

# **Part I: Pretrained Language Models**

**KDD 2022 Tutorial**

**Adapting Pretrained Representations for Text Mining**

**Yu Meng, Jiaxin Huang, Yu Zhang, Jiawei Han**

**Computer Science, University of Illinois at Urbana-Champaign**

**Aug 14, 2022**

# Outline

---

- Introduction to text representations 
- Static word embeddings
- Deep contextualized embeddings via neural language models

# Overview of Text Representation Development

- Texts need to be represented as numbers/vectors so that computer programs can process them
- How were texts represented in history?

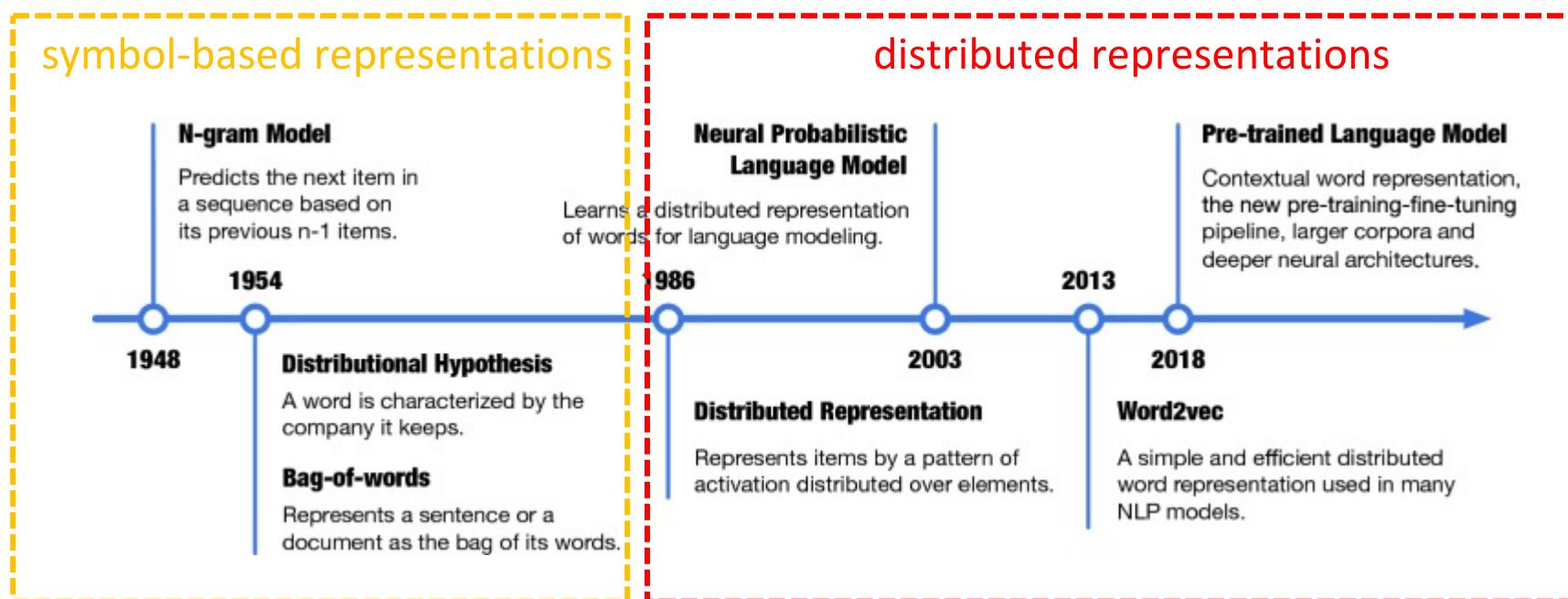


Figure from: Liu Z., Lin Y., Sun M. (2020) Representation Learning and NLP. In: Representation Learning for Natural Language Processing. Springer, Singapore.

# Symbol-Based Text Representations

---

- One-to-one correspondence between text units and representation elements
- e.g., “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]
- Bag-of-words representation of documents: Describe a document according to which words are present, ignoring word ordering
  - e.g., “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

---

- ❑ The Distributional Hypothesis: “a word is characterized by the company it keeps”
  - ❑ words that are 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
- ❑ We focus on distributed representations in this tutorial

# Outline

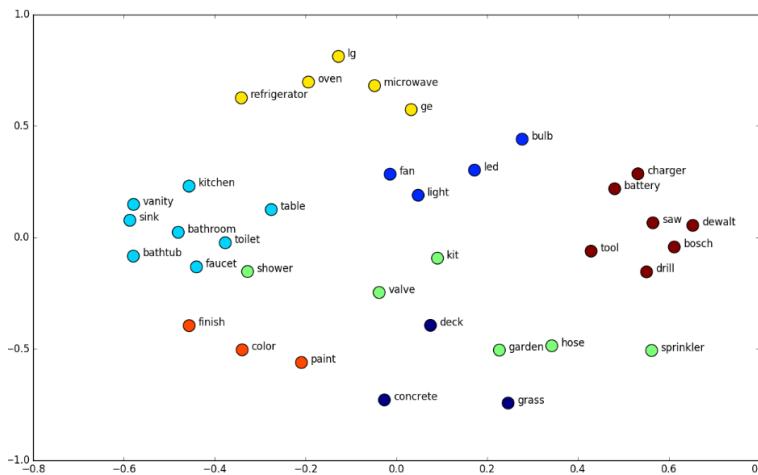
---

- Introduction to text representations
- Static word embeddings
- Deep contextualized embeddings via neural language models

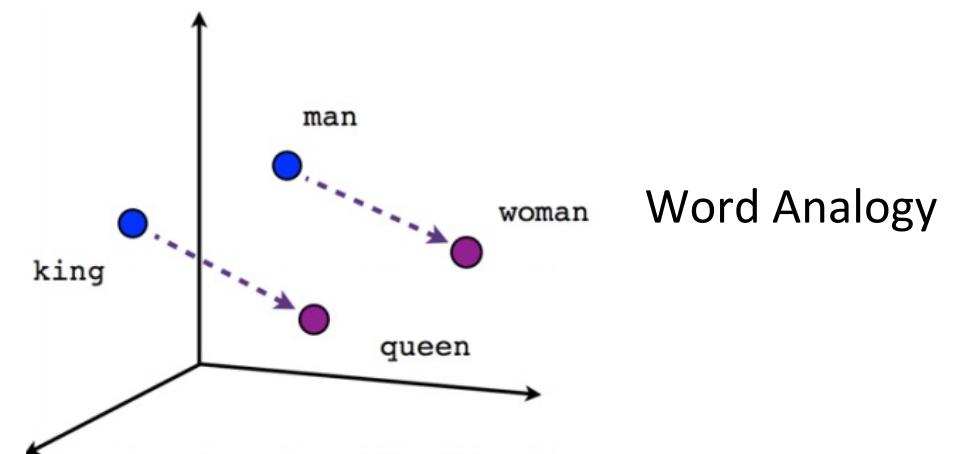


# Introduction to Text Embeddings

- ❑ Unsupervised/Self-supervised learning of text representations—No annotation needed
- ❑ Embed one-hot vectors into lower-dimensional space—Address “curse of dimensionality”
- ❑ Word embedding captures useful properties of word semantics
  - ❑ Word similarity: Words with similar meanings are embedded closer
  - ❑ Word analogy: Linear relationships between words (e.g., king - queen = man - woman)



Word Similarity



Word Analogy

# Outline

---

- Introduction to text representations
- Static word embeddings
  - Euclidean space: Word2Vec, GloVe
  - Hyperbolic space: Poincaré embeddings
  - Spherical space: JoSE
- Deep contextualized embeddings via neural language models



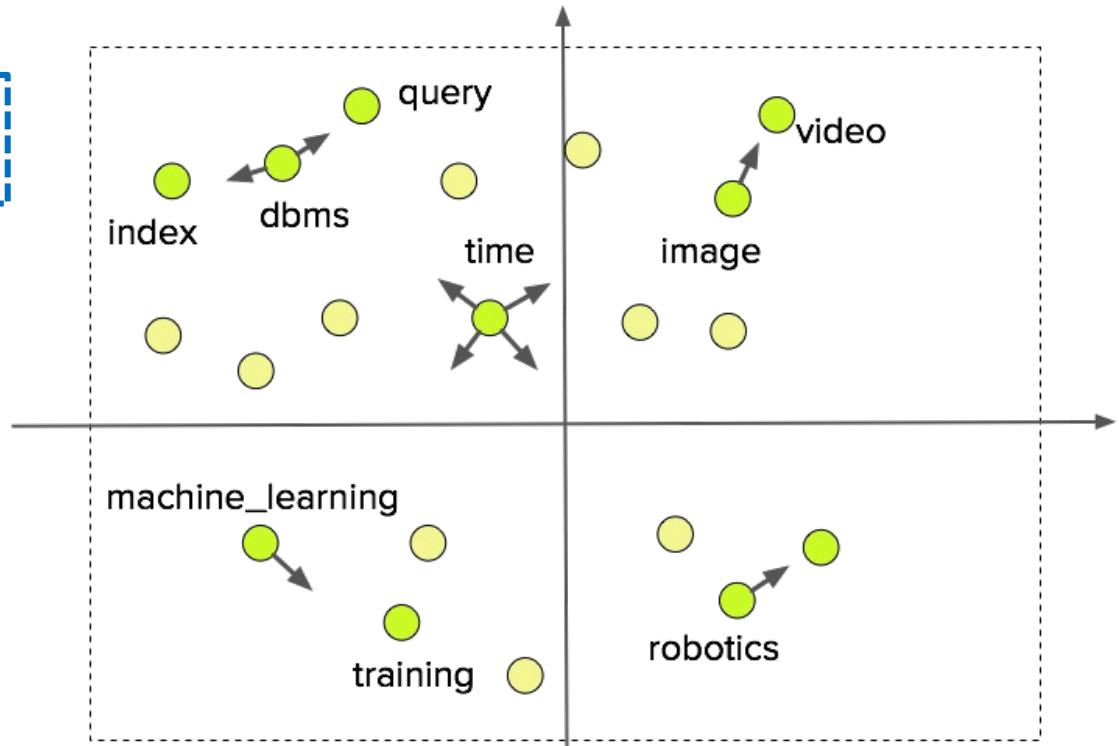
# Word2Vec

- Many text embeddings are learned in the Euclidean space (without constraints on vectors)
- Word2Vec maximizes the probability of observing a word based on its local contexts
- As a result, semantically similar terms are more likely to have close embeddings

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

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

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

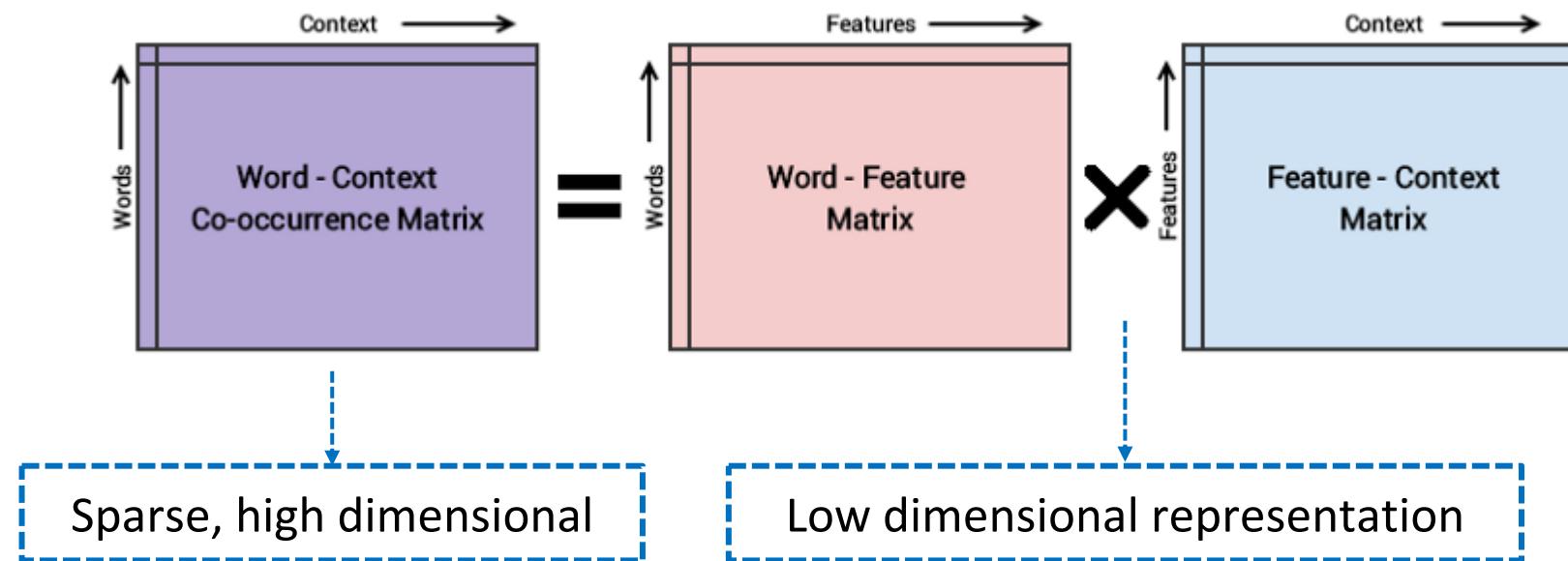


Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS.

