We'll use Google Forms (released later today; announcement on Canvas) to collect pre-lecture questions and post-lecture feedback and share them with the presenters
  - **Deadlines**: pre-lecture questions are due one day before the lecture (e.g., For Monday lectures, you need to submit the question by Sunday 11:59 pm); post-lecture feedback is due each Friday (both Monday & Wednesday feedback is due Friday 11:59 pm)

UNIVERSITY of VIRGINIA

Course Format & Grading (Recap & Updates)

- Course Website: https://yumeng5.github.io/teaching/2024-spring-cs6501
- **Project (50%):**
  - Complete a research project, present your results, and submit a project report
  - Work in a team of 2 or 3 (any deviation from this size requires prior approval from the instructor) – may or may not be the same team as your presentation group
  - (Type 1) A comprehensive survey report: carefully examine and summarize existing literature on a topic covered in this course; provide detailed and insightful discussions on the unresolved issues, challenges, and potential future opportunities within the chosen topic
  - (Type 2) A hands-on project: not constrained to the course topics but must be centered around NLP; doesn't have to involve large language models (e.g., train or analyze smaller-scale language models for specific tasks); eligible for extra credits if publishable
  - **Project proposal**: 5% (deadline: 2/5)
  - **Mid-term report**: 10% (deadline: 3/13)
  - **Final presentation** (deadline: 4/24) **and final report** (deadline: 5/8): 35%

Agenda

- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

UNIVERSITY of VIRGINIA

Agenda

- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

How to Represent Texts?

- **Symbol-based word representations**: One-to-one correspondence between text units and representation elements
- Examples: “dogs” = [1, 0, 0, 0, 0]; “cats” = [0, 1, 0, 0, 0]; “cars” = [0, 0, 1, 0, 0]; “like” = [0, 0, 0, 1, 0]; “I” = [0, 0, 0, 0, 1]
- **Symbol-based document representations**: Describe a document according to which words are present, ignoring word ordering
- Examples: “I like dogs” may be represented as [1, 0, 0, 1, 1]
- Can further weigh words with Term Frequency (TF) and/or Inverse Document Frequency (IDF)
- **Issues**: Many dimensions needed (curse of dimensionality!); vectors do not reflect semantic similarity

Distributed Text Representations: Embeddings

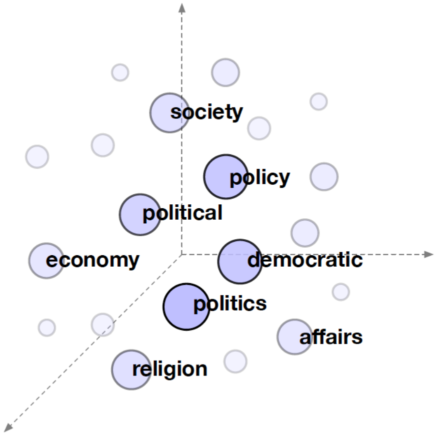
- The distributional hypothesis: “A word is characterized by the company it keeps”
  - Words used and occur in the same contexts tend to purport similar meanings
- Distributed representations (i.e., embeddings)
  - The representation of any text unit is distributed over all vector dimensions as continuous values (instead of 0/1s)
  - **Advantage**: Vectors are dense and lower-dimensional, better at capturing semantic similarity
- Distributed representations are usually learned based on the distributional hypothesis—vector space similarity reflects semantic similarity
- Distributed representations are the foundations of language models

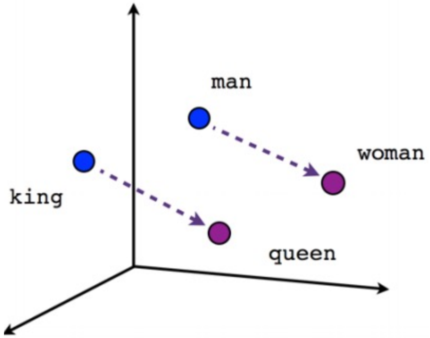
Distributed Text Representations: Embeddings

Represent words as vectors

Representations contain semantic information

Text corpus





Learning Word Embeddings

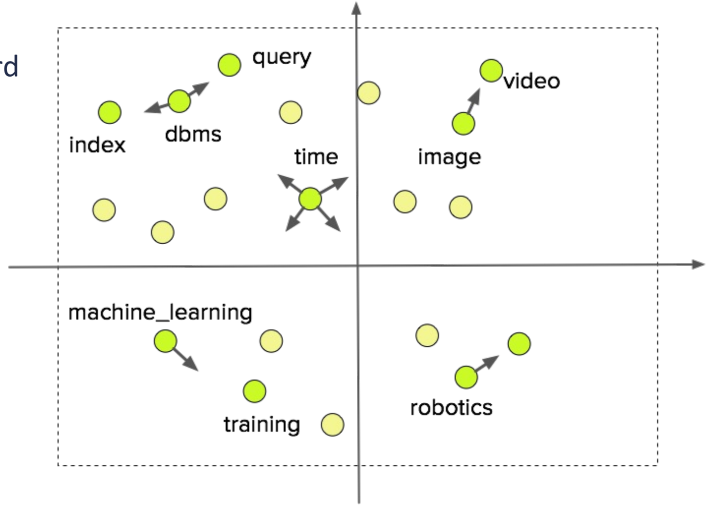
- General idea of word2vec:
  - Maximize the probability of observing context words based on target words
  - As a result, semantically similar terms are more likely to have close embeddings

