Washington University in St. Louis
James McKelvey School of Engineering

CSE 561A: Large Language Models

Fall 2025

Lecture 2: Language Model Architectures and Pre-training

Course Reminder

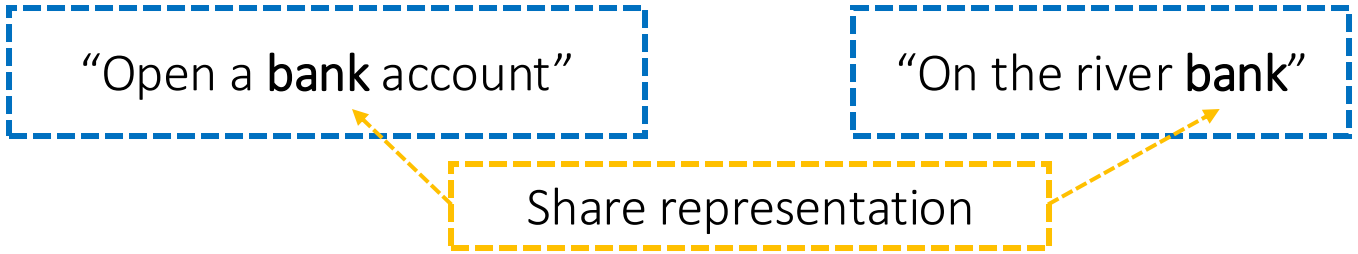
- Sign-up for a presentation slot: https://docs.google.com/spreadsheets/d/1NPgbSOqR5KbmpfVsJpatABgqwNMGaqFssKQUWJ4D-3U/edit?usp=sharing
- When it is your turn, waitlist offers are sent via emails (by workday system). If you don’t accept it, you will not be enrolled.

Content

- **Self-Attention**
- Transformer Architecture
- Different Pre-training Objectives
  - Decoder-Only Models (GPT)
  - Encoder-Only Models (BERT)
  - Encoder-Decoder Models (T5, BART)

Recap: Context-Free Embedding

- 1) Each word is mapped to only one vector regardless of its context!
  - E.g. “bank” is a polysemy, but only has one representation



- 2) It does not consider the order of words
- 3) It treats the words in the context window equally
- Solution: We need **contextualized** text representations!
  - Injecting context information into words via advanced model architectures

Self-Attention: Intuition

- Example:
- “The chicken didn’t cross the road because it was too tired”

- Suppose we are learning attention for the word “**it**”
- With self-attention, “**it**” can decide which other words in the sentence it should focus on to better understand its meaning
- Might assign high attention to “**chicken**” (the subject) & “**road**” (another noun)
- Might assign less attention to words like “the” or “didn’t”

Self-Attention: Example

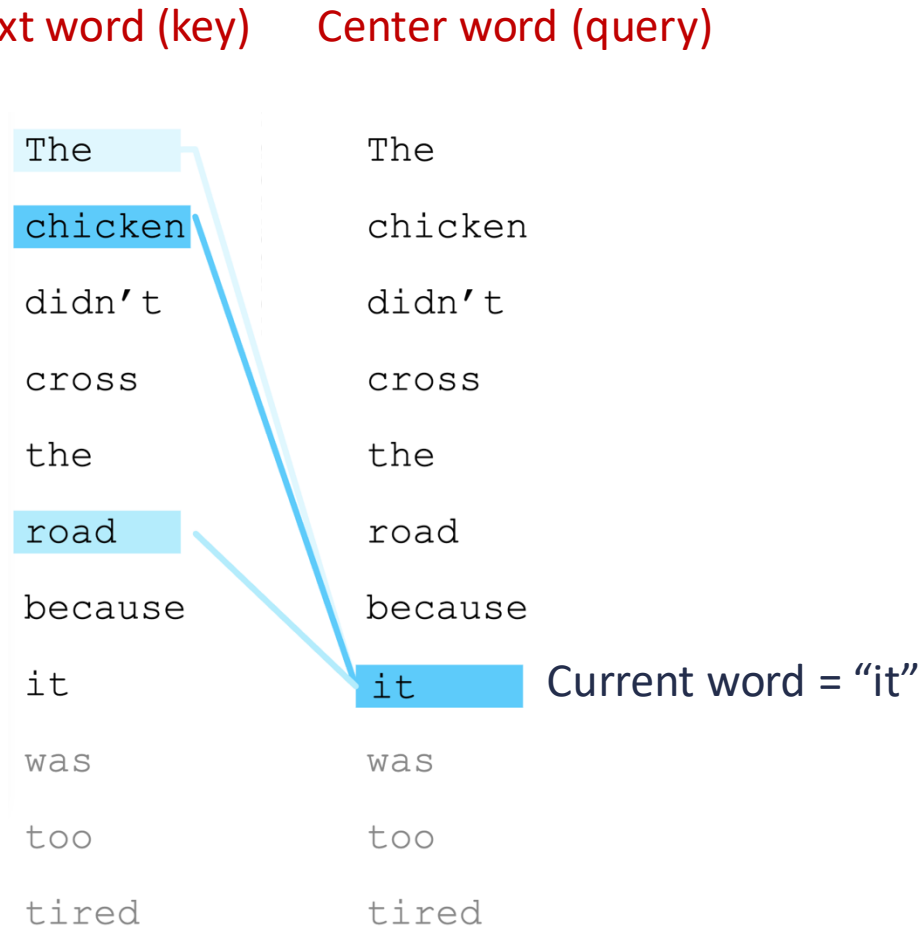
- Goal: Derive a **contextual** representation for the center word as a weighted sum of context representations!

Center word representation

Context word representation

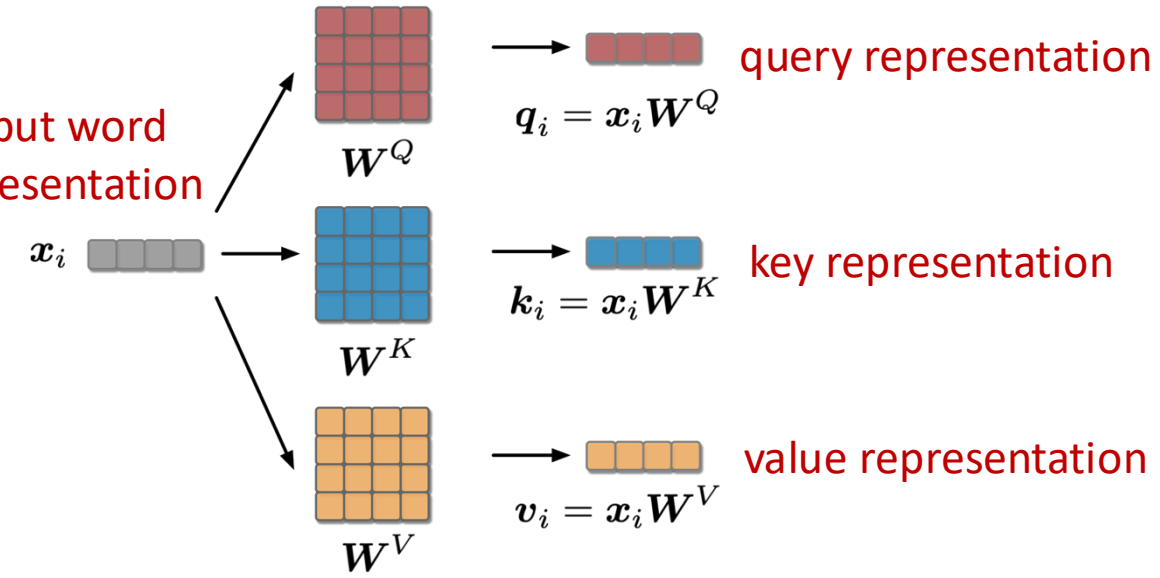
$$\boldsymbol{a}_i = \sum_{x_j \in \boldsymbol{x}} \alpha_{ij} \boldsymbol{x}_j, \quad \sum_{x_j \in \boldsymbol{x}} \alpha_{ij} = 1$$

Attention score $i \rightarrow j$, summed to 1



Self-Attention: Linear Projections

- As opposed to center/context representations for each word, we use three different representations for Query, Key, and Value vectors.
- **Query Vector (Q)**:
  - Represent the current word seeking information about
- **Key Vector (K)**:
  - Represent the reference (context) against which the query is compared
- **Value Vector (V)**:
  - Represent the actual content associated with each token to be aggregated as final output



Without Projection
- Same representation for Q, K, V
- Limited expressiveness
- Fixed attention patterns

Self-Attention: Matrix Calculation

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

First token (key vector)

Last token (key vector)

First token (query vector)

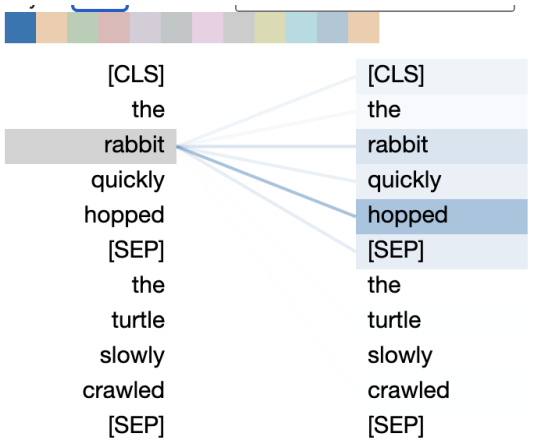
Last token (query vector)