# GloVe

- ❑ GloVe factorizes a global co-occurrence matrix derived from the entire corpus
- ❑ Low-dimensional representations are obtained by solving a least-squares problem to “recover” the co-occurrence matrix

$$J = \sum_{i,j=1}^V f(X_{ij}) (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$



# Outline

---

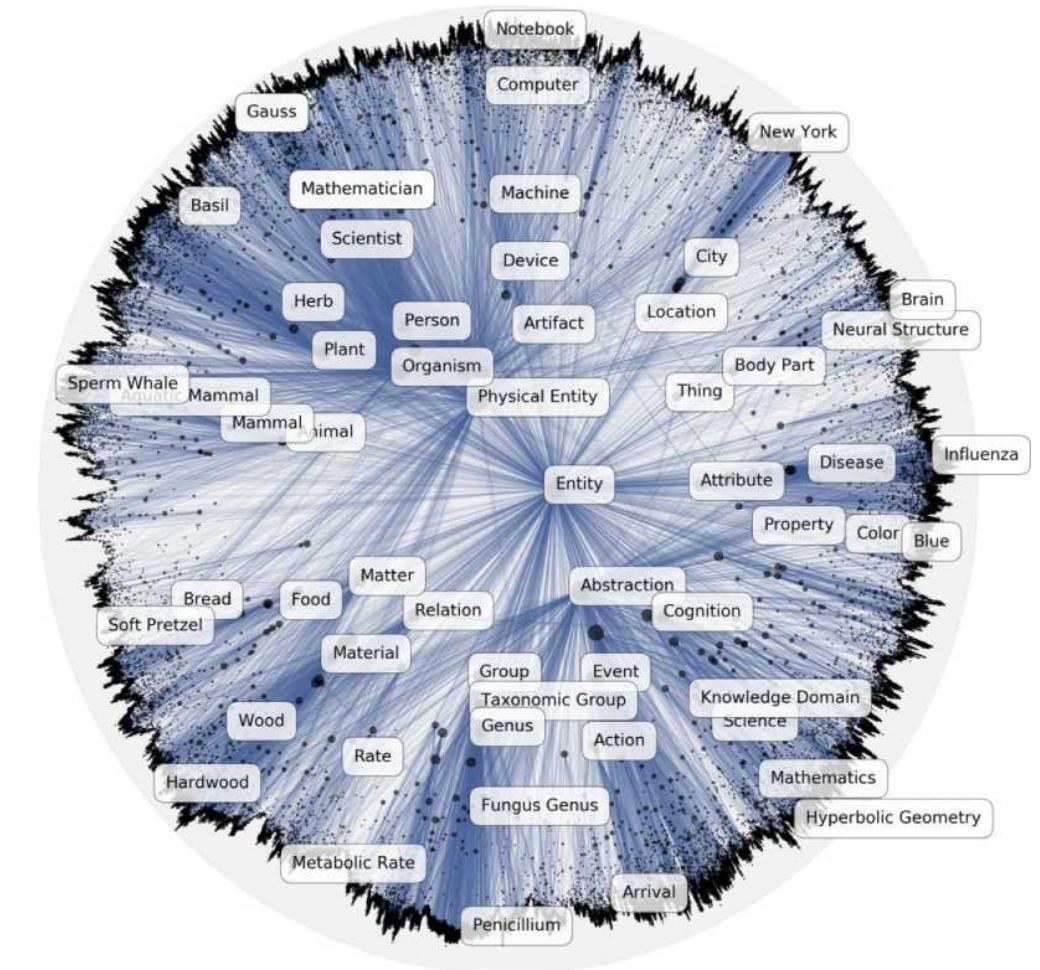
- Introduction to text representations
- Static word embeddings
  - Euclidean space: Word2Vec, GloVe
  - Hyperbolic space: Poincaré embeddings
  - Spherical space: JoSE
- Deep contextualized embeddings via neural language models



# Hyperbolic Embedding: Poincaré embedding

- Why non-Euclidean embedding space?
  - Data can have specific structures that Euclidean-space models struggle to capture
- The hyperbolic space
  - Continuous version of trees
  - Naturally equipped to model hierarchical structures
- Poincaré embedding
  - Learn hierarchical representations by pushing general terms to the origin of the Poincaré ball, and specific terms to the boundary

$$d(\mathbf{u}, \mathbf{v}) = \text{arcosh} \left( 1 + 2 \frac{\|\mathbf{u} - \mathbf{v}\|^2}{(1 - \|\mathbf{u}\|^2)(1 - \|\mathbf{v}\|^2)} \right)$$



Nickel, M., & Kiela, D. (2017). Poincaré Embeddings for Learning Hierarchical Representations. NIPS.

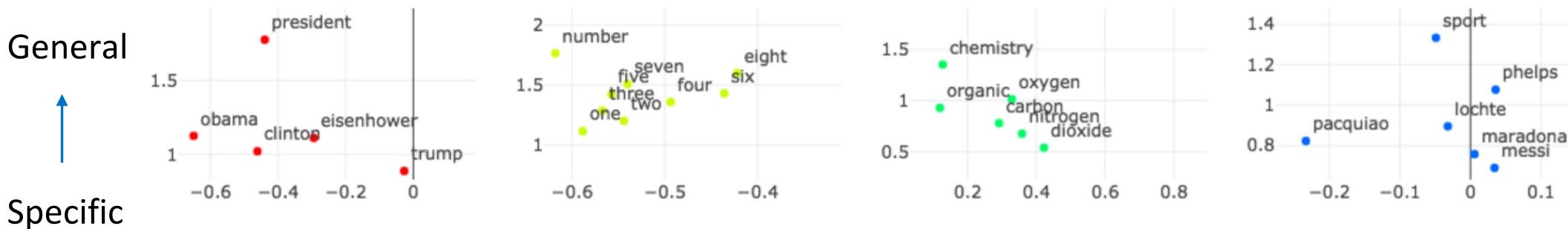
# Texts in Hyperbolic Space: Poincaré GloVe

- ❑ GloVe in hyperbolic space
- ❑ Motivation: latent hierarchical structure of words exists among text
  - ❑ Hypernym-hyponym
  - ❑ Textual entailment
- ❑ Approach: use hyperbolic kernels!
- ❑ Effectively model generality/specifity

$$J = \sum_{i,j=1}^V f(X_{ij}) \left( \boxed{w_i^T \tilde{w}_j} + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \quad \text{GloVe}$$

Hyperbolic metric

$$J = \sum_{i,j=1}^V f(X_{ij}) \left( \boxed{-h(d(w_i, \tilde{w}_j))} + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \quad \text{Poincaré GloVe}$$



Tifrea, A., Bécigneul, G., & Ganea, O. (2019). Poincaré GloVe: Hyperbolic Word Embeddings. ICLR.

# Outline

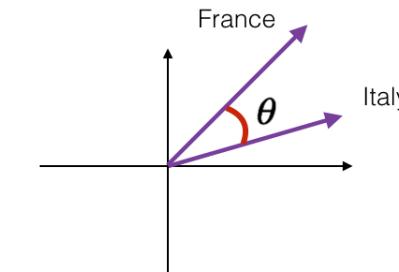
---

- Introduction to text representations
- Static word embeddings
  - Euclidean space: Word2Vec, GloVe
  - Hyperbolic space: Poincaré embeddings
  - Spherical space: JoSE
- Deep contextualized embeddings via neural language models



# Directional Analysis for Text Embeddings

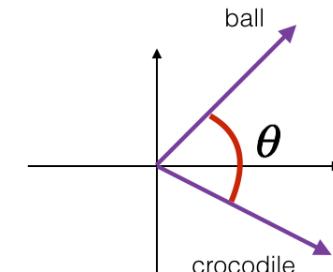
- How to use text embeddings? Mostly directional similarity (i.e., cosine similarity)
- Word similarity is derived using cosine similarity



France and Italy are quite similar

$\theta$  is close to  $0^\circ$

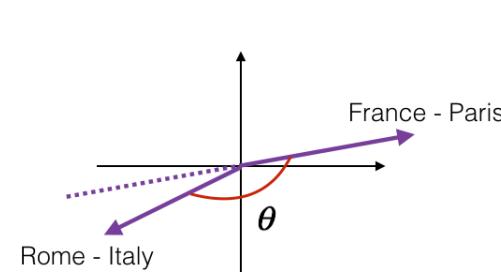
$\cos(\theta) \approx 1$



ball and crocodile are not similar

$\theta$  is close to  $90^\circ$

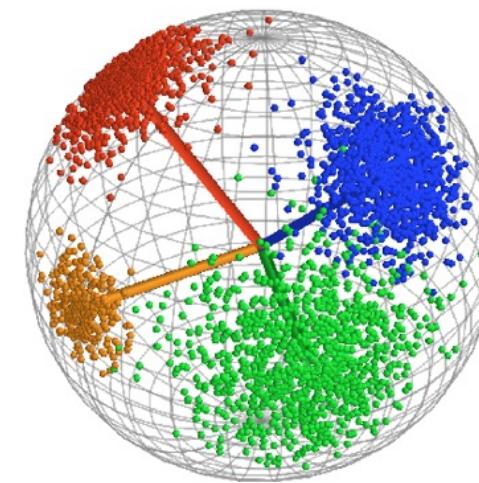
$\cos(\theta) \approx 0$



the two vectors are similar but opposite  
the first one encodes (city - country)  
while the second one encodes (country - city)

$\theta$  is close to  $180^\circ$

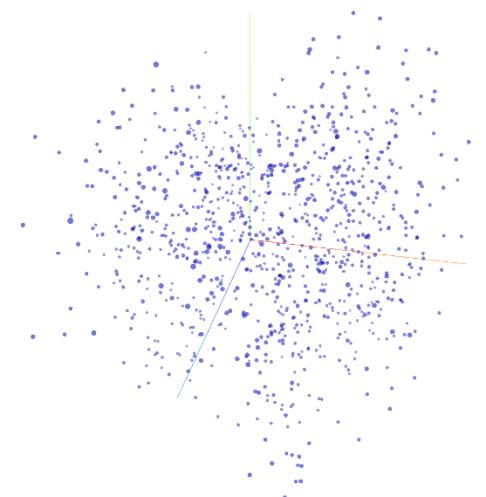
$\cos(\theta) \approx -1$



- Better clustering performances when embeddings are normalized, and spherical clustering algorithms are used (Spherical K-means)
- Vector direction is what actually matters!