Co-occurred words in a **local context window**

Target word

Training objective: $$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^\top v_{w_I}\right)}$$



Paper: (word2vec) https://arxiv.org/pdf/1310.4546.pdf

Agenda

- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

UNIVERSITY of VIRGINIA

Contextualized Text Representations

- Why aren't word embeddings enough?
- Word embeddings are static (context-free), but word meanings are not
  - Each word has one representation regardless of specific contexts it appears in
- Example: "bank" is a polysemy, but only has one representation
- Solution: learn contextualized representations by injecting context information into words via advanced model architectures

"Open a **bank** account"   "On the river **bank**"

shared representation

Transformer for Contextualized Sequence Modeling

Transformer block overview

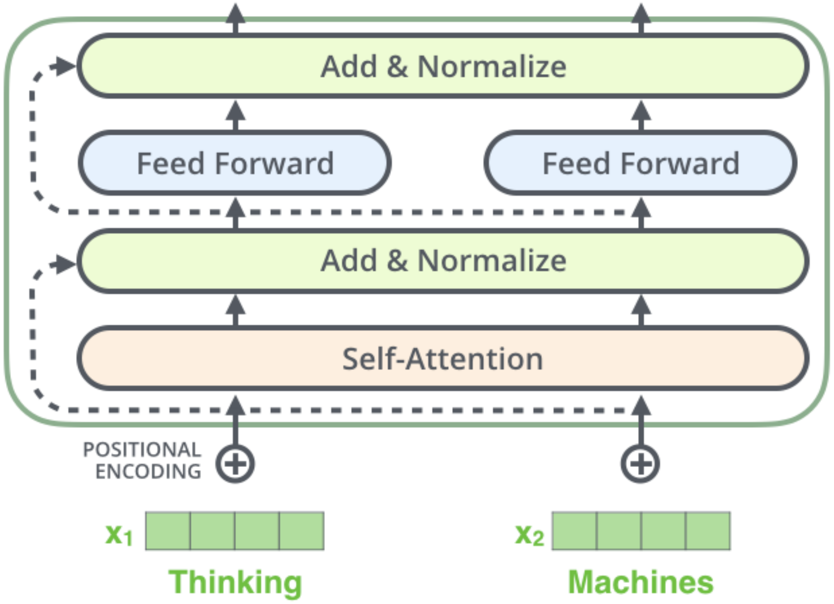


Figure source: https://jalammar.github.io/illustrated-transformer/

Transformer: Self-Attention Mechanism

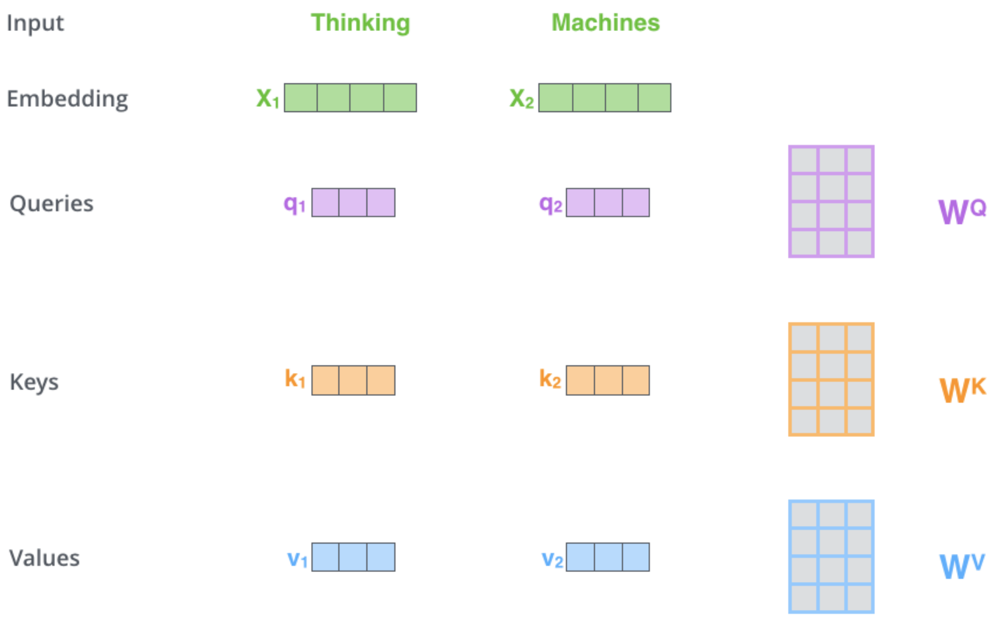
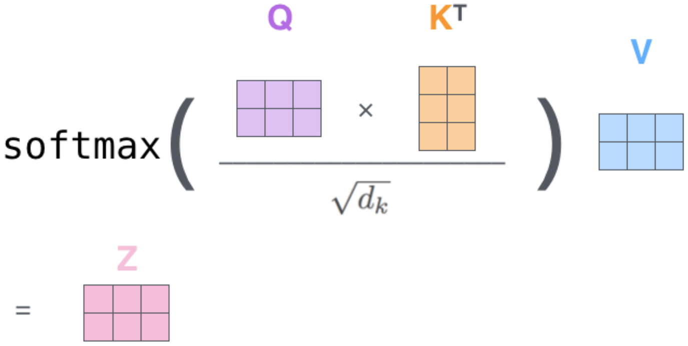