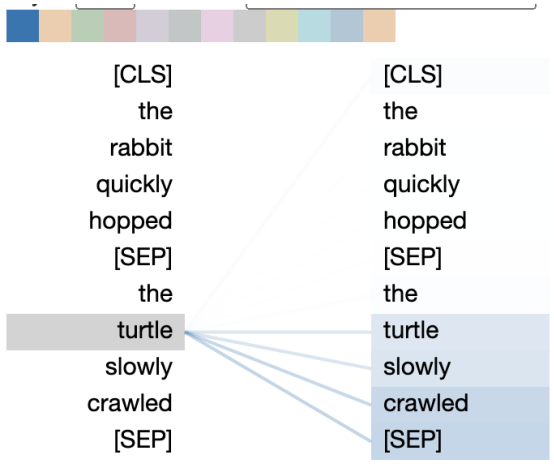
QK^T

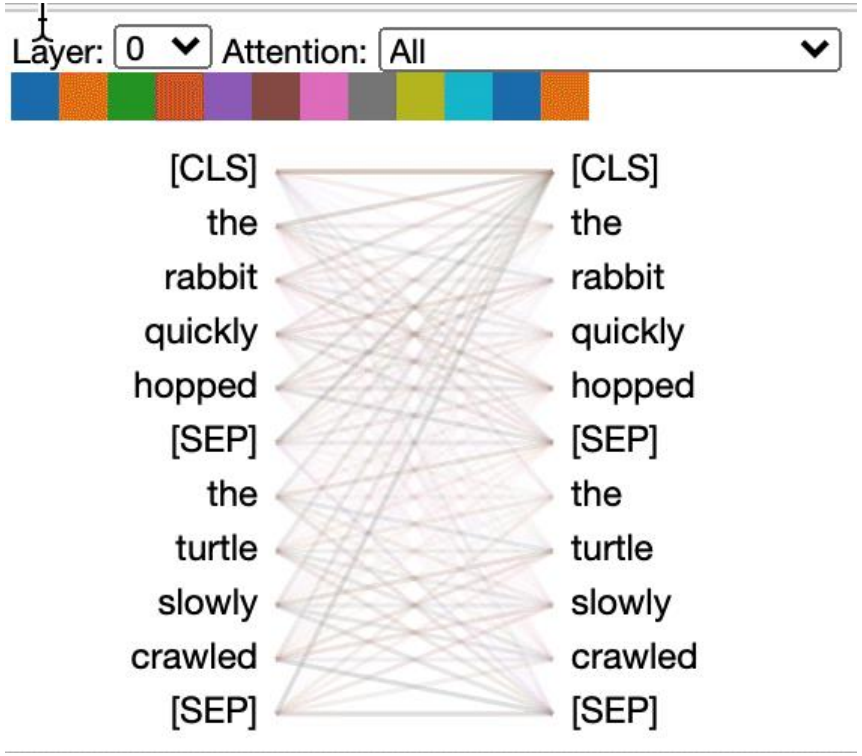
Attention Matrix

Self-Attention Demo

- Demo: https://github.com/jessevig/bertviz





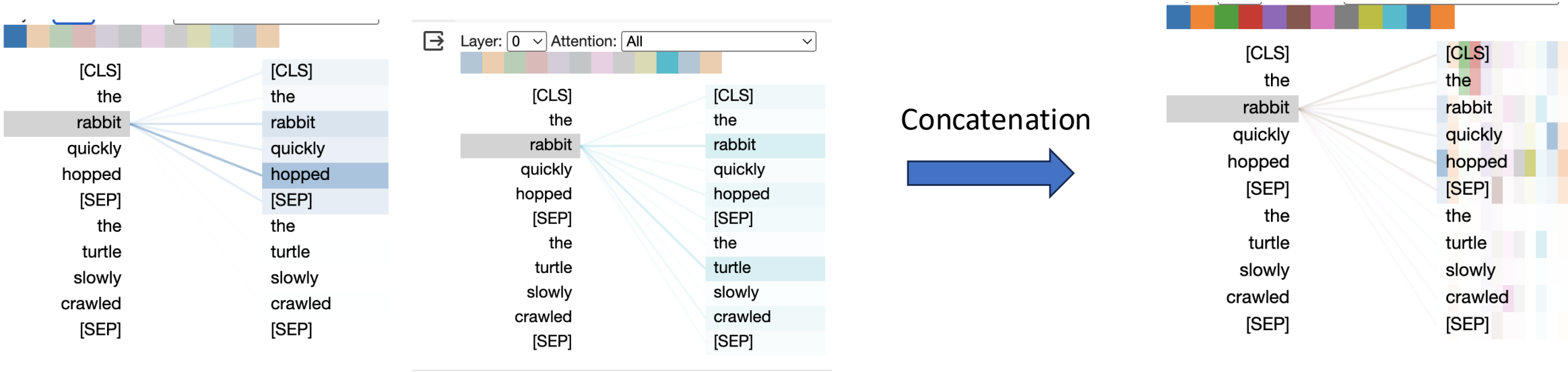


Multi-Head Attention

- Input: Multiple Independent sets of query, key, value matrices
- Output: Concatenate the outputs of attention heads
- Advantage: Each attention head focus on one subspace

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

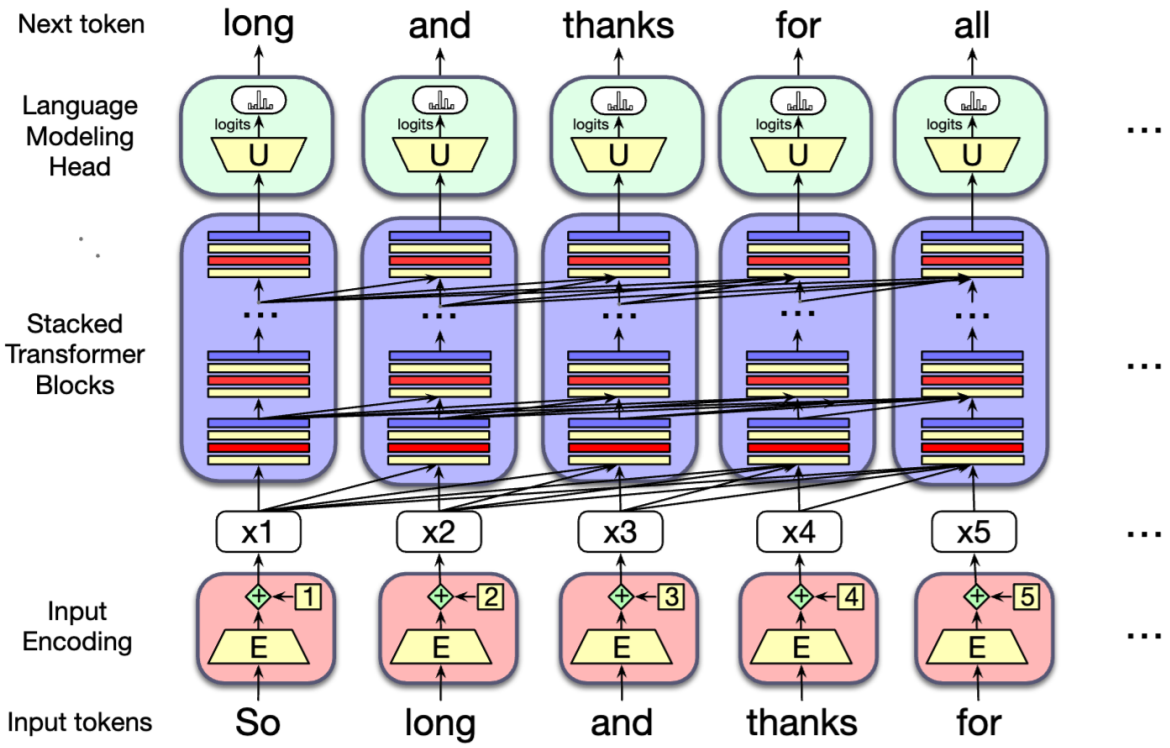


Content

- Self-Attention
- **Transformer Architecture**
- Different Pre-training Objectives
  - Decoder-Only Models (GPT)
  - Encoder-Only Models (BERT)
  - Encoder-Decoder Models (T5, BART)

Transformer: Overview

- The most important architecture for language modeling (almost all LLMs are based on Transformers)!



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

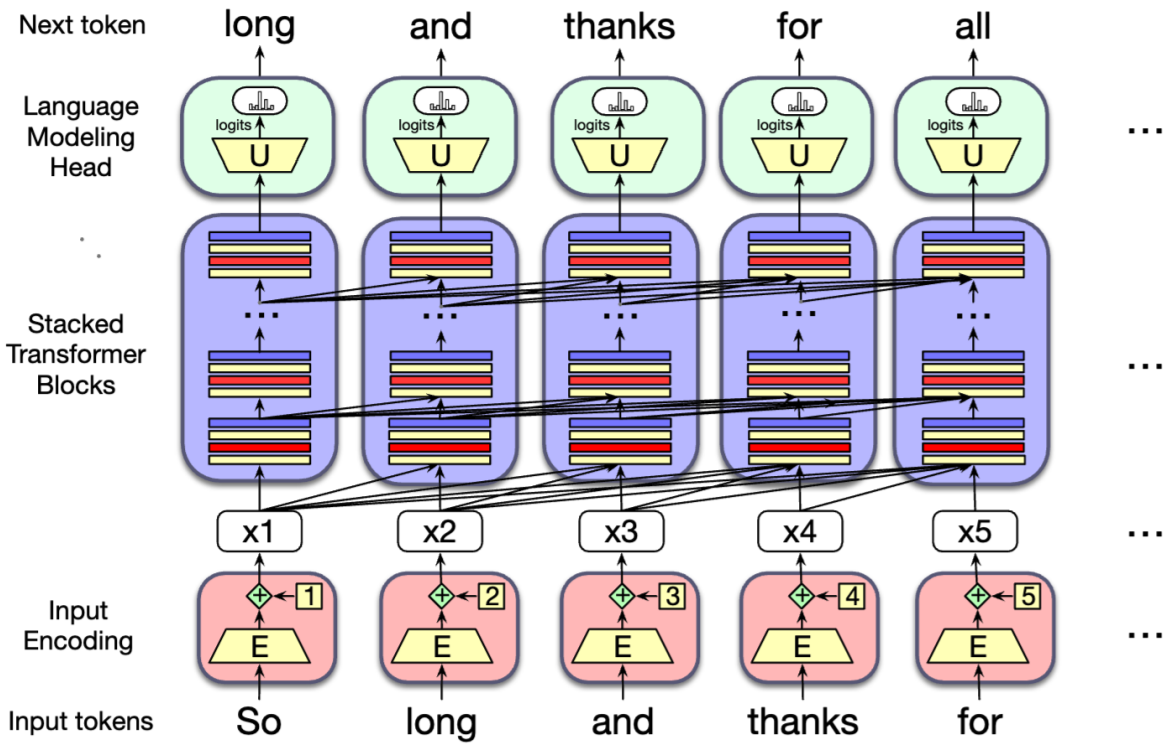
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Transformer: https://arxiv.org/pdf/1706.03762