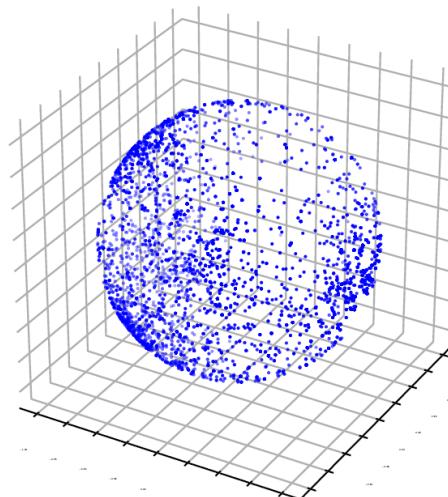
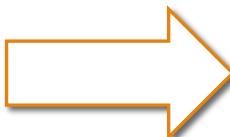
# Issues with Previous Embedding Frameworks

- Although directional similarity has shown effective for various applications, previous embeddings (e.g., Word2Vec, GloVe, fastText) are trained in the Euclidean space
- A gap between training space and usage space: Trained in Euclidean space but used on sphere



Embedding Training in Euclidean Space

Post-processing  
(Normalization)



Embedding Usage on the Sphere  
(Similarity, Clustering, etc.)

# Inconsistency Between Training and Usage

- The objective we optimize during training is not really the one we use
- Regardless of the different training objective, Word2Vec, GloVe and fastText all optimize the embedding **dot product** during training, but **cosine similarity** is what used in applications

Embedding dot product is optimized during training

The diagram illustrates the consistency between training and usage for three models. At the top, a blue arrow points from the text "Embedding dot product is optimized during training" down to three mathematical equations. Each equation has a dashed blue circle around a specific term, which is highlighted with a blue arrow pointing to it from the text above.

Word2Vec:  $p(w_O|w_I) = \frac{\exp(v'_{w_O}^\top v_{w_I})}{\sum_{w=1}^W \exp(v'_w^\top v_{w_I})}$

GloVe:  $J = \sum_{i,j=1}^V f(X_{ij})((w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2)$

fastText:  $s(w, c) = \sum_{g \in \mathcal{G}_w} (\mathbf{z}_g^\top \mathbf{v}_c)$

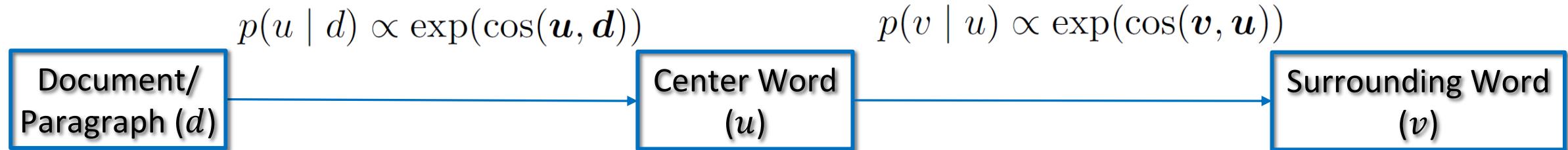
Word2Vec

GloVe

fastText

# Spherical Text Embedding: Generative Model

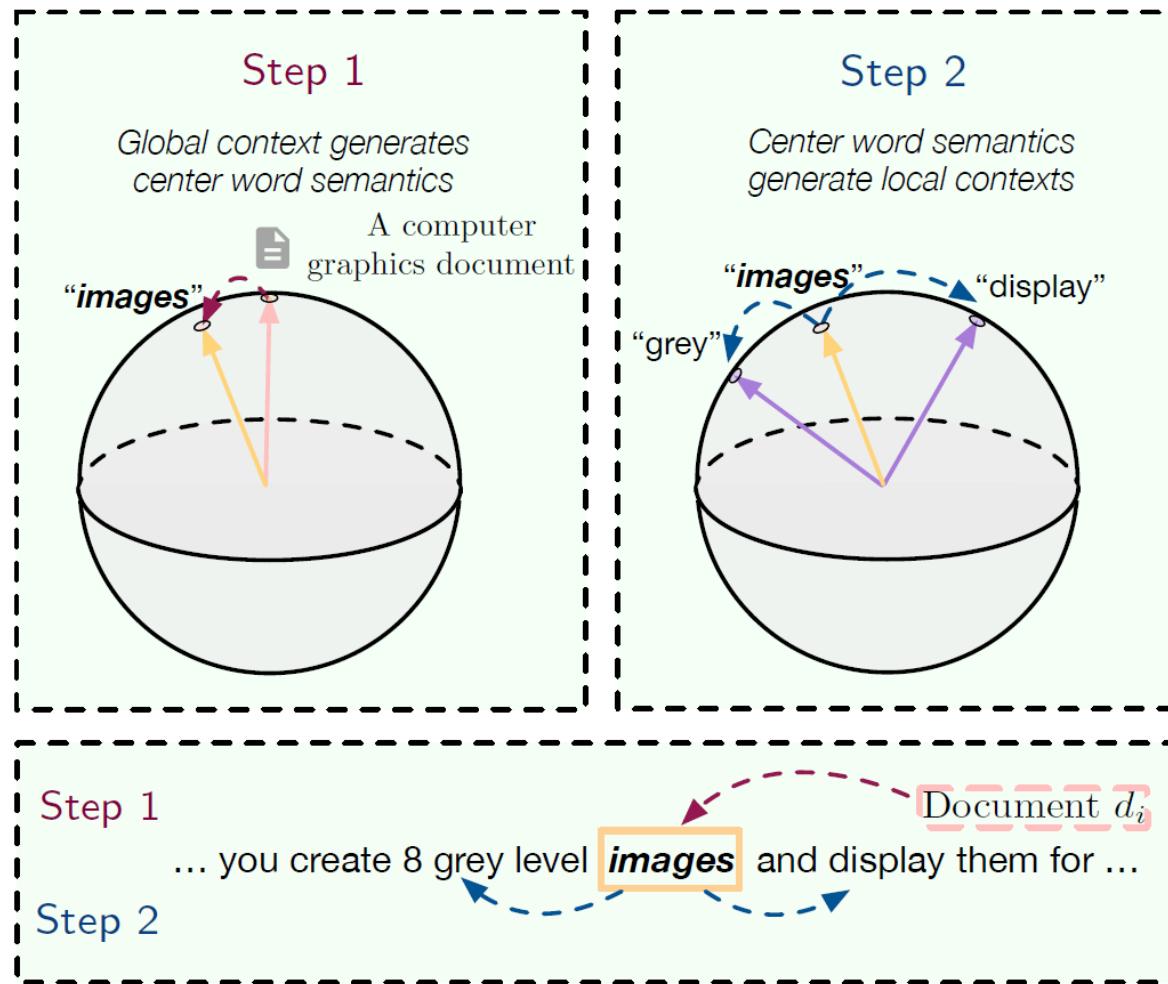
- We design a generative model on the sphere that follows how humans write articles:
  - We first have a general idea of the paragraph/document, and then start to write down each word in consistent with not only the paragraph/document, but also the surrounding words
  - Assume a two-step generation process:



Meng, Y., Huang, J., Wang, G., Zhang, C., Zhuang, H., Kaplan, L.M., & Han, J. (2019). Spherical Text Embedding. NeurIPS.

# Spherical Text Embedding: Illustration

## □ Understanding the spherical generative model



# Outline

---

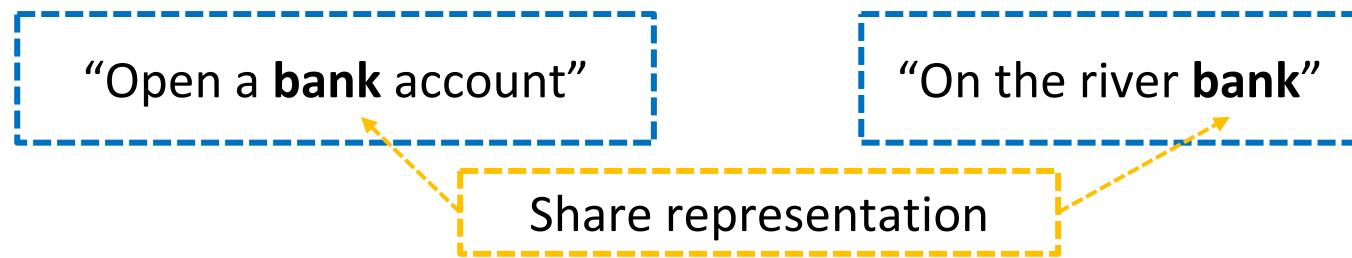
- Introduction to text representations
- Static word embeddings
- Deep contextualized embeddings via neural language models



# From Static Embedding to Contextualized Embedding

---

- ❑ Previous unsupervised word embeddings like Word2Vec and GloVe learn **static** word embedding
  - ❑ Each word has one representation regardless of specific contexts it appears in
  - ❑ E.g., “bank” is a polysemy, but only has one representation



- ❑ Deep neural language models overcome this problem by learning **contextualized** word semantics

# Pretrained Language Models: Overview

---

- The “pretrain-finetune” paradigm has become the prominent practice in a wide variety of text applications
- “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 using 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

# Outline

---

- Introduction to text representations
- Static word embeddings
- Deep contextualized embeddings via neural language models
  - Pretrained Language Models: Categorization by Architecture
  - Language Model Deployment



# Categorization of Pretrained Language Models

---

- ❑ There are multiple ways to categorize PLMs
  - ❑ By pretraining objectives: Standard language modeling, masked language modeling, permuted language modeling...
  - ❑ By pretraining settings: Multilingual, knowledge-enriched, domain-specific...
- ❑ In this presentation, we categorize PLMs **by architecture** which correlates with the task type PLMs are used for:
  - ❑ **Decoder-Only (Unidirectional) PLM:** Predict the next token based on previous tokens, usually used for **language generation tasks** (e.g., GPT)
  - ❑ **Encoder-Only (Bidirectional) PLM:** Predict masked/corrupted tokens based on all other (uncorrupted) tokens, usually used for **language understanding/classification tasks** (e.g., BERT, XLNet, ELECTRA)
  - ❑ **Encoder-Decoder (Sequence-to-Sequence) PLM:** Generate output sequences given masked/corrupted input sequences, can be used for both **language understanding and generation tasks** (e.g., T5, BART)

# Outline

---

- ❑ Introduction to text representations
- ❑ Static word embeddings
- ❑ Deep contextualized embeddings via neural language models
  - ❑ Pretrained Language Models: Categorization by Architecture
    - ❑ Decoder-Only (Unidirectional) PLM
    - ❑ Encoder-Only (Bidirectional) PLM
    - ❑ Encoder-Decoder (Sequence-to-Sequence) PLM
  - ❑ Language Model Deployment



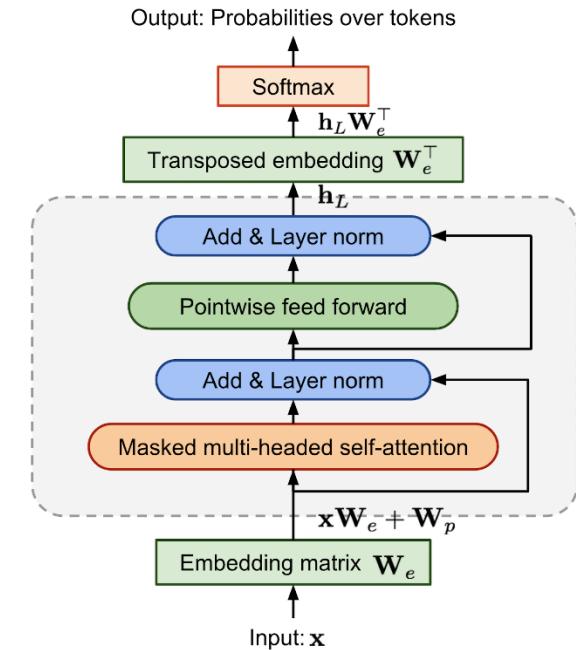
# GPT-Style Pretraining: Introduction

- Generative Pretraining (GPT [1], GPT-2 [2], GPT-3 [3]):
- Leverage unidirectional context (usually left-to-right) for next token prediction (i.e., language modeling)