Figure source: https://jalammar.github.io/illustrated-transformer/

Transformer: Self-Attention Computation



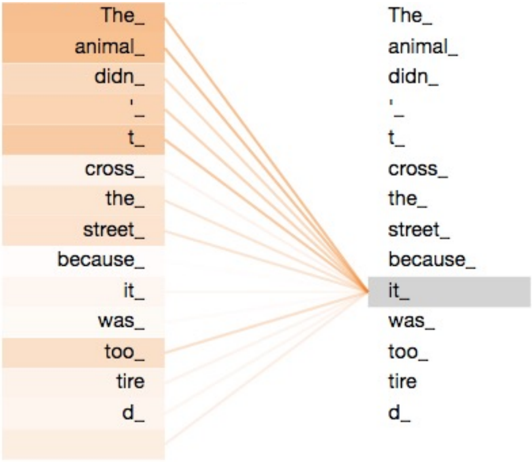
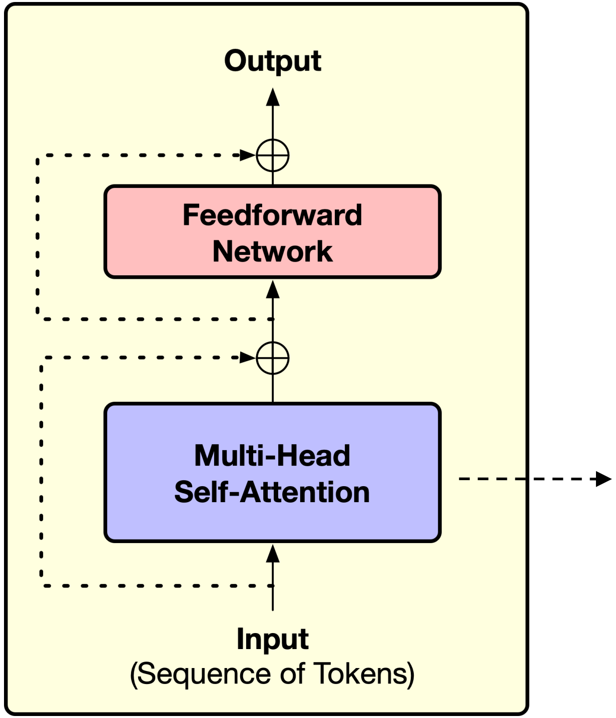


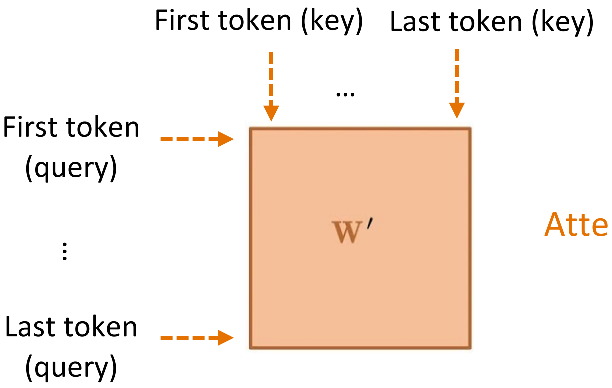
Figure source: https://jalammar.github.io/illustrated-transformer/

Agenda

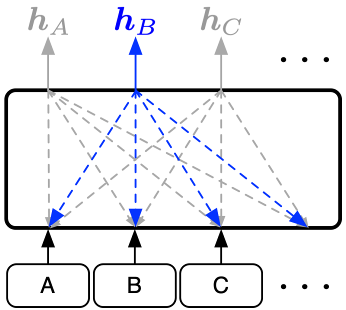
- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