Transformer Model Architecture

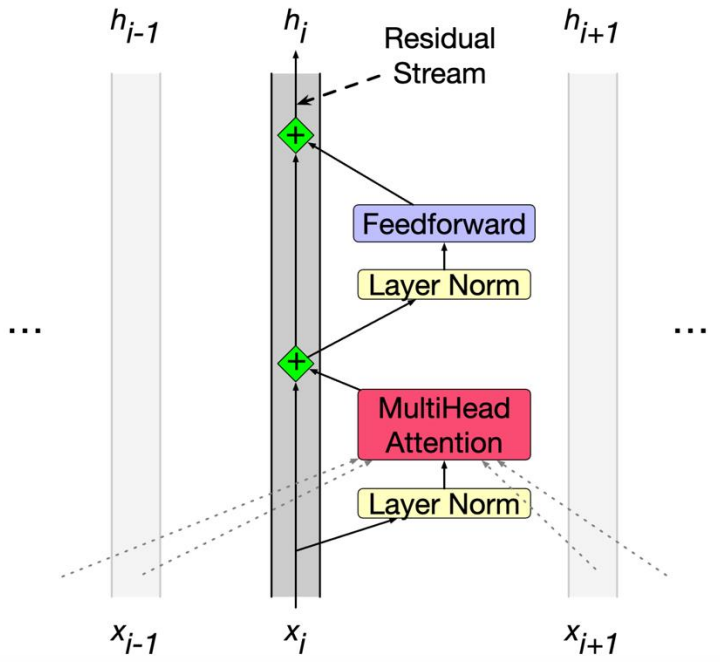
- Tokenizer + Input Embedding Layer (context-free embeddings)
- Positional Encoding Layer
- Transformer layers
  - encoder layers: text understanding
  - decoder layers: text generation
- Output: Linear + SoftMax layer for next word prediction



Transformer Layer

Each Transformer layer contains the following important components:

- Multi-head self-attention
- Layer normalization (LayerNorm)
- Feedforward network (FFN)
- Residual connections + layer norm

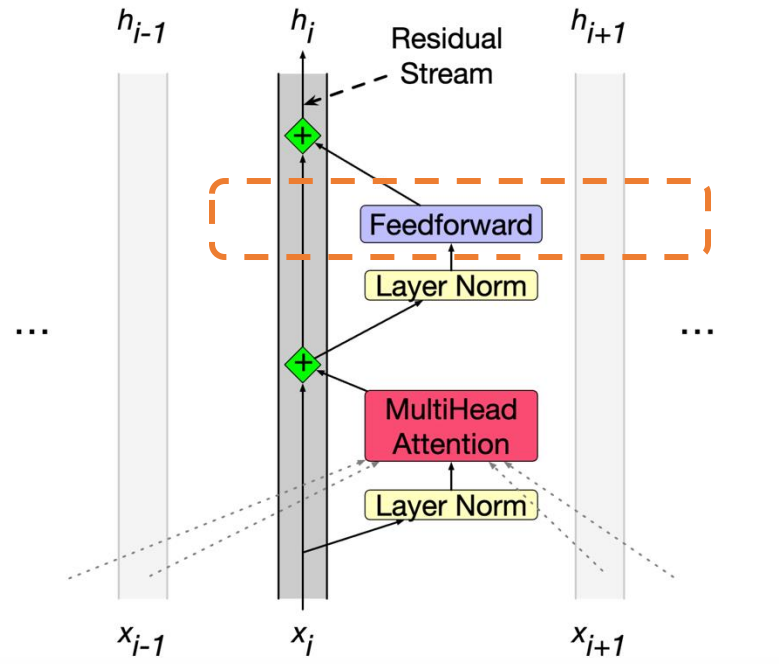


Feedforward Network (FFN)

- 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 FFN weights applied to every token
- Weights are different across different Transformer layers

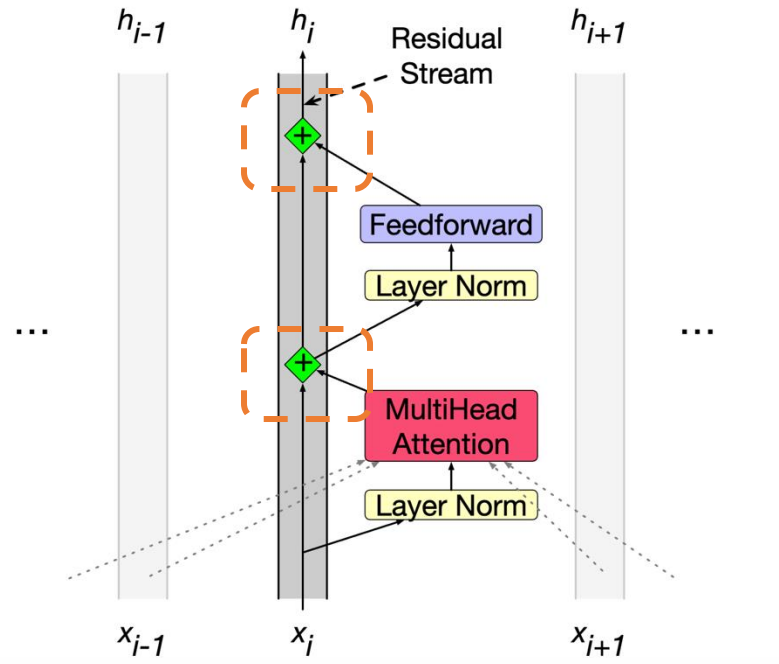


Residual Connections

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

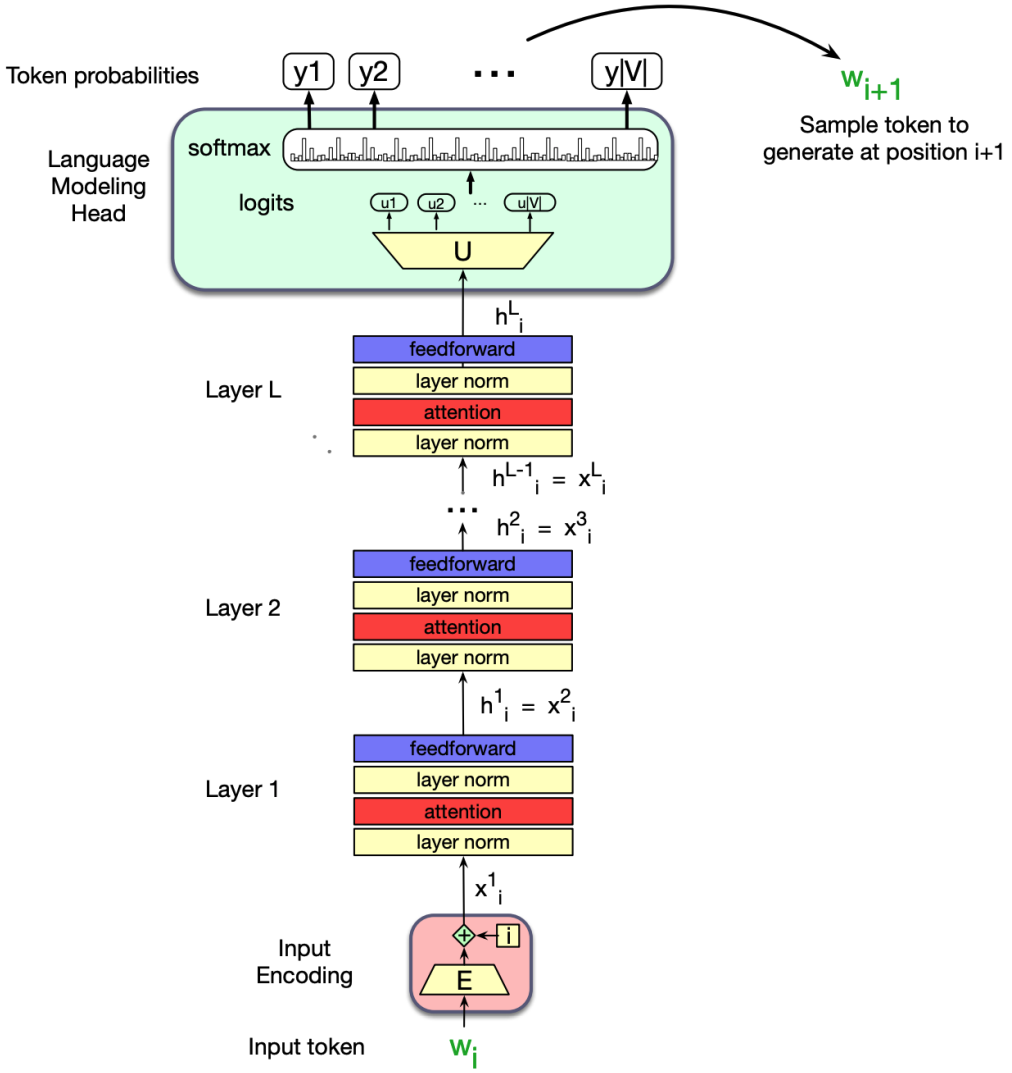
$$\boldsymbol{y} = \boldsymbol{x} + f(\boldsymbol{x})$$

- Benefits
  - Address the vanishing gradient problem
  - Facilitate information flow across the network
  - Help scale up model: help with training very deep networks