*k* previous tokens as context

$$\mathcal{L}_{LM} = - \sum_i \log p(x_i | x_{i-k}, \dots, x_{i-1})$$

- The Transformer uses **unidirectional** attention masks (i.e., every token can only attend to previous tokens)



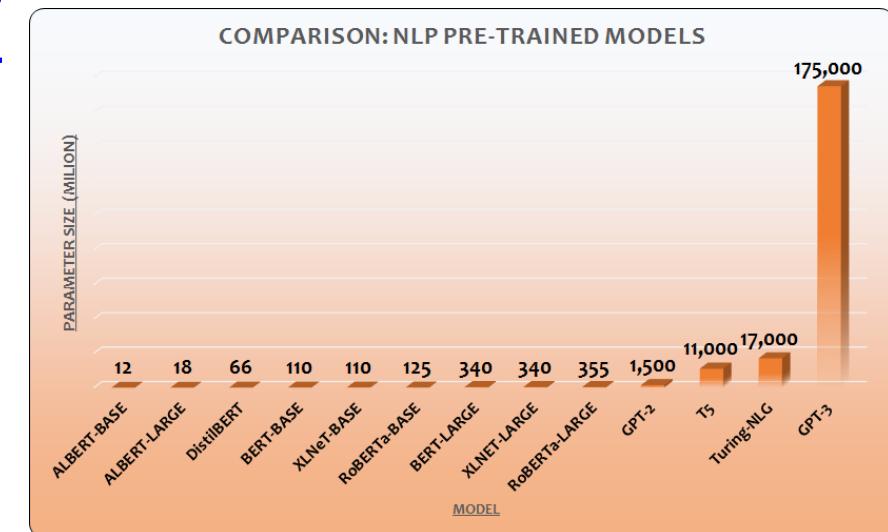
[1] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog

[2] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. NeurIPS.

# GPT-Style Pretraining: Text Generation

- ❑ Unidirectional LMs are commonly used for **text generation tasks** (e.g., summarization, translation, ...)
- ❑ They can be very, very large (GPT-3 has 175 billion parameters!) and have very strong text generation abilities (e.g., generated articles make human evaluators difficult to distinguish from articles written by humans)
- ❑ A demo of real articles vs. generated texts by GPT-2 trained on 10K Nature Papers: <https://stefanzukin.com/enigma/>



# Outline

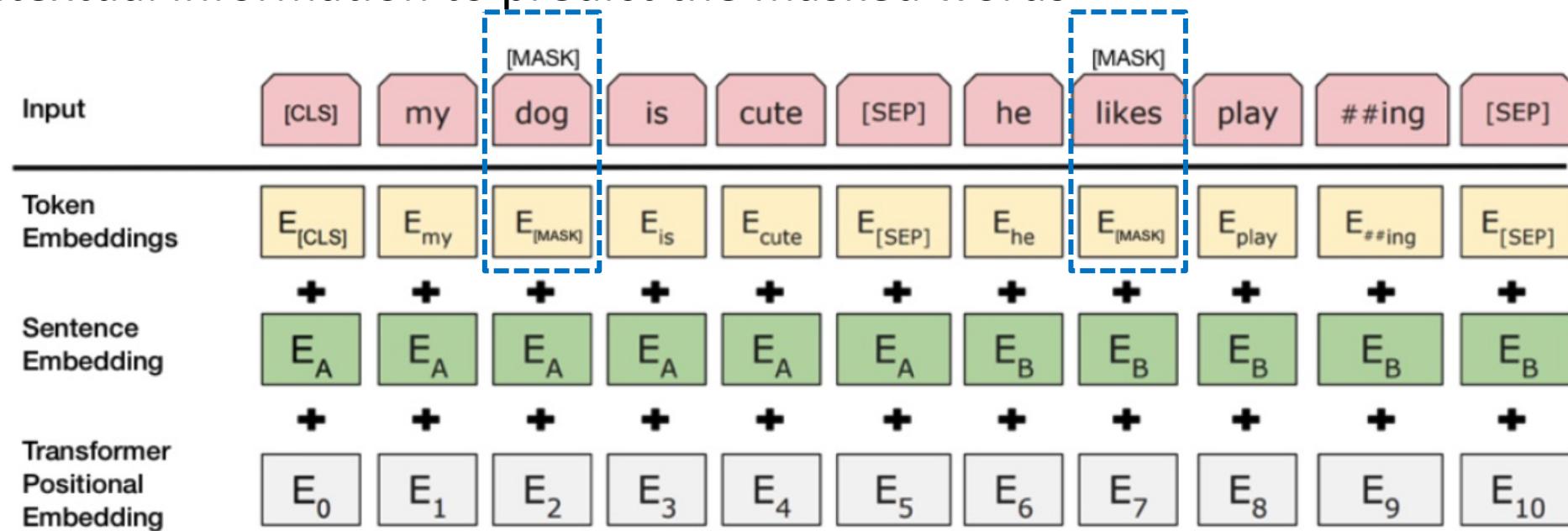
---

- ❑ Introduction to text representations
- ❑ Static word embeddings
- ❑ Deep contextualized embeddings via neural language models
  - ❑ Pretrained Language Models: Categorization by Architecture
    - ❑ Decoder-Only (Unidirectional) PLM
    - ❑ Encoder-Only (Bidirectional) PLM
    - ❑ Encoder-Decoder (Sequence-to-Sequence) PLM
  - ❑ Language Model Deployment



# BERT: Masked Language Modeling

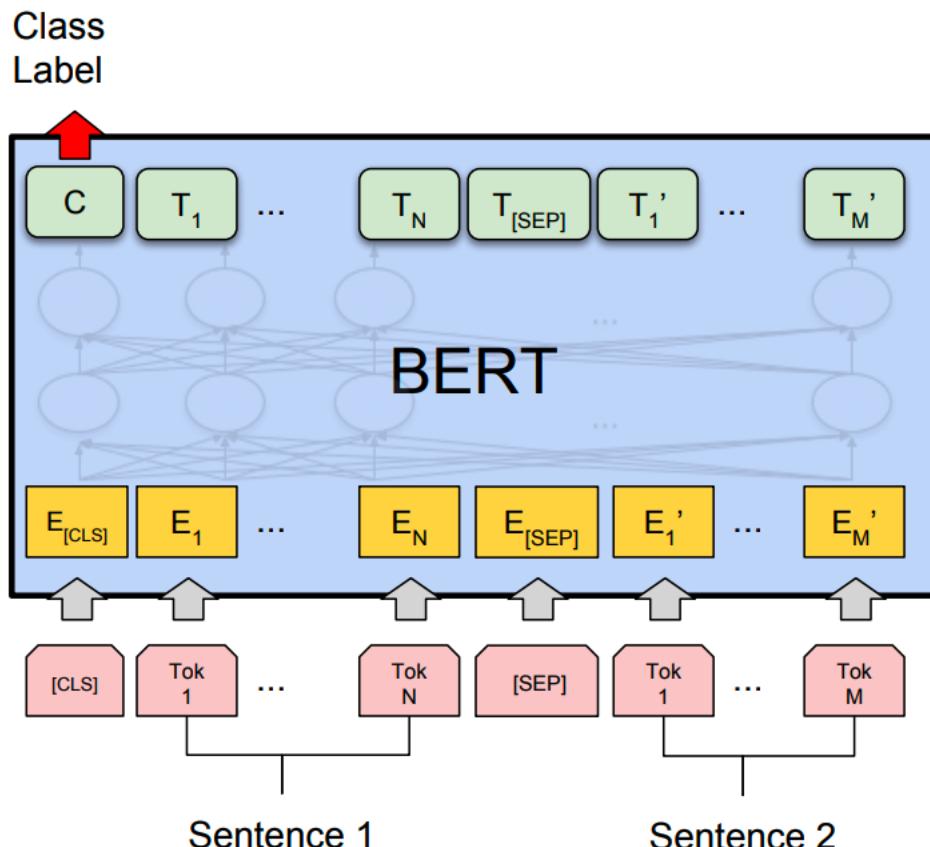
- Bidirectional: BERT leverages a Masked LM learning to introduce **real bidirectionality** training
- Masked LM: With 15% words randomly masked, the model learns bidirectional contextual information to predict the masked words



Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." NAACL (2019).

# BERT: Next Sentence Prediction

- Next Sentence Prediction: learn to predict if the second sentence in the pair is the subsequent sentence in the original document



# RoBERTa

- ❑ Several simple modifications that make BERT more **effective**:
  - ❑ train the model longer, with bigger batches over more data
  - ❑ remove the next sentence prediction objective
  - ❑ train on longer sequences
  - ❑ dynamically change the masking pattern applied to the training data

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	<b>94.6/89.4</b>	<b>90.2</b>	<b>96.4</b>
BERT <sub>LARGE</sub>						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
XLNet <sub>LARGE</sub>						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.

# ALBERT

---

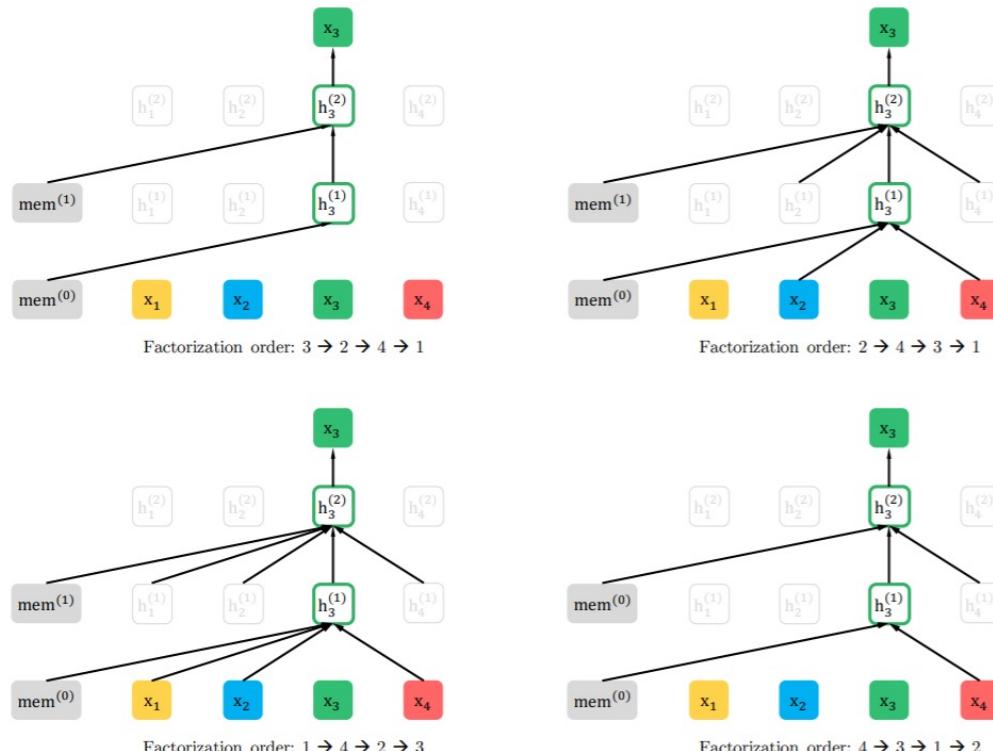
- Simple modifications that make BERT more **efficient**:
  - Factorized embedding parameterization: use lower-dimensional token embeddings; project token embeddings to hidden layer dimension
  - Cross-layer parameter sharing: Share feed-forward network parameters/attention parameters across layers
  - Inter-sentence coherence loss: change the next sentence prediction task to sentence order prediction

	Model	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	<b>94.1/88.3</b>	<b>88.1/85.1</b>	<b>88.0</b>	<b>95.2</b>	<b>82.3</b>	<b>88.7</b>	0.3x

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite BERT for self-supervised learning of language representations. ICLR.