Language Model Architecture: Encoders





Attention Matrix

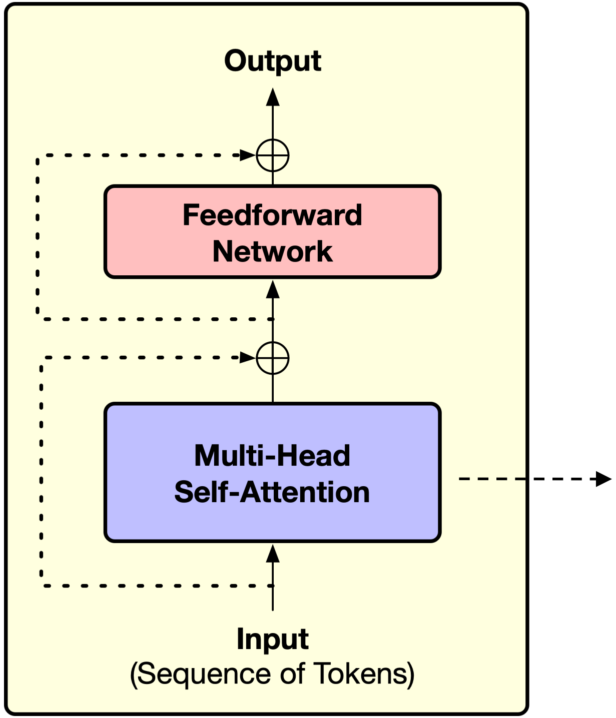


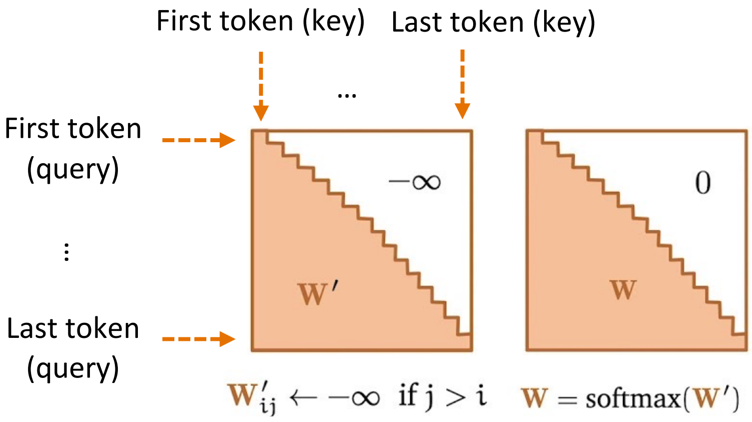
every token attends to all tokens

Bidirectional Self-Attention

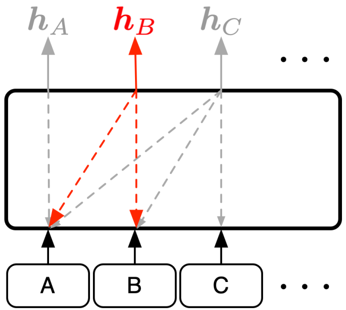
Transformer Encoders

Language Model Architecture: Encoders





Attention Matrix (before/after softmax)



every token attends to its previous tokens

Unidirectional Self-Attention

Transformer Decoders

Transformer Encoders vs. Decoders

- Encoders:
  - Each token can attend to all other tokens
  - Suitable for natural language understanding (NLU) tasks
- Decoders:
  - Each token can only attend to previous tokens
  - Suitable for natural language generation (NLG) tasks

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

NLG:
Text summarization
Machine translation
Dialogue system
Question answering
…

Agenda

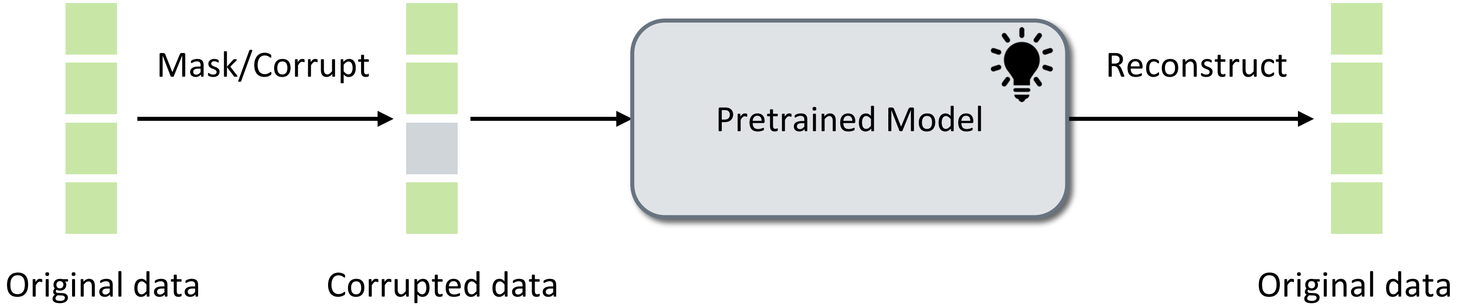
- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

Overview of Pretraining

- The “pretrain-finetune” paradigm has proven very successful in building language models for NLP tasks
- **Pretraining**: Train Transformer-based language models via self-supervised objectives on large-scale general-domain corpora
- **Fine-tuning**: Adapt the pretrained language models (PLMs) by further training on task-specific data (task-specific fine-tuning) or general-purpose data (language model alignment)
- The power of pretraining: Encode generic linguistic features and knowledge learned from large-scale data, which can be effectively transferred to the downstream applications

Overview of Pretraining

- 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!