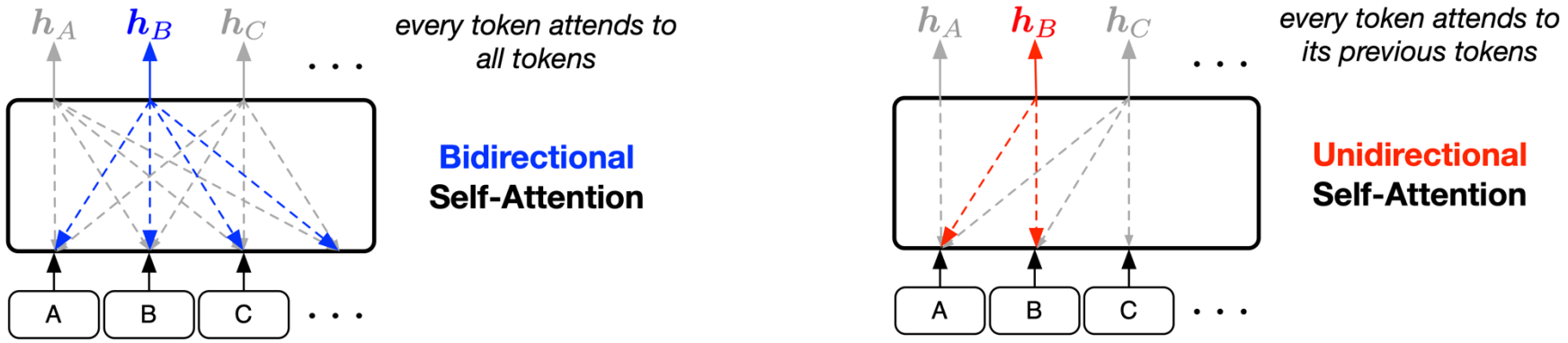
Stacked Transformer Layers

- Each layer processes and refines representations from previous layer
  - Each layer has its own unique parameters
  - Within a layer, same FFN weights applied to all tokens
- Information flow
  - Lower layers: capture local patterns, syntax information
  - Middle layers: merge local information
  - Upper layers: model high-level semantics, task-specific features
- Language model head is added to the final layer
  - Usually apply the weight tying trick (share weights between input embeddings and the output embeddings



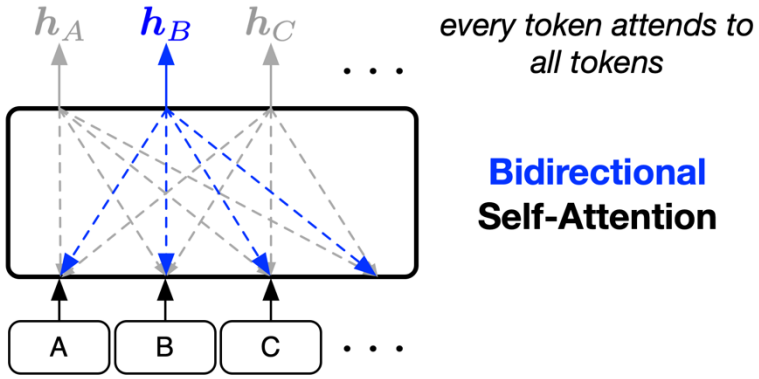
Encoder vs. Decoder

- Encoder models (bidirectional attention): e.g., BERT
  - Used for text understanding tasks: text classification, named entity recognition
- Decoder models (unidirectional attention): e.g., GPT-2
  - Used for text generation tasks: machine translation, document summarization, question answering



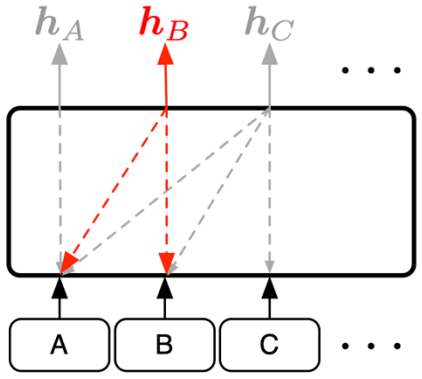
- Both encoder and decoder models can be pre-trained via large corpus, but need different training objectives!

Different Attention Matrix



every token attends to all tokens

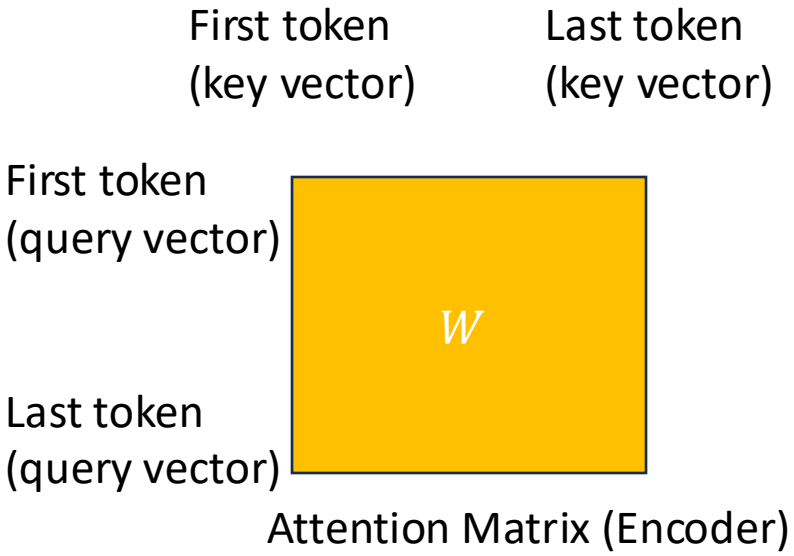
Bidirectional Self-Attention



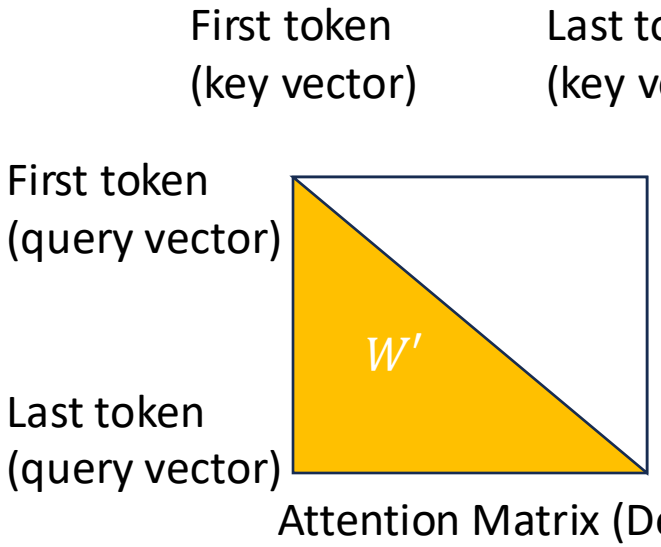
every token attends to its previous tokens

Unidirectional Self-Attention

- Attention Matrix



Attention Matrix (Encoder)



Attention Matrix (Decoder)

Content

- Self-Attention
- Transformer Architecture
- **Different Pre-training Objectives**
  - Decoder-Only Models (GPT)
  - Encoder-Only Models (BERT)
  - Encoder-Decoder Models (T5, BART)

Pretraining: Motivation

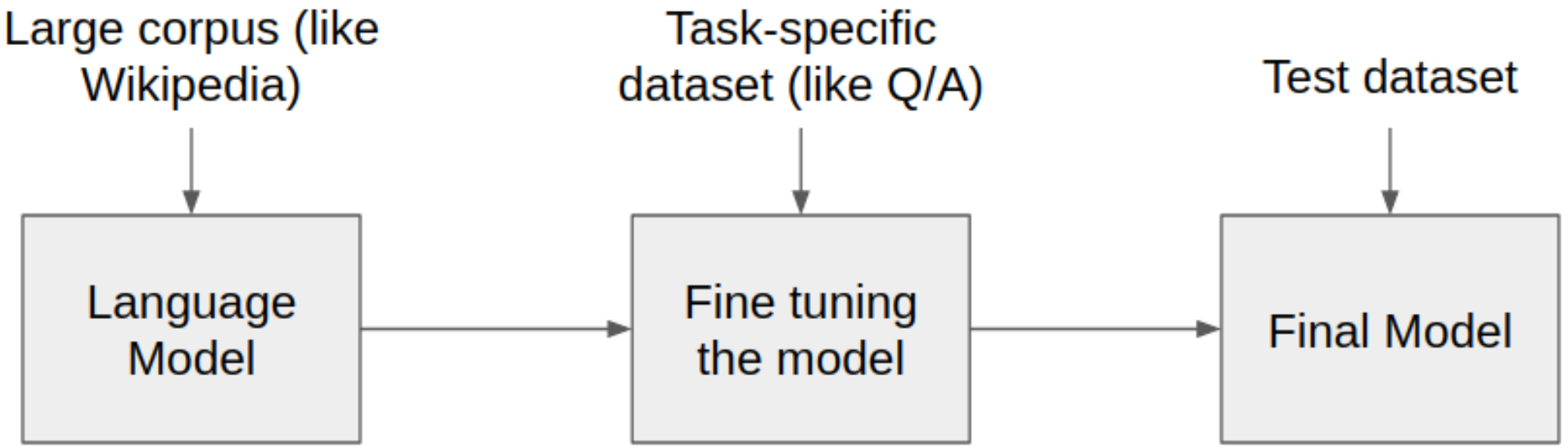
- Before pretraining became prevalent in NLP, most NLP models were trained from scratch for each downstream task
- **Data scarcity**: many NLP tasks do not have large labeled datasets available (costly to obtain)
- **Poor generalization**: models trained from scratch on specific tasks do not generalize 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

Pretraining: Data

- 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

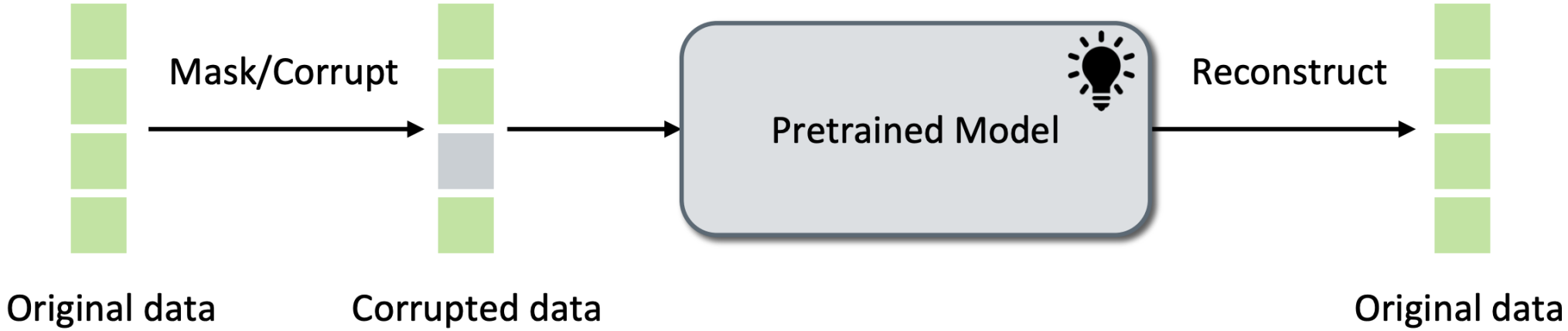
Pretrain-Finetune Paradigm

- “Pretraining”: Train deep language models (usually Transformer models) via **self-supervised** objectives on **large-scale general-domain corpora**
- “Fine-tuning”: Adapt the pretrained language models (PLMs) to downstream tasks by further training on task-specific data
- The power of PLMs: Encode generic linguistic features and knowledge learned through large-scale pretraining, which can be effectively transferred to the target applications