# XLNet: Autoregressive Language Modeling

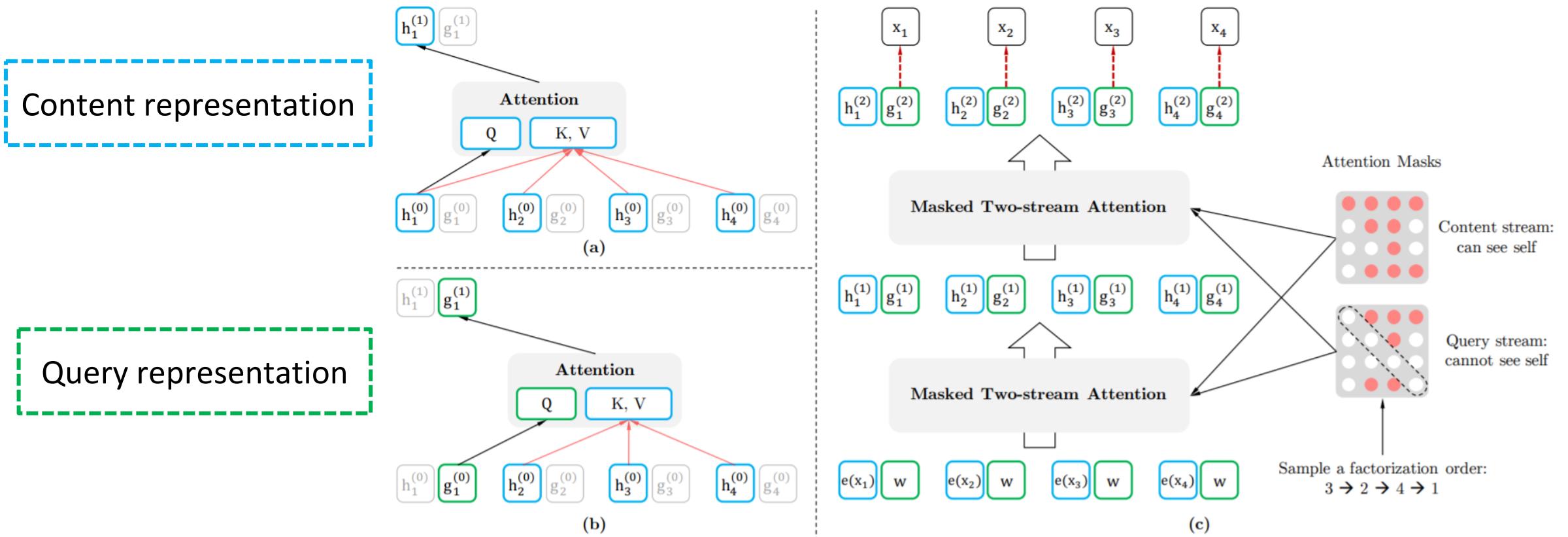
- ❑ Issues with BERT: Masked tokens are predicted independently, and [MASK] token brings discrepancy between pretraining and fine-tuning
- ❑ XLNet uses Permutation Language Modeling



- ❑ Permutes the text sequence and predicts the target word using the remaining words in the sequence
- ❑ Since words in the original sequence are permuted, both forward direction information and backward direction information are leveraged

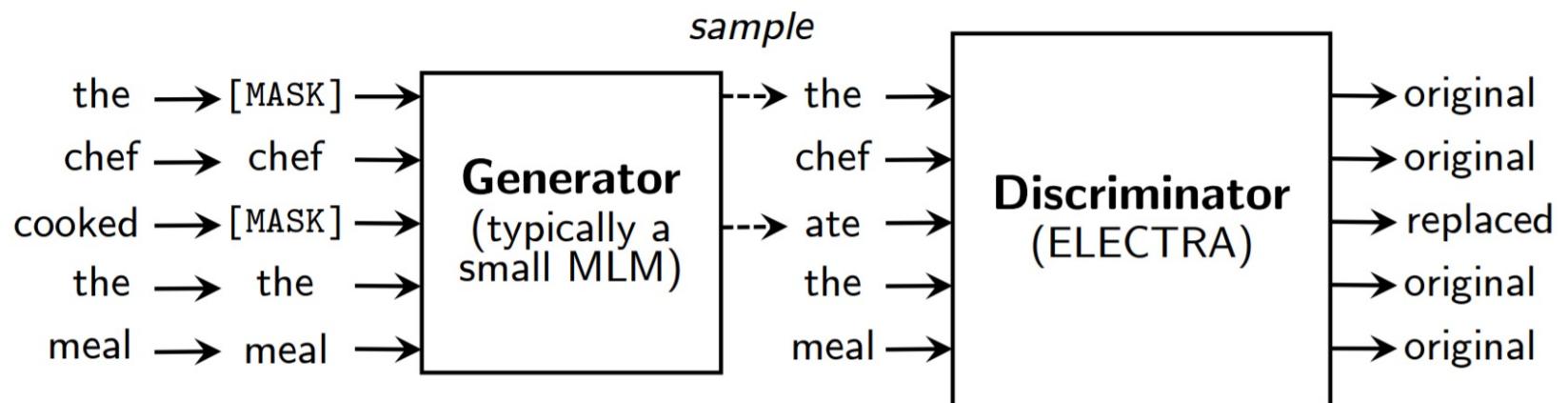
# XLNet: Two-Stream Self-Attention

- Content representation: Encodes both token position as well as content
- Query representation: Encodes only token position



# ELECTRA

- ❑ Change masked language modeling to a more sample-efficient pretraining task, **replaced token detection**
- ❑ Why more efficient:
  - ❑ Replaced token detection trains on all tokens, instead of just on those that are masked (15%)
  - ❑ The generator trained with MLM is small (parameter size is ~1/10 of discriminator)
  - ❑ The discriminator is trained with a binary classification task, instead of MLM (classification over the entire vocabulary)



Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. ICLR.

# ELECTRA

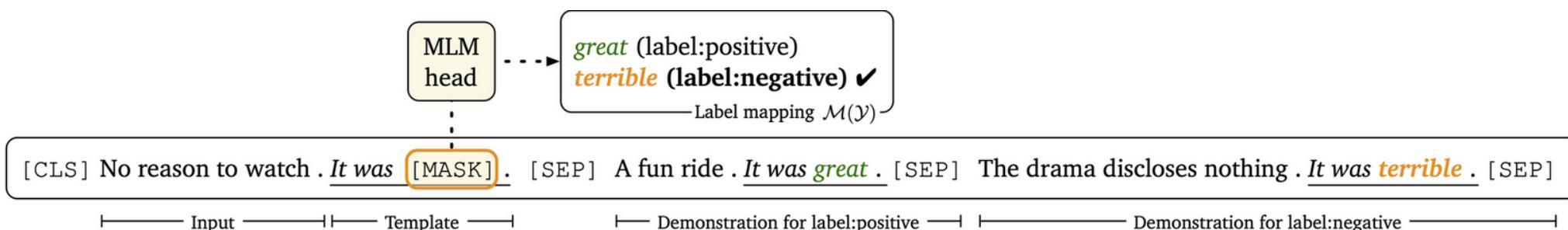
---

- Better GLUE (General Language Understanding Evaluation) test performance than previous MLM-based models under the same compute (measured by Floating Point Operations)

Model	Train FLOPs	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI	Avg.*	Score
BERT	1.9e20 (0.06x)	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	65.1	79.8	80.5
RoBERTa	3.2e21 (1.02x)	67.8	96.7	89.8	91.9	90.2	90.8	95.4	88.2	89.0	88.1	88.1
ALBERT	3.1e22 (10x)	69.1	<b>97.1</b>	<b>91.2</b>	92.0	90.5	<b>91.3</b>	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	<b>97.1</b>	90.5	<b>92.6</b>	90.4	90.9	–	88.5	<b>92.5</b>	89.1	–
ELECTRA	3.1e21 (1x)	<b>71.7</b>	<b>97.1</b>	90.7	92.5	<b>90.8</b>	<b>91.3</b>	<b>95.8</b>	<b>89.8</b>	<b>92.5</b>	<b>89.5</b>	<b>89.4</b>

# Challenges with ELECTRA-Style Pretraining

- ❑ What are the potential issues with ELECTRA-style pretraining?
- ❑ The main model (i.e., discriminator) in ELECTRA is trained via a binary classification task, which is simpler than language modeling tasks (usually over-30,000-way classification tasks), but raises two challenges:
  - ❑ Lack of the **language modeling capability** of the main model which is a necessity in some tasks (e.g., prompt-based fine-tuning)
  - ❑ The binary classification task may not be fine-grained enough to capture certain **word-level** semantics that are critical for token-level tasks

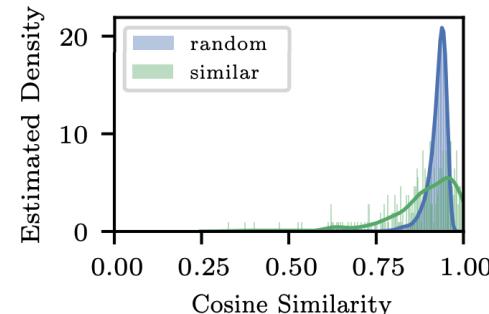


Prompt-based fine-tuning transfers the PLMs' language modeling ability to downstream tasks

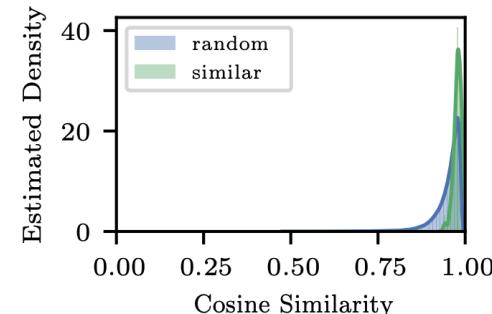
# Challenges with ELECTRA-Style Pretraining

- ❑ What are the potential issues with ELECTRA-style pretraining?
- ❑ Representations from Transformer-based language models often reside in a narrow cone in the embedding space, which raises the risk of degeneration and requires post-adjustment for meaningful sequence representations
  - ❑ Two random sentences have high similarity scores (lack of **uniformity**)
  - ❑ Two closely related sentences may have more different representations (lack of **alignment**)
- ❑ Plots: Distribution of cosine similarities between sequence pairs using their [CLS] embeddings from pretrained models
  - ❑ random: random sentence pairs from pretraining corpus
  - ❑ similar: semantically similar pairs annotated with maximum similarity from STS-B

RoBERTa sequence embedding space:

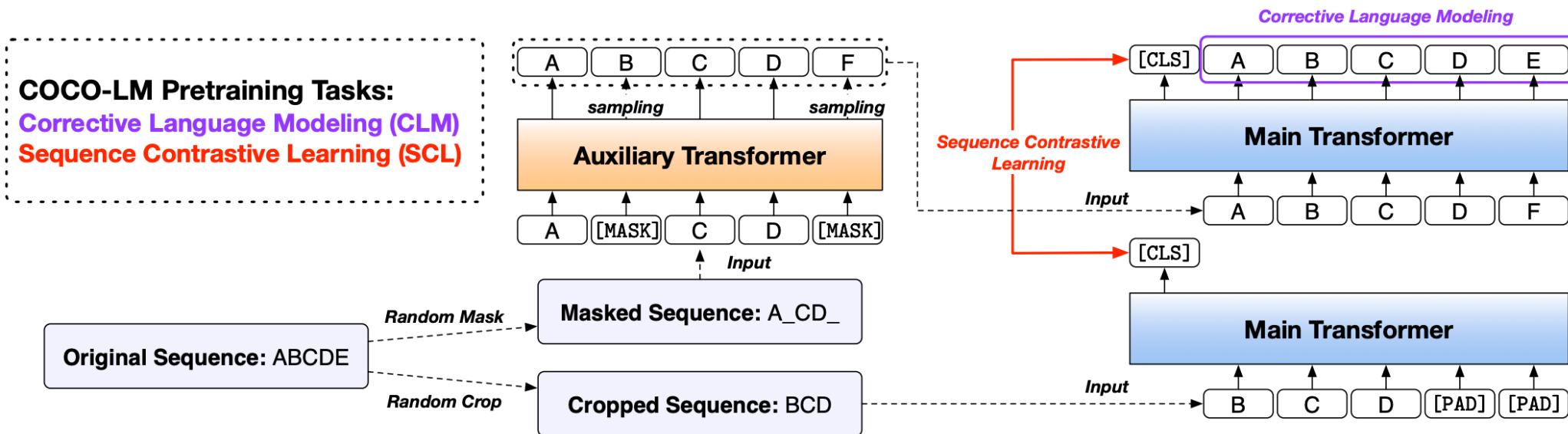


ELECTRA sequence embedding space:



# COCO-LM: Method

- COCO-LM has two new pretraining tasks upon the corrupted sequences that address the challenges in ELECTRA-style pretraining
  - Corrective Language Modeling (CLM)
  - Sequence Contrastive Learning (SCL)



Meng, Y., Xiong, C., Bajaj, P., Bennett, P., Han, J., & Song, X. (2021). COCO-LM: Correcting and contrasting text sequences for language model pretraining. NeurIPS.

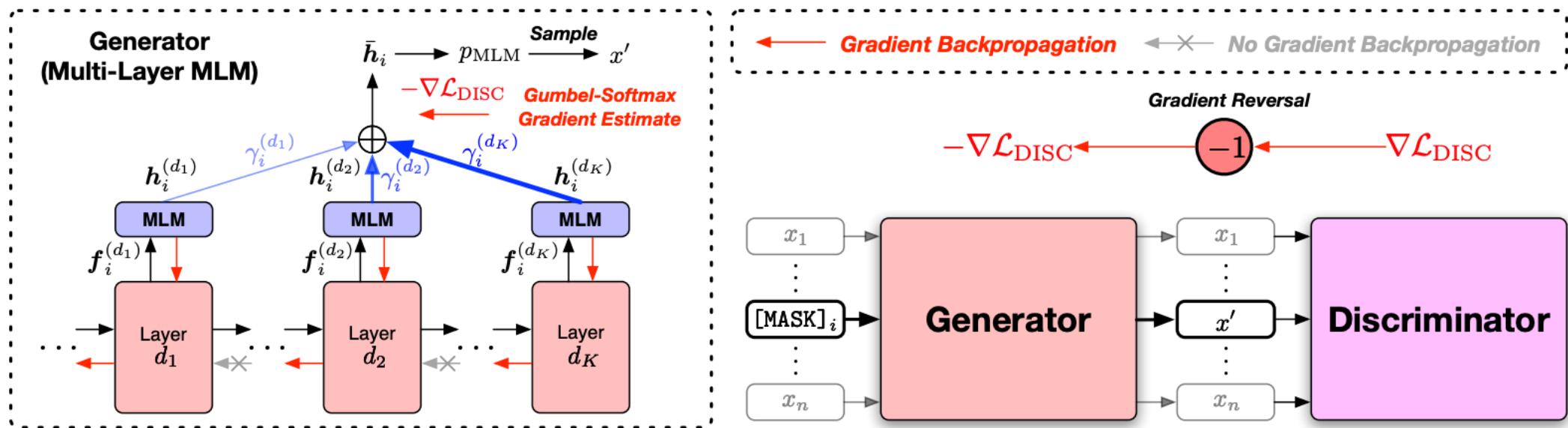
# COCO-LM: Results

- Outperforming previous PLMs on GLUE and SQuAD 2.0 dev sets
- One of the state-of-the-art PLMs for NLU tasks ([Blog Post by Microsoft](#))

Model	Params	GLUE DEV Single Task								SQuAD 2.0 DEV		
		MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	AVG	EM	F1
<b>Base Setting:</b> BERT Base Size, Wikipedia + Book Corpus (16GB)												
BERT [11]	110M	84.5/-	91.3	91.7	93.2	58.9	68.6	87.3	89.5	83.1	73.7	76.3
RoBERTa [31]	125M	84.7/-	—	—	92.7	—	—	—	—	—	—	79.7
XLNet [62]	110M	85.8/85.4	—	—	92.7	—	—	—	—	—	78.5	81.3
ELECTRA [7]	110M	86.0/85.3	90.0	91.9	93.4	64.3	70.8	84.9	89.1	83.7	80.5	83.3
MC-BERT [61]	110M	85.7/85.2	89.7	91.3	92.3	62.1	75.0	86.0	88.0	83.7	—	—
DeBERTa [24]	134M	86.3/86.2	—	—	—	—	—	—	—	—	79.3	82.5
TUPE [27]	110M	86.2/86.2	91.3	92.2	93.3	63.6	73.6	89.9	89.2	84.9	—	—
RoBERTa (Ours)	110M	85.8/85.5	91.3	92.0	<b>93.7</b>	60.1	68.2	87.3	88.5	83.3	77.7	80.5
ELECTRA (Ours)	110M	86.9/86.7	91.9	92.6	93.6	<b>66.2</b>	75.1	88.2	89.7	85.5	79.7	82.6
<b>COCO-LM</b>	110M	<b>88.5/88.3</b>	<b>92.0</b>	<b>93.1</b>	93.2	63.9	<b>84.8</b>	<b>91.4</b>	<b>90.3</b>	<b>87.2</b>	<b>82.4</b>	<b>85.2</b>
<b>Base++ Setting:</b> BERT Base Size, Bigger Training Data, and/or More Training Steps												
XLNet [62]	110M	86.8/-	91.4	91.7	94.7	60.2	74.0	88.2	89.5	84.6	80.2	—
RoBERTa [31]	125M	87.6/-	91.9	92.8	94.8	63.6	78.7	90.2	91.2	86.4	80.5	83.7
UniLM V2 [1]	110M	88.5/-	91.7	93.5	<b>95.1</b>	65.2	81.3	<b>91.8</b>	91.0	87.1	83.3	86.1
DeBERTa [24]	134M	88.8/88.5	—	—	—	—	—	—	—	—	83.1	86.2
CLEAR [59]	110M	86.7/-	90.0	92.9	94.5	64.3	78.3	89.2	89.8	85.7	—	—
<b>COCO-LM</b>	134M	<b>90.2/90.0</b>	<b>92.2</b>	<b>94.2</b>	94.6	<b>67.3</b>	<b>87.4</b>	91.2	<b>91.8</b>	<b>88.6</b>	<b>85.4</b>	<b>88.1</b>
<b>Large++ Setting:</b> BERT Large Size, Bigger Training Data, and More Training Steps												
XLNet [62]	360M	90.8/90.8	92.3	94.9	<b>97.0</b>	69.0	85.9	90.8	92.5	89.2	87.9	90.6
RoBERTa [31]	356M	90.2/90.2	92.2	94.7	96.4	68.0	86.6	90.9	92.4	88.9	86.5	89.4
ELECTRA [7]	335M	90.9/-	92.4	95.0	96.9	69.1	88.0	90.8	92.6	89.4	88.0	90.6
DeBERTa [24]	384M	91.1/91.1	92.3	95.3	96.8	70.5	—	—	—	—	88.0	90.7
<b>COCO-LM</b>	367M	<b>91.4/91.6</b>	<b>92.8</b>	<b>95.7</b>	96.9	<b>73.9</b>	<b>91.0</b>	<b>92.2</b>	<b>92.7</b>	<b>90.8</b>	<b>88.2</b>	<b>91.0</b>
Megatron <sub>1.3B</sub> [48]	1.3B	90.9/91.0	92.6	—	—	—	—	—	—	—	87.1	90.2
Megatron <sub>3.9B</sub> [48]	3.9B	91.4/91.4	92.7	—	—	—	—	—	—	—	88.5	91.2

# AMOS: Adversarial Curriculum for Pretraining

- ❑ Use a multi-layer MLM generator to create training signals (i.e., replaced tokens) of different levels of difficulty
- ❑ Automatically learn a mixture of the multi-layer MLM generator's outputs to construct the most difficult signals for the discriminator learning for better sample efficiency



Meng, Y., Xiong, C., Bajaj, P., Bennett, P. N., Han, J., & Song, X. (2022). Pretraining Text Encoders with Adversarial Mixture of Training Signal Generators. ICLR.

# AMOS: Results

---

## □ Further improvements over COCO-LM on GLUE

Model	Params	GLUE DEV Single Task							SQuAD 2.0			
		MNLI	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	Avg	EM	F1
<b>Base Setting:</b> BERT Base Size, Wikipedia + Book Corpus (16GB)												
BERT (Devlin et al., 2019)	110M	84.5/-	91.3	91.7	93.2	58.9	68.6	87.3	89.5	83.1	73.7	76.3
RoBERTa (Liu et al., 2019)	110M	85.8/85.5	91.3	92.0	93.7	60.1	68.2	87.3	88.5	83.3	77.7	80.5
XLNet (Yang et al., 2019)	110M	85.8/85.4	-	-	92.7	-	-	-	-	-	78.5	81.3
DeBERTa (He et al., 2021)	134M	86.3/86.2	-	-	-	-	-	-	-	-	79.3	82.5
TUPE (Ke et al., 2020)	110M	86.2/86.2	91.3	92.2	93.3	63.6	73.6	89.9	89.2	84.9	-	-
ELECTRA (Clark et al., 2020)	110M	86.9/86.7	91.9	92.6	93.6	66.2	75.1	88.2	89.7	85.5	79.7	82.6
+HP <sub>Loss</sub> +Focal (Hao et al., 2021)	110M	87.0/86.9	92.7	91.7	92.6	66.7	90.7	81.3	91.0	86.7	83.0	85.6
MC-BERT (Xu et al., 2020)	110M	85.7/85.2	89.7	91.3	92.3	62.1	75.0	86.0	88.0	83.7	-	-
COCO-LM (Meng et al., 2021)	110M	88.5/88.3	92.0	93.1	93.2	63.9	84.8	<b>91.4</b>	90.3	87.2	82.4	85.2
<b>AMOS</b>	<b>110M</b>	<b>88.9/88.7</b>	<b>92.3</b>	<b>93.6</b>	<b>94.2</b>	<b>70.7</b>	<b>86.6</b>	90.9	<b>91.6</b>	<b>88.6</b>	<b>84.2</b>	<b>87.2</b>
<b>Base++ Setting:</b> BERT Base Size, Bigger Training Data, and/or More Training Steps												
XLNet (Yang et al., 2019)	110M	86.8/-	91.4	91.7	94.7	60.2	74.0	88.2	89.5	84.6	80.2	-
RoBERTa (Liu et al., 2019)	125M	87.6/-	91.9	92.8	94.8	63.6	78.7	90.2	91.2	86.4	80.5	83.7
UniLM V2 (Bao et al., 2020)	110M	88.5/-	91.7	93.5	95.1	65.2	81.3	<b>91.8</b>	91.0	87.1	83.3	86.1
DeBERTa (He et al., 2021)	134M	88.8/88.5	-	-	-	-	-	-	-	-	83.1	86.2
CLEAR Wu et al. (2020)	110M	86.7/-	90.0	92.9	94.5	64.3	78.3	89.2	89.8	85.7	-	-
COCO-LM (Meng et al., 2021)	134M	90.2/90.0	92.2	94.2	94.6	67.3	<b>87.4</b>	91.2	91.8	88.6	<b>85.4</b>	<b>88.1</b>
<b>AMOS</b>	<b>134M</b>	<b>90.5/90.4</b>	<b>92.4</b>	<b>94.4</b>	<b>95.5</b>	<b>71.8</b>	86.6	91.7	<b>92.0</b>	<b>89.4</b>	85.0	87.9