Agenda

- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

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 objective

$$\mathcal{L}_{\mathrm{LM}} = -\sum_i \log p(x_i \mid \underbrace{x_{i-k}, \ldots, x_{i-1}}_{\text{previous tokens as contexts}})$$

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

Decoder Pretraining: Illustration

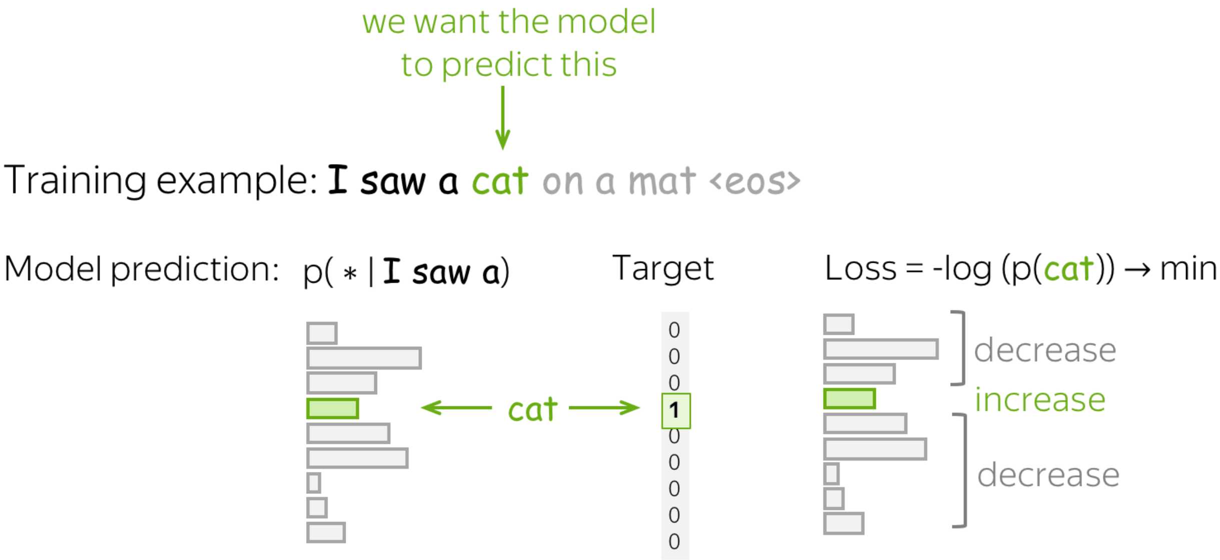


Figure source: https://lenavoita.github.io/nlp_course/language_modeling.html

Language Modeling as 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

(Few-Shot) In-Context Learning

After pretraining, decoder models can do in-context learning (next lecture!)