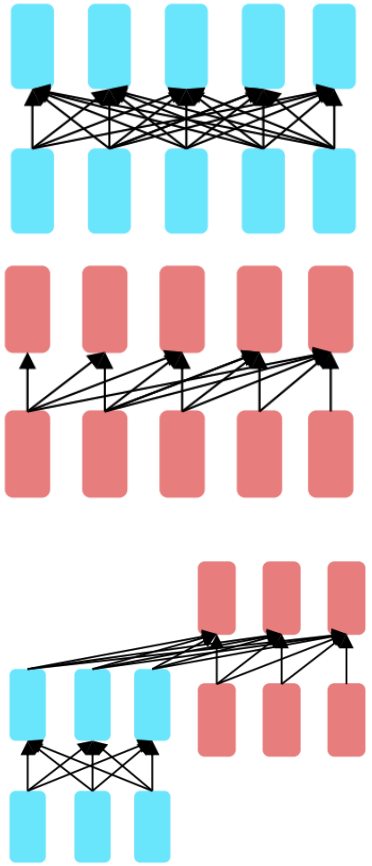
Overview of Pretraining

- Self-supervised learning
- Make a part of the input unknown to the model
- Let the model predict that unknown part based on the known part



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



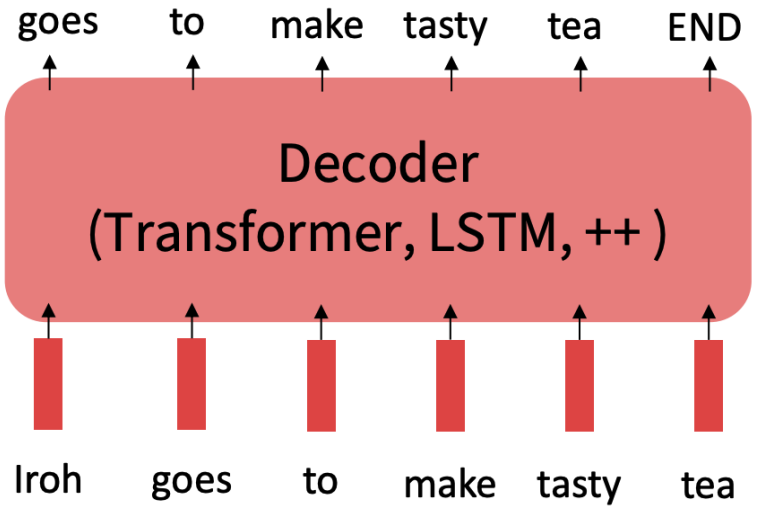
NLU:
Text classification
Named entity recognition
Relation extraction
Sentiment analysis
…

NLG:
Text summarization
Machine translation
Dialogue system
Question answering
…

Decoder Pretraining (GPT)

- Decoder architecture is the prominent choice in large language models
- Pretraining decoders are first introduced in GPT (generative pretraining) models
- Recall the language modeling task: Model $p_\theta(w_t|w_{1:t-1})$, the probability distribution over words given their past contexts.
- Follow the standard language modeling (cross-entropy) objective

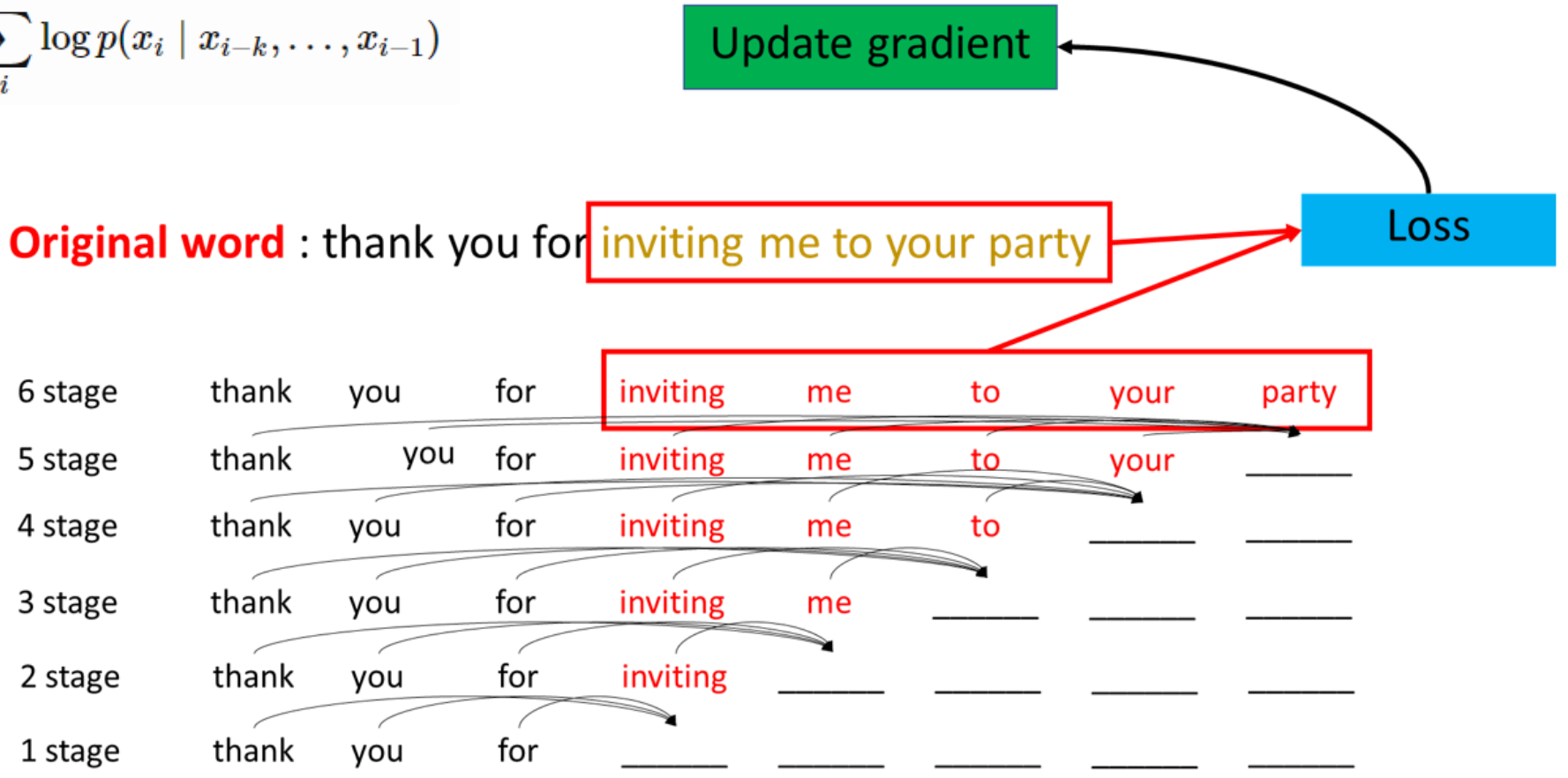
all previous tokens as context



[1] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog.

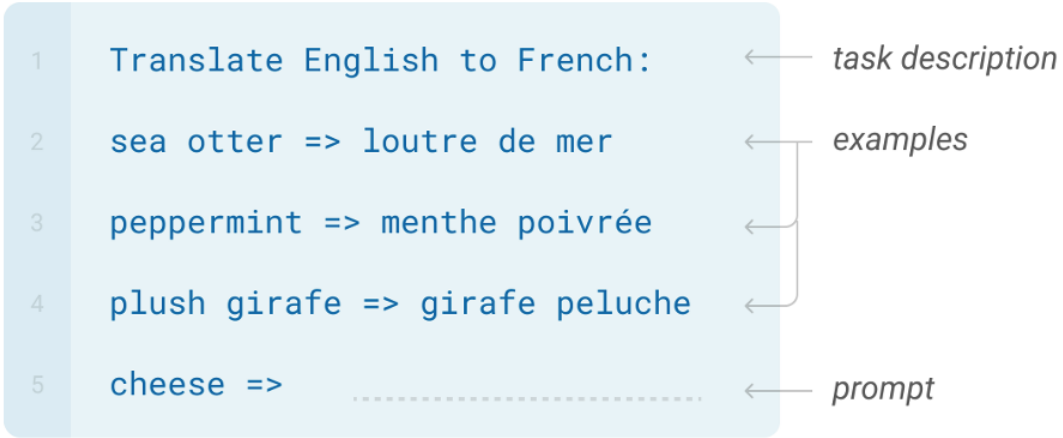
Decoder Pretraining

$$\mathcal{L}_{\text{LM}} = -\sum_{i} \log p(x_i \mid x_{i-k}, \ldots, x_{i-1})$$