# Outline

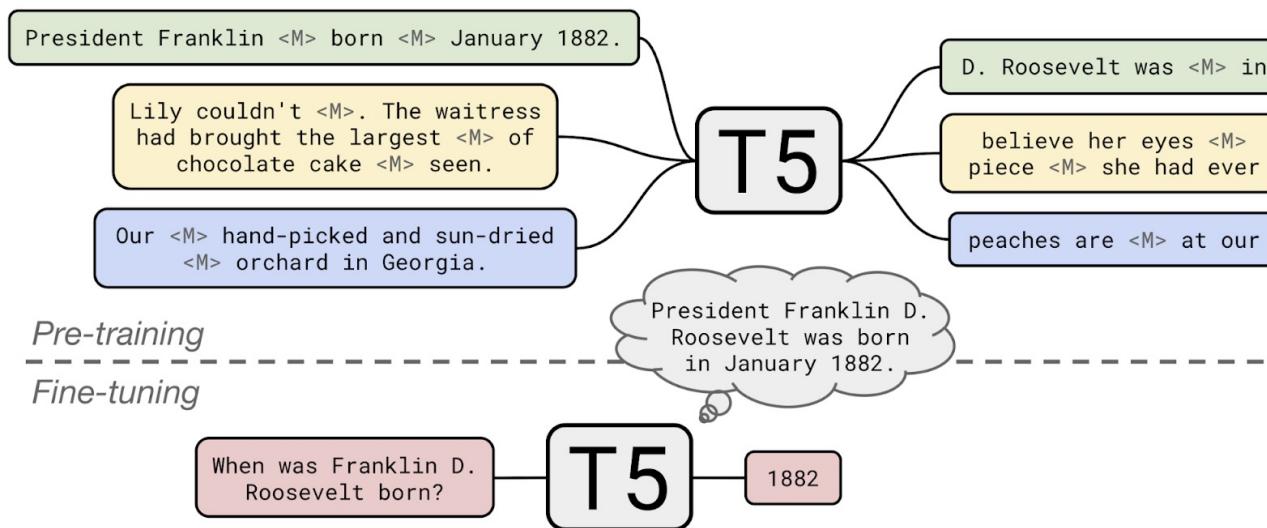
---

- ❑ Introduction to text representations
- ❑ Static word embeddings
- ❑ Deep contextualized embeddings via neural language models
  - ❑ Pretrained Language Models: Categorization by Architecture
    - ❑ Decoder-Only (Unidirectional) PLM
    - ❑ Encoder-Only (Bidirectional) PLM
    - ❑ Encoder-Decoder (Sequence-to-Sequence) PLM
  - ❑ Language Model Deployment



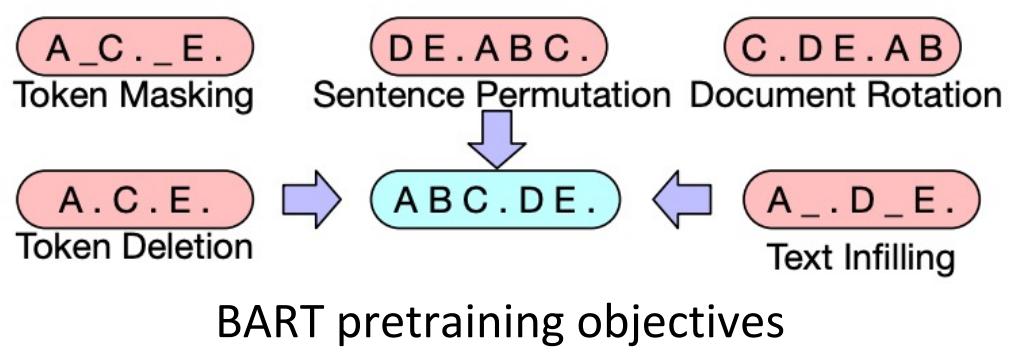
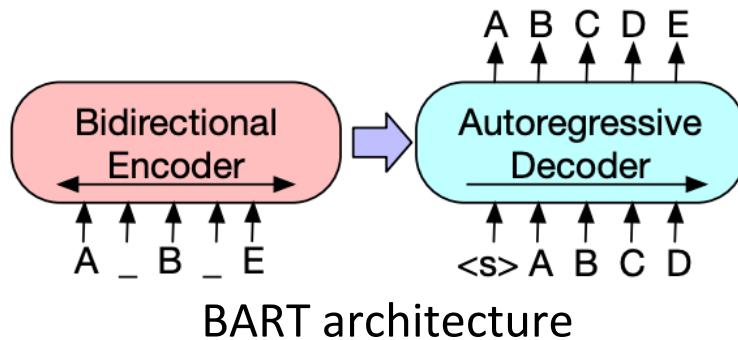
# T5

- ❑ T5: Text-to-Text Transfer Transformer
- ❑ Pretraining: Mask out spans of texts; generate the original spans
- ❑ Fine-Tuning: Convert every task into a sequence-to-sequence generation problem



# BART

- 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
- Fine-Tuning:
  - For classification tasks: Feed the same input into the encoder and decoder, and use the final decoder token for classification
  - For generation tasks: The encoder takes the input sequence, and the decoder generates outputs autoregressively



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.

# Outline

---

- Introduction to text representations
- Static word embeddings
- Deep contextualized embeddings via neural language models
  - Language Model Pretraining
  - Language Model Deployment



# Deployment of Pretrained Language Models

---

- ❑ Pretrained language models (PLMs) are usually trained on large-scale general domain corpora to learn generic linguistic features that can be transferred to downstream tasks
- ❑ Common usages of PLMs in downstream tasks
  - ❑ Fine-tuning: Update all parameters in the PLM encoder and task-specific layers (linear layer for standard fine-tuning or MLM layer for prompt-based fine-tuning) to fit downstream data
  - ❑ Prompt-based methods: Convert tasks to cloze-type token prediction problems; can be used for either fine-tuning or zero-shot inference
  - ❑ Parameter-efficient tuning: Only update a small portion of PLM parameters and keep other (majority) parameters unchanged

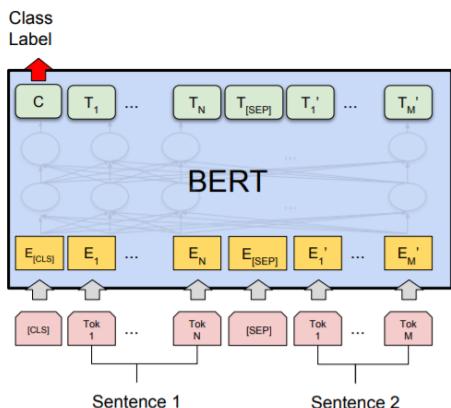
# Outline

---

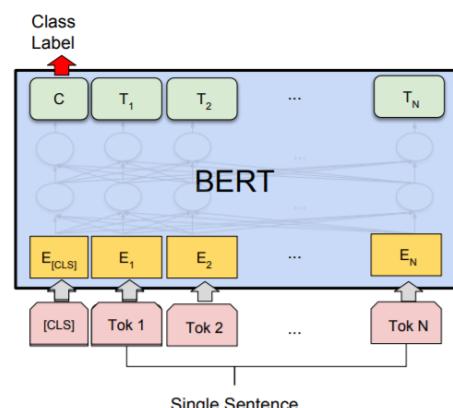
- ❑ Introduction to text representations
- ❑ Static word embeddings
- ❑ Deep contextualized embeddings via neural language models
  - ❑ Language Model Pretraining
  - ❑ Language Model Deployment
    - ❑ Standard fine-tuning 
    - ❑ Prompt-based methods
    - ❑ Parameter-efficient tuning

# Standard Fine-Tuning of PLMs

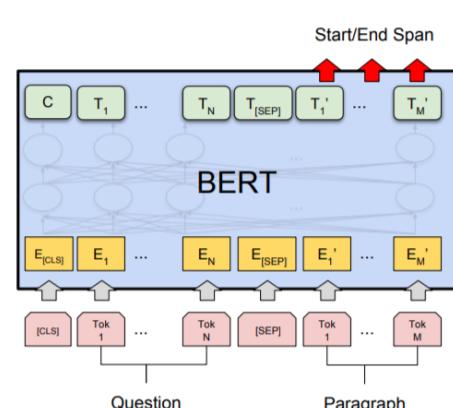
- Add task-specific layers (usually one or two linear layers) on top of the embeddings produced by the PLMs (sequence-level tasks use [CLS] token embeddings; token-level tasks use real token embeddings)
- Task-specific layers and the PLMs are jointly fine-tuned with task-specific training data



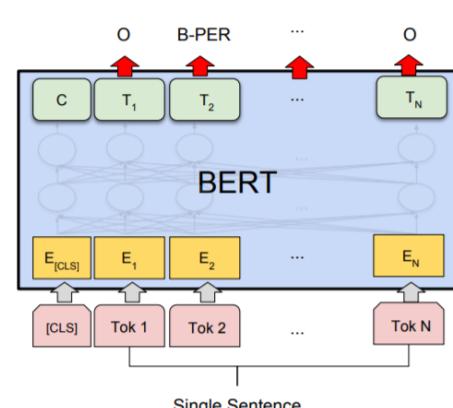
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



(b) Single Sentence Classification Tasks:  
SST-2, CoLA



(c) Question Answering Tasks:  
SQuAD v1.1



(d) Single Sentence Tagging Tasks:  
CoNLL-2003 NER

# Outline

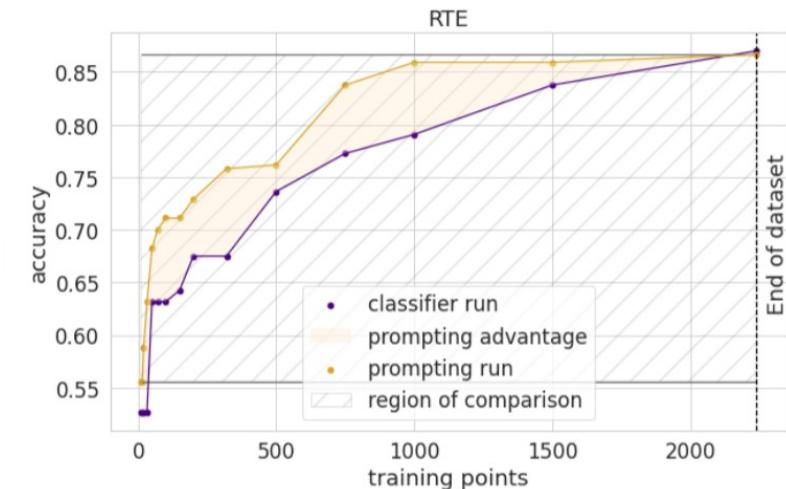
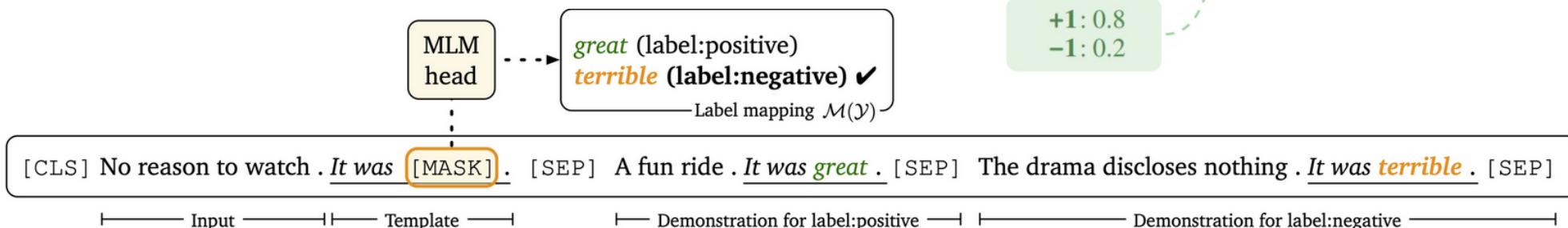
---

- ❑ Introduction to text representations
- ❑ Static word embeddings
- ❑ Deep contextualized embeddings via neural language models
  - ❑ Language Model Pretraining
  - ❑ Language Model Deployment
    - ❑ Standard fine-tuning
    - ❑ Prompt-based methods
    - ❑ Parameter-efficient tuning



# Prompt-Based Fine-Tuning of PLMs

- Task descriptions are created to convert training examples to cloze questions
- Highly resemble the pretraining tasks (MLM) so that pretraining knowledge could be better leveraged
- Better than standard fine-tuning especially for few-shot settings



Schick, T., & Schütze, H. (2021). Exploiting cloze questions for few shot text classification and natural language inference. EACL.