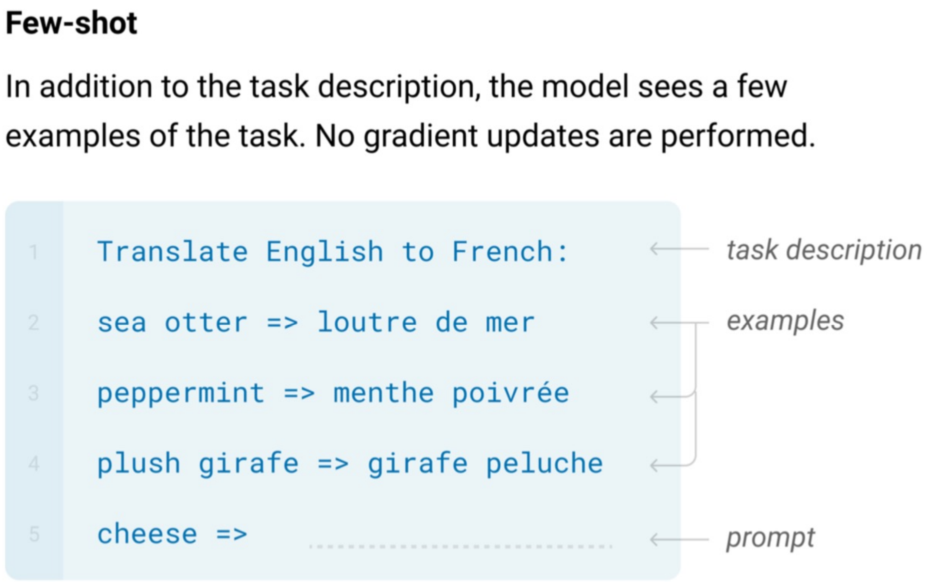
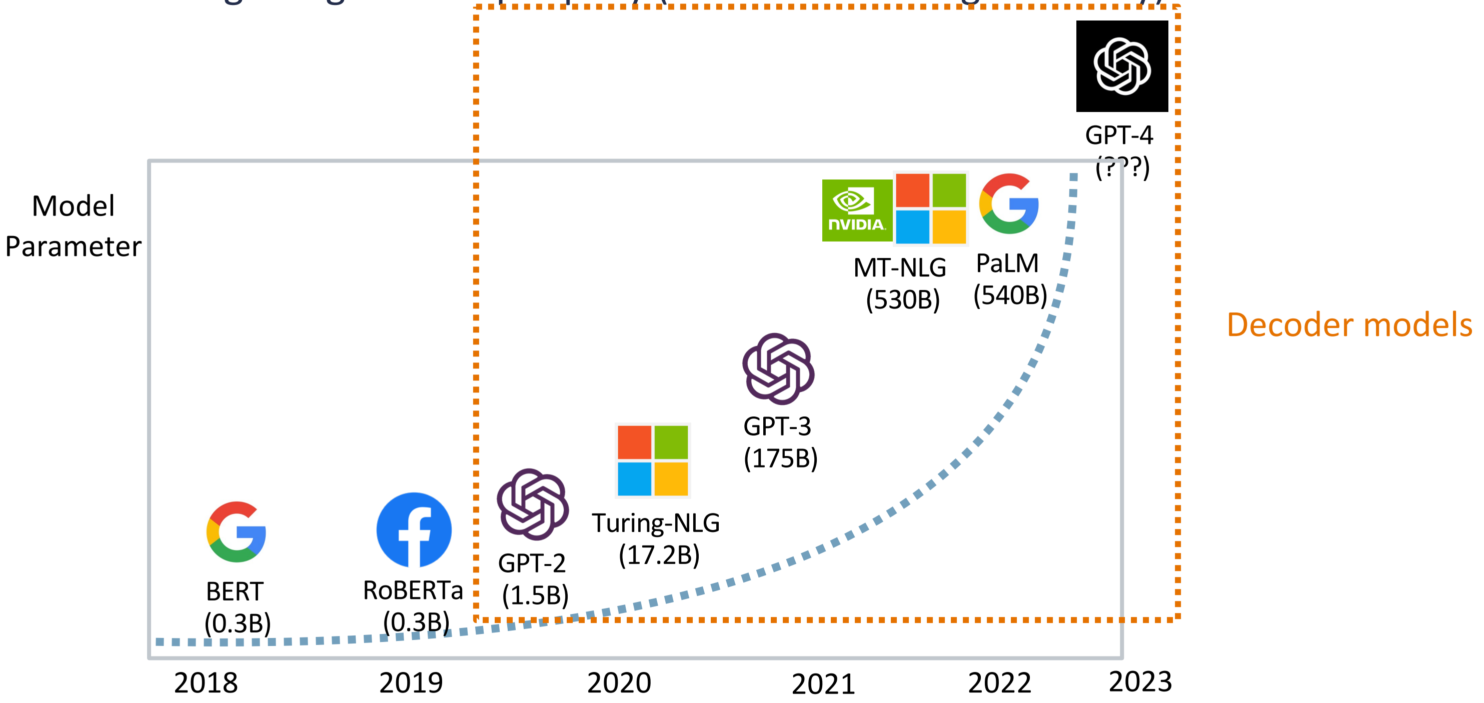


Figure source: https://ai.stanford.edu/blog/in-context-learning/

Large Language Models (Decoder Models)

Decoder models are getting scaled up rapidly (next week: emergent ability)!