Usage of Decoder Models

- Question Answering

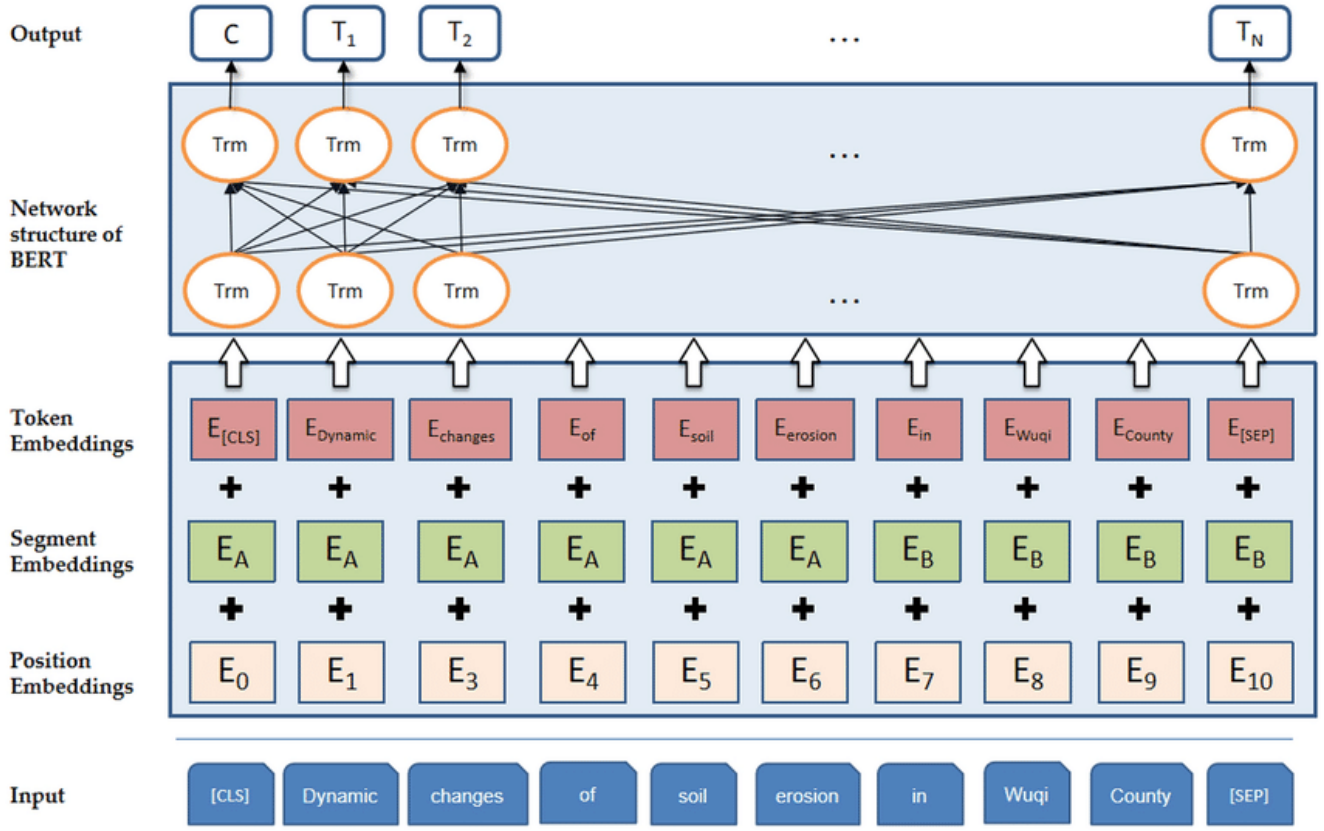


Content

- Self-Attention
- Transformer Architecture
- Different Pre-training Objectives
  - Decoder-Only Models (GPT)
  - **Encoder-Only Models (BERT)**
  - Encoder-Decoder Models (T5, BART)

BERT Model Architecture

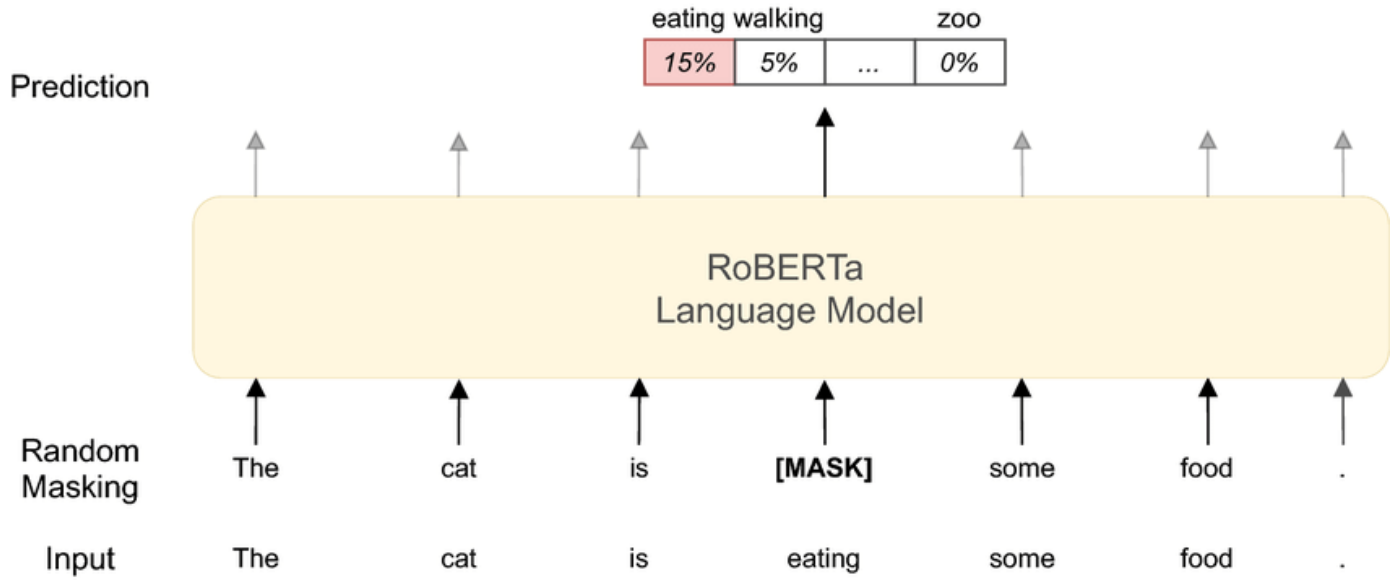
- Bidirectional attention: each token can attend to its left and right context for self-attention



[1] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Devlin et al. NAACL'19.

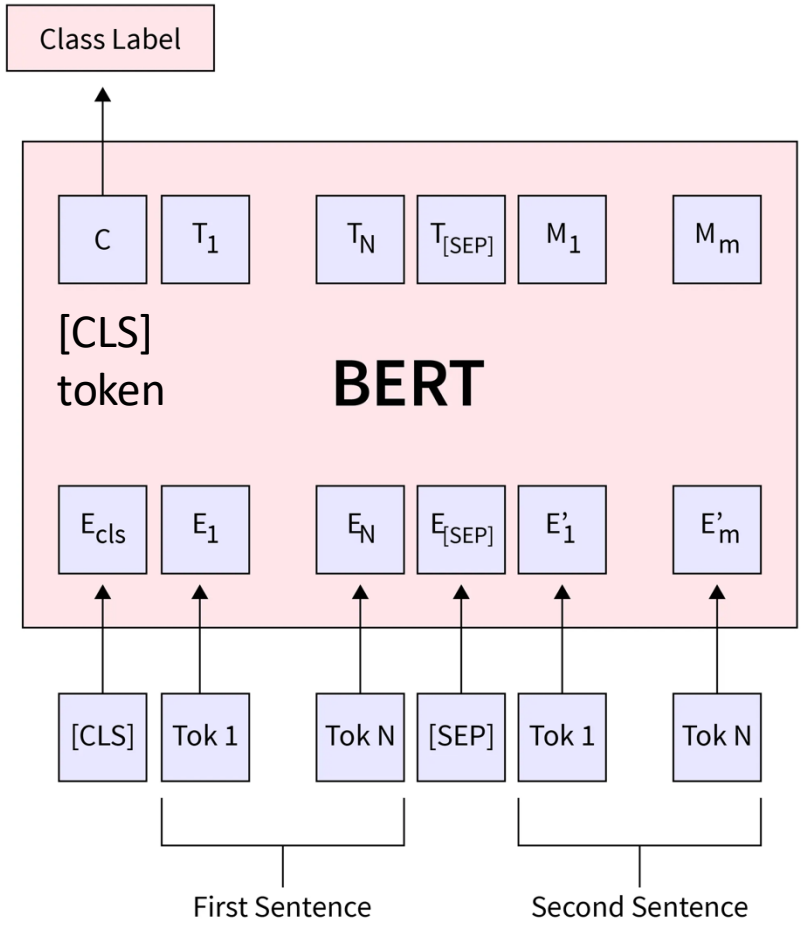
Encoder Model Pre-training

- Pre-training objectives
  - Masked language modeling (MLM) + Next Sentence Prediction
  - MLM: 15% of tokens are randomly corrupted (masked) for model prediction



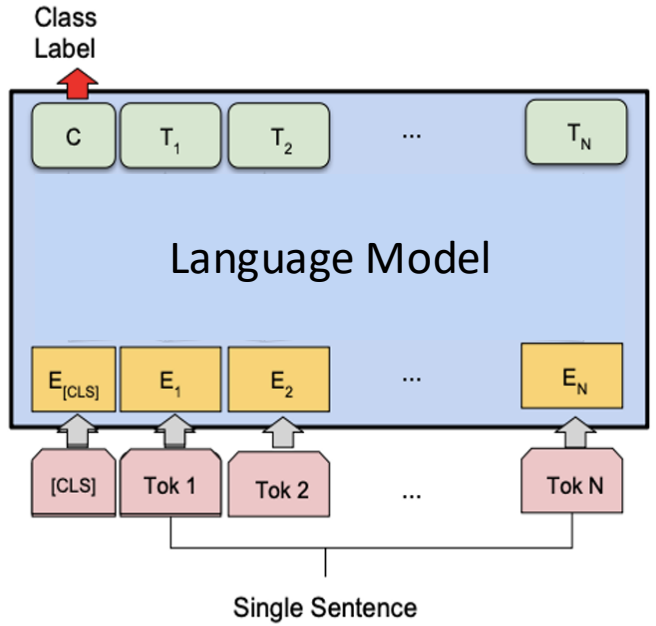
Next Sentence Prediction

- Next Sentence Prediction (NSP)
- 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
- Positive samples: two contiguous sentences in the corpus.
- Negative samples: sample another sentence for sentence A.
- Class Labels: <is_next, not_next>



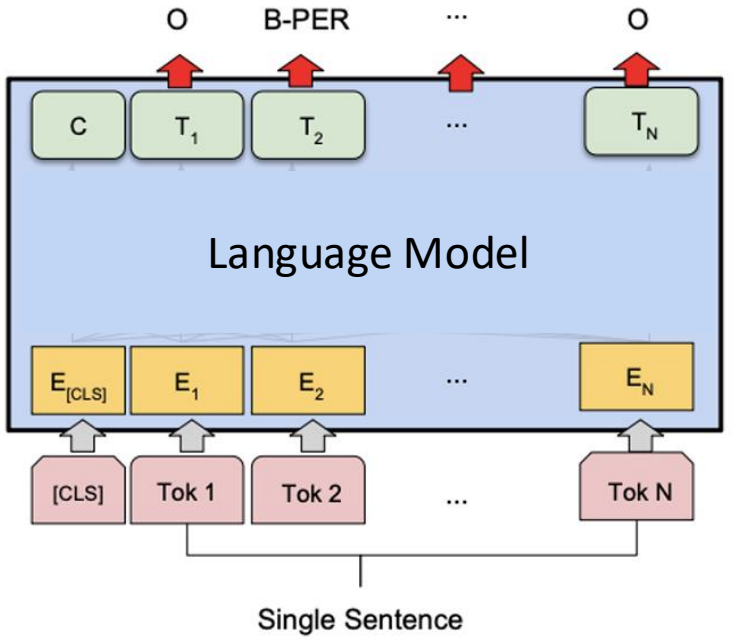
Usage of Encoder Models (I)

- Sentence classification tasks
  - Text Classification Tasks
    - Input: The bike is too small and I want to return it.
    - Output: <refund, return, check_status>
  - Sentiment Analysis
    - Input: The restaurant is crowded and I waited my food for 30 minutes!
    - Output: <positive, negative>



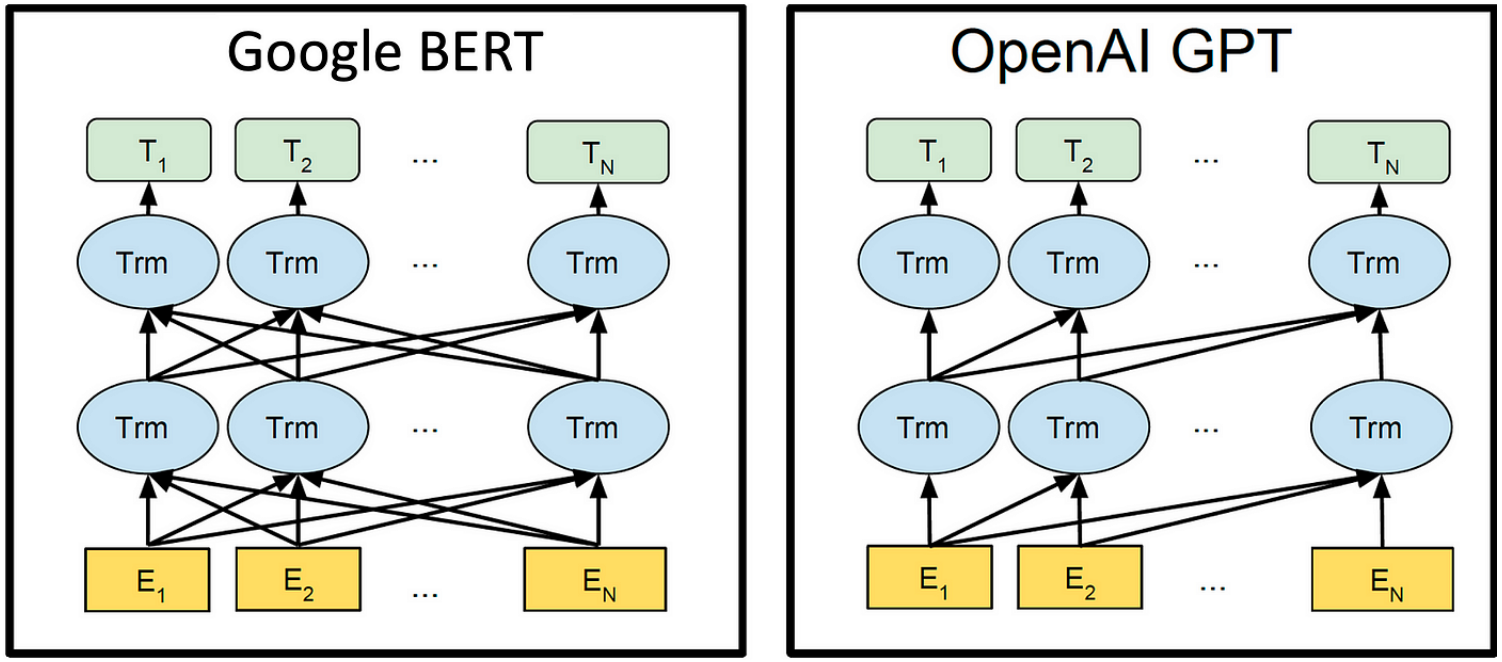
Usage of Encoder Models (II)

- Token-level tasks
  - Named Entity Recognition
  - Input: St. Louis is located in the state of Missouri .
  - Output: <Begin-Location> <Inside-location> O O O O O O <Begin-Location> O



Comparison with GPT Model

- Training objective: MLM prediction vs. left-to-right token prediction



BERT vs. GPT on NLU tasks

- GLUE Benchmark for natural language understanding
- BERT is better at language understanding

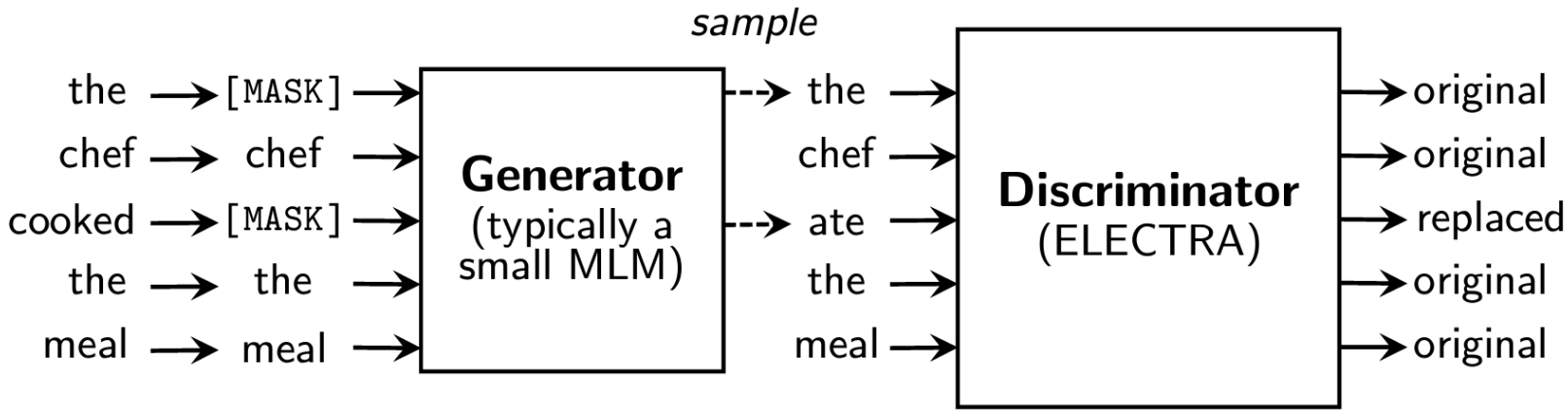
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	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**

Variants of BERT Model

- RoBERTa (RoBERTa: A Robustly Optimized BERT Pretraining Approach. Liu et al. 2019)
  - Training the model longer on more data with bigger batches
  - Remove the next sentence prediction objective
  - Dynamically change the [MASK] patterns in each epoch

Variants of BERT Model

- ELECTRA (ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. Clark et al. 2020)
  - Replaced token detection by corrupting text sequences with an auxiliary MLM
  - Works better than BERT because the input text for ELECTRA does not contain [MASK] tokens (no discrepancy between training and test data)

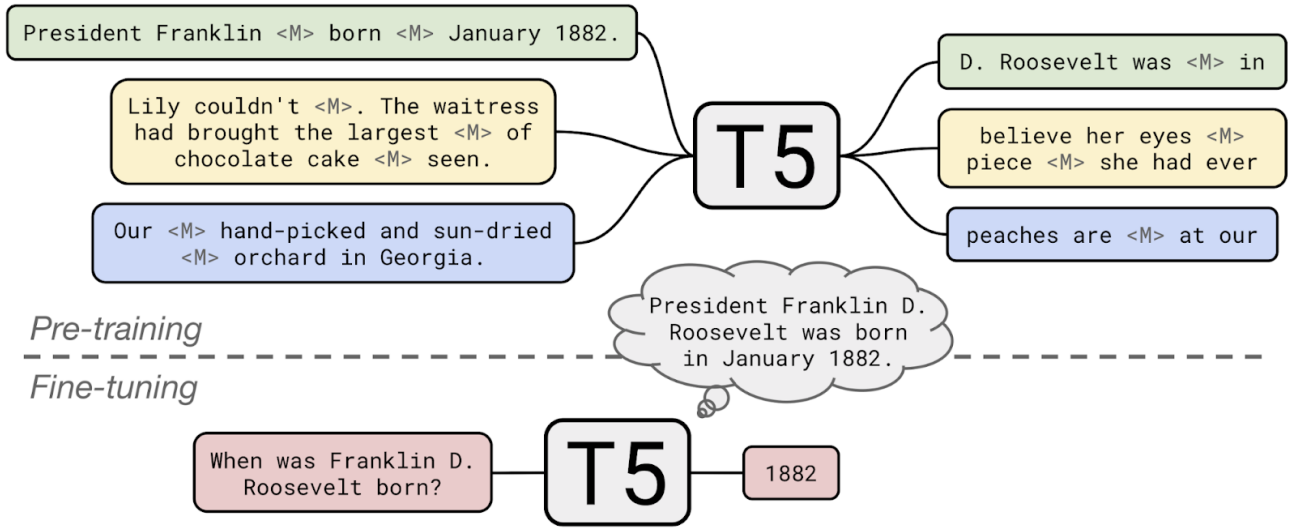


Content

- Self-Attention
- Transformer Architecture
- Different Pre-training Objectives
  - Decoder-Only Models (GPT)
  - Encoder-Only Models (BERT)
  - **Encoder-Decoder Models (T5, BART)**