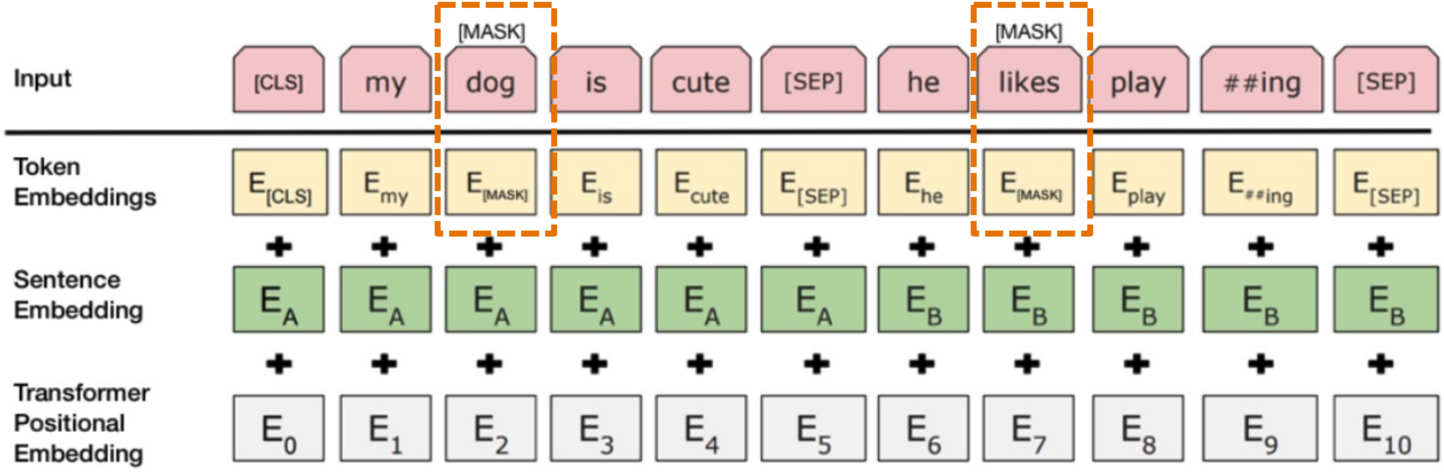
UNIVERSITY of VIRGINIA

Agenda

- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

Encoder Pretraining: BERT

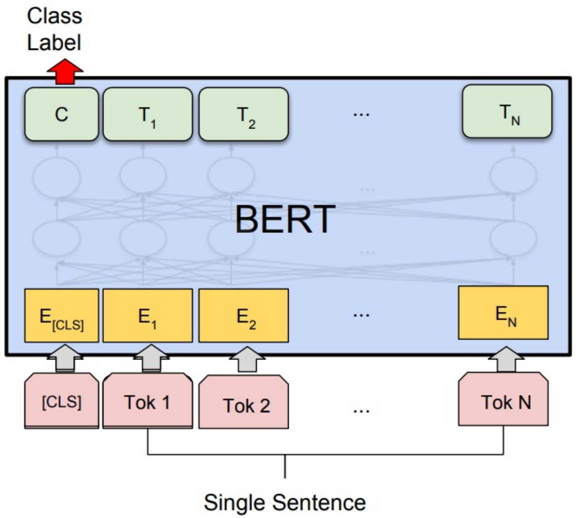
- BERT pretrains encoder models with bidirectionality
- Masked language modeling (MLM): With 15% words randomly masked, the model learns bidirectional contextual information to predict the masked words



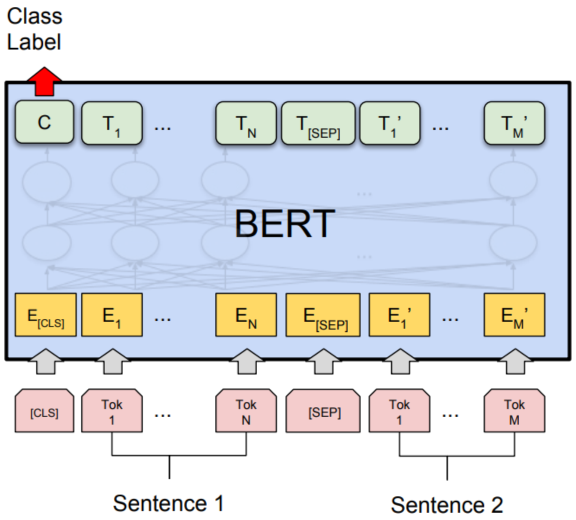
Paper: (BERT) https://arxiv.org/pdf/1810.04805.pdf

BERT Fine-Tuning

Fine-tuning pretrained BERT models takes different forms depending on task types



Single sequence classification



Sequence-pair classification

BERT vs. GPT on NLU tasks

- BERT outperforms GPT-1 on a set of NLU tasks
- Why are encoder models better than decoder models for NLU?
- Are encoder models still better than state-of-the-art (large) decoder models?

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**

Paper: (Can ChatGPT Understand Too?) https://arxiv.org/pdf/2302.10198.pdf

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

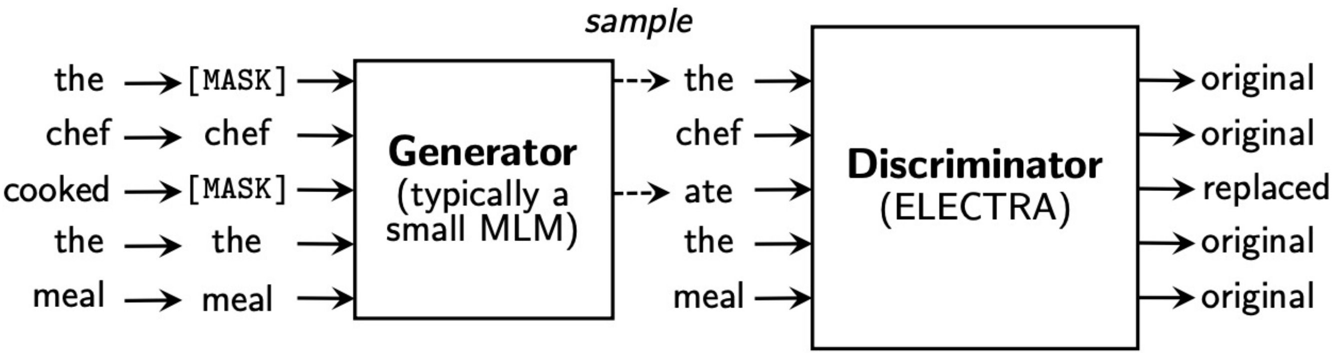
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$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