T5 Model

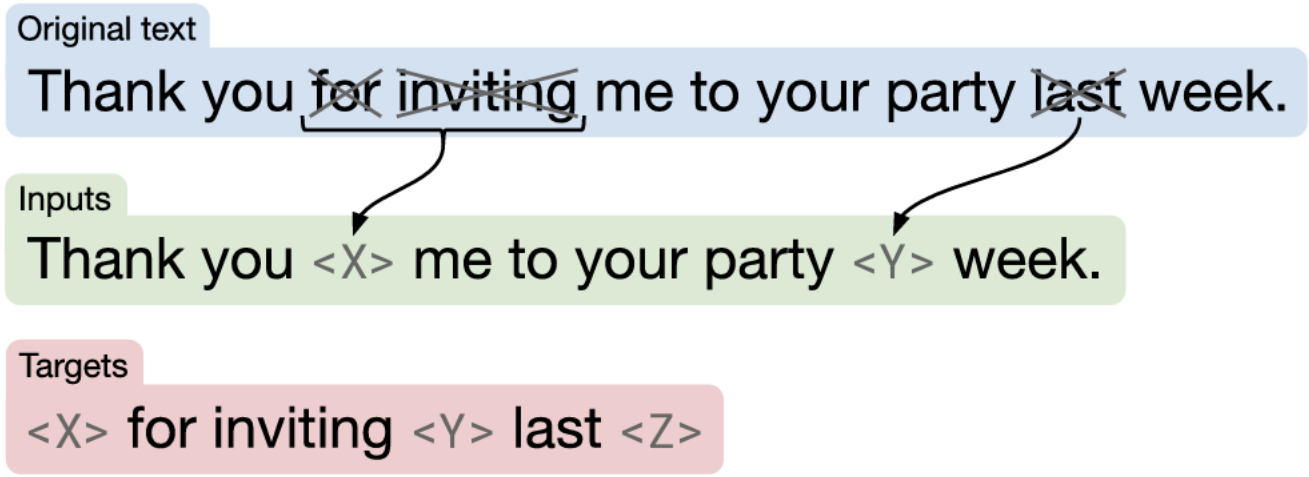
- How to predict a span of masked tokens within a sentence?
- BERT model requires the number of [MASK] token to be given in prior, while GPT models are causal left-to-right models
- T5: **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer (parameters: 60M~11B)



Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.

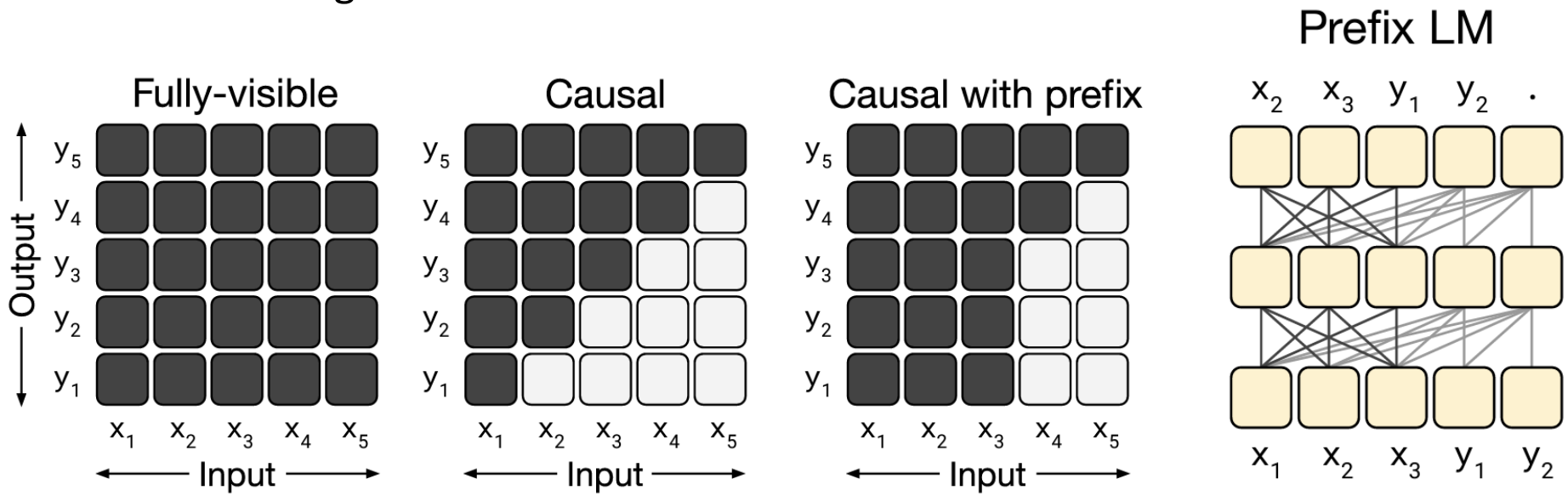
Training of T5 Model

- Pretraining: Mask out spans of texts; generate the original spans
- Fine-Tuning: Convert every task into a sequence-to-sequence generation problem
- Text-to-Text: Uncertain number of tokens in the input, and uncertain number of tokens in the output



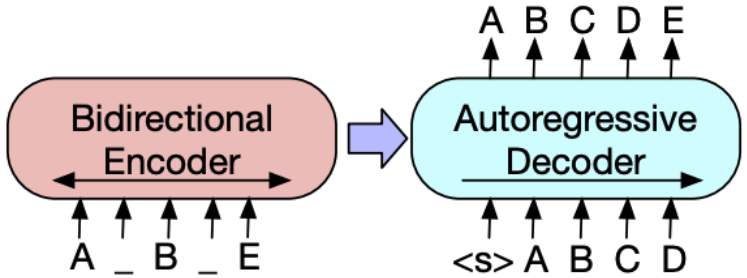
T5 Attention

- A “fully-visible” attention mechanism is placed at the input sequence.
  - Input Sequence:
    - translate English to German : That is good . target :
  - Target Output:
    - Das ist gut .

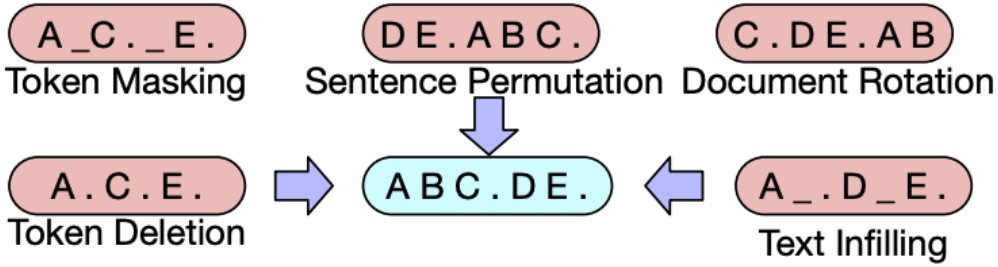


BART Model

- BART: Denoising autoencoder for pretraining sequence-to-sequence models
- Pretraining: Apply a series of noising schemes (e.g., masks, deletions, permutations…) to input sequences and train the model to recover the original sequences



BART architecture



BART pretraining objectives

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ACL.

Performance Comparison

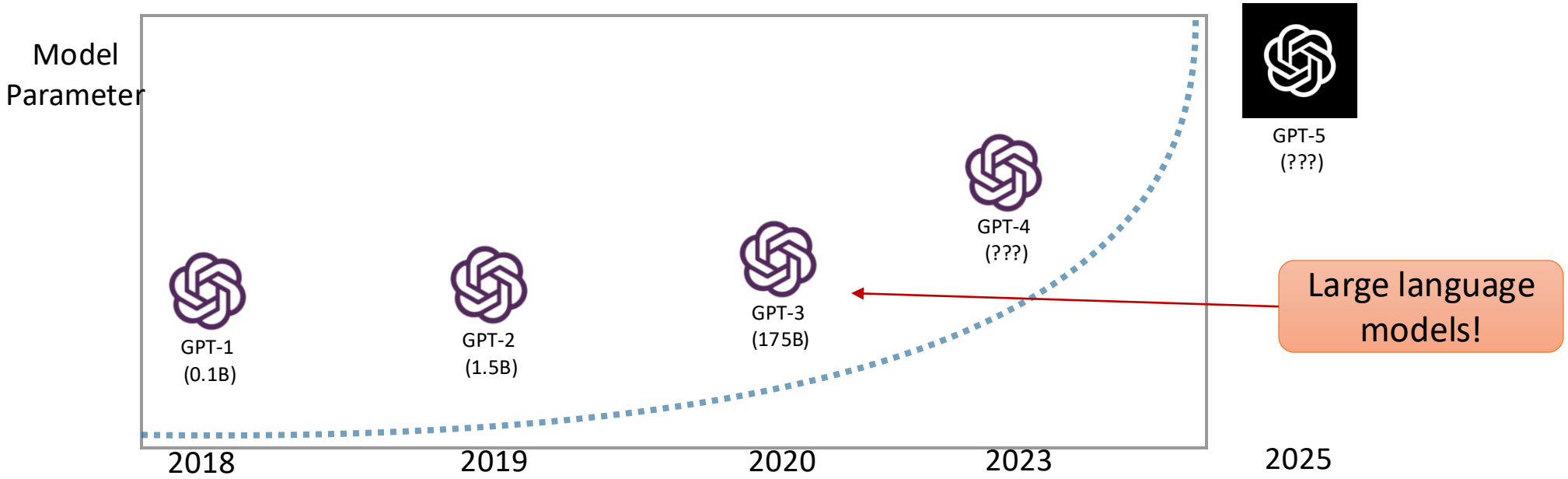
- Comparable to encoder models on language understanding tasks
- Better performance on language generation tasks

	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**

Next Class: Scaling up Language Models

- 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

Discussion Question

- There has been fewer successful attempts to scaling up BERT-based bidirectional models (e.g., 100x) than unidirectional models. Why does unidirectional model has better scaling performance than bidirectional models?