Paper: (RoBERTa) https://arxiv.org/pdf/1907.11692.pdf

UNIVERSITY of VIRGINIA

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



Paper: (ELECTRA) https://arxiv.org/pdf/2003.10555.pdf

ELECTRA Performance

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

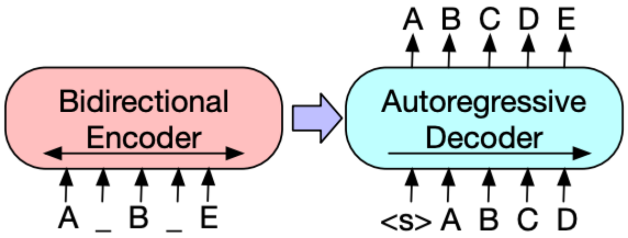
Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	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**

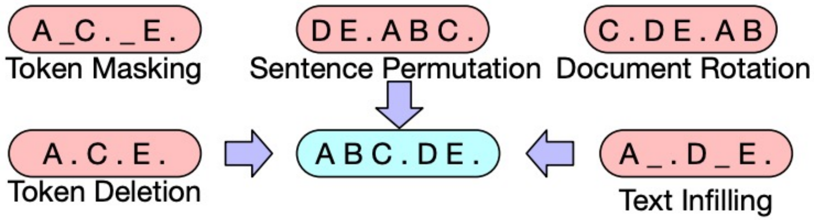
Agenda

- Language Model Architecture
  - Word Embeddings
  - Transformer
  - Encoder and Decoder Architecture
- Language Model Pretraining
  - Decoder Pretraining
  - Encoder Pretraining
  - Encoder-Decoder Pretraining

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





Paper: (BART) https://arxiv.org/pdf/1910.13461.pdf

UNIVERSITY of VIRGINIA

BART Performance

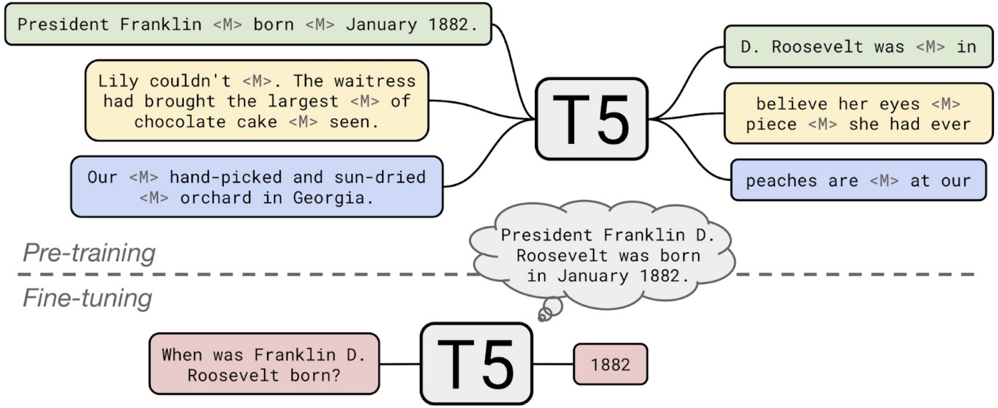
- Comparable to encoders on NLU tasks
- Good performance on NLG tasks

	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 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**

UNIVERSITY of VIRGINIA

Summary

- We introduced the language model architectures
  - Input tokens represented as dense vectors (embeddings)
  - Transformers learn contextualized representations
  - Transformer encoder vs. decoder
- We introduced pretraining methods for various language model architectures
  - Pretraining allows the models to acquire general linguistic & world knowledge
  - Different pretraining objectives/settings need to be designed for different architectures
  - Under the same model sizes, encoder models are better at NLU tasks; decoder models are used for NLG tasks
  - Encoder-decoder models: Good NLU & NLG performance, but less efficient than decoder models for NLG (discussed in efficiency lectures)
- We will mainly focus on decoder models in this course
  - Current large language models (LLMs) are (almost) all decoder models
  - Decoder models are more versatile for various applications
  - Decoder models can be scaled up to extremely large sizes (next week)

Next Time

- Large language models (LLMs)
  - GPT-3
  - LLaMA-2
- In-context learning (ICL)
  - What matters for ICL?
  - Why are LLMs able to perform ICL?

Thank You!

Yu Meng
University of Virginia
yumeng5@virginia.edu