Le Scao, T., & Rush, A. M. (2021). How many data points is a prompt worth? NAACL.

# Prompt-Based Fine-Tuning of PLMs

- ❑ Further improve prompt-based few-shot fine-tuning:
  - ❑ Prompt templates and label words can be automatically generated
  - ❑ Demonstrations can be concatenated with target sequences to provide hints

	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	Subj (acc)	TREC (acc)	CoLA (Matt.)
Majority <sup>†</sup>	50.9	23.1	50.0	50.0	50.0	50.0	18.8	0.0
Prompt-based zero-shot <sup>†</sup>	83.6	35.0	80.8	79.5	67.6	51.4	32.0	2.0
“GPT-3” in-context learning	84.8 (1.3)	30.6 (0.9)	80.5 (1.7)	87.4 (0.8)	63.8 (2.1)	53.6 (1.0)	26.2 (2.4)	-1.5 (2.4)
Fine-tuning	81.4 (3.8)	43.9 (2.0)	76.9 (5.9)	75.8 (3.2)	72.0 (3.8)	90.8 (1.8)	88.8 (2.1)	<b>33.9</b> (14.3)
Prompt-based FT (man)	92.7 (0.9)	47.4 (2.5)	87.0 (1.2)	90.3 (1.0)	84.7 (2.2)	91.2 (1.1)	84.8 (5.1)	9.3 (7.3)
+ demonstrations	92.6 (0.5)	<b>50.6</b> (1.4)	86.6 (2.2)	90.2 (1.2)	<b>87.0</b> (1.1)	<b>92.3</b> (0.8)	87.5 (3.2)	18.7 (8.8)
Prompt-based FT (auto)	92.3 (1.0)	49.2 (1.6)	85.5 (2.8)	89.0 (1.4)	85.8 (1.9)	91.2 (1.1)	88.2 (2.0)	14.0 (14.1)
+ demonstrations	<b>93.0</b> (0.6)	49.5 (1.7)	<b>87.7</b> (1.4)	<b>91.0</b> (0.9)	86.5 (2.6)	91.4 (1.8)	<b>89.4</b> (1.7)	21.8 (15.9)
Fine-tuning (full) <sup>†</sup>	95.0	58.7	90.8	89.4	87.8	97.0	97.4	62.6
	MNLI (acc)	MNLI-mm (acc)	SNLI (acc)	QNLI (acc)	RTE (acc)	MRPC (F1)	QQP (F1)	STS-B (Pear.)
Majority <sup>†</sup>	32.7	33.0	33.8	49.5	52.7	81.2	0.0	-
Prompt-based zero-shot <sup>†</sup>	50.8	51.7	49.5	50.8	51.3	61.9	49.7	-3.2
“GPT-3” in-context learning	52.0 (0.7)	53.4 (0.6)	47.1 (0.6)	53.8 (0.4)	60.4 (1.4)	45.7 (6.0)	36.1 (5.2)	14.3 (2.8)
Fine-tuning	45.8 (6.4)	47.8 (6.8)	48.4 (4.8)	60.2 (6.5)	54.4 (3.9)	76.6 (2.5)	60.7 (4.3)	53.5 (8.5)
Prompt-based FT (man)	68.3 (2.3)	70.5 (1.9)	77.2 (3.7)	64.5 (4.2)	69.1 (3.6)	74.5 (5.3)	65.5 (5.3)	71.0 (7.0)
+ demonstrations	<b>70.7</b> (1.3)	<b>72.0</b> (1.2)	<b>79.7</b> (1.5)	<b>69.2</b> (1.9)	68.7 (2.3)	77.8 (2.0)	<b>69.8</b> (1.8)	73.5 (5.1)
Prompt-based FT (auto)	68.3 (2.5)	70.1 (2.6)	77.1 (2.1)	68.3 (7.4)	<b>73.9</b> (2.2)	76.2 (2.3)	67.0 (3.0)	75.0 (3.3)
+ demonstrations	70.0 (3.6)	<b>72.0</b> (3.1)	77.5 (3.5)	68.5 (5.4)	71.1 (5.3)	<b>78.1</b> (3.4)	67.7 (5.8)	<b>76.4</b> (6.2)
Fine-tuning (full) <sup>†</sup>	89.8	89.5	92.6	93.3	80.9	91.4	81.7	91.9

Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. ACL

# Prompt-Based Zero-Shot Inference

- ❑ Even without any training, knowledge can be extracted from PLMs through cloze patterns
- ❑ PLMs can serve as knowledge bases
  - ❑ Pros: require no schema engineering, and support an open set of queries
  - ❑ Cons: retrieved answers are not guaranteed to be accurate
- ❑ Could be used for unsupervised open-domain QA systems

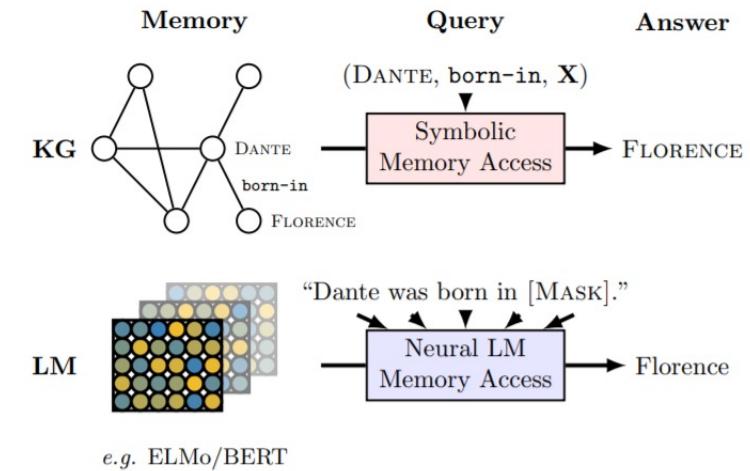


Figure 1: Querying knowledge bases (KB) and language models (LM) for factual knowledge.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases? EMNLP.

# Prompt-Based Few-Shot Inference

- Large PLMs (e.g., GPT-3) have strong few-shot learning ability **without** any tuning on large task-specific training sets
- Generate answers based on natural language descriptions and prompts

The three settings we explore for in-context learning

## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



## One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

## Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



# Zero-Shot Fine-Tuning of PLMs

---

- ❑ Prompt-based approaches have remarkable few-shot fine-tuning performance, but their zero-shot performance is significantly worse
- ❑ Without any task-specific samples, it is challenging for PLMs to interpret the prompts that come in different formats and are unseen in the pretraining data
- ❑ The current mainstream of zero-shot learning is based on transfer learning
  - ❑ Train PLMs on a large variety of different tasks with abundant annotations, and transfer to unseen tasks
  - ❑ Require many **cross-task annotations** and **gigantic model sizes** which are not practical for common application scenarios

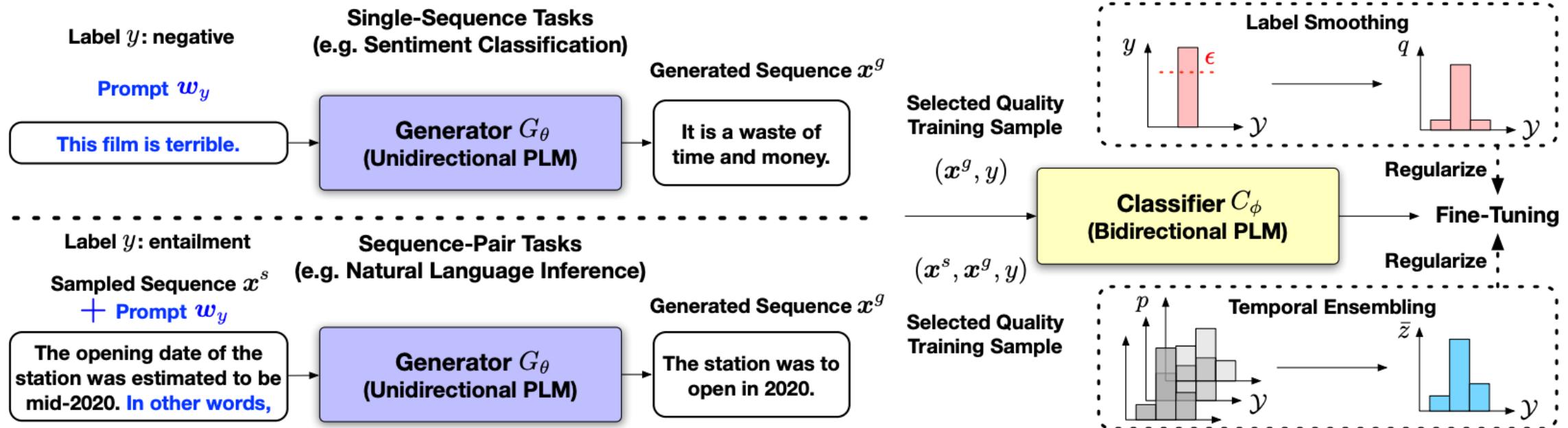
# **Zero-Shot Fine-Tuning of PLMs**

---

- ❑ Can we do fully zero-shot learning, without any task-related or cross-task annotations?
- ❑ When there are no training data, we can create them from scratch using PLMs!
- ❑ Humans can generate training data pertaining to a specific label upon given a label-descriptive prompt (e.g., “write a negative review:”)
- ❑ We can leverage the strong text generation power of PLMs to do the same job

# Prompt-Based Zero-Shot Training Data Generation

- SuperGen: A **Supervision Generation** approach
- Use a unidirectional PLM to generate class-conditioned texts guided by prompts
- Fine-tune a bidirectional PLM on the generated data for the corresponding task



Meng, Y., Huang, J., Zhang, Y., & Han, J. (2022). Generating Training Data with Language Models: Towards Zero-Shot Language Understanding. *arXiv preprint arXiv:2202.04538*.

# Zero-Shot Learning Results

- Using the same prompt-based fine-tuning method, zero-shot SuperGen (fine-tuned on generated training data) is comparable or even better than strong few-shot methods (fine-tuned on 32 manually annotated training samples per class)

Method	MNLI-(m/mm) (Acc.)	QQP (F1)	QNLI (Acc.)	SST-2 (Acc.)	CoLA (Matt.)	RTE (Acc.)	MRPC (F1)	AVG
<b>Zero-Shot Setting:</b> No task-specific data (neither labeled nor unlabeled).								
Prompting <sup>†</sup>	50.8 <sub>0.0</sub> /51.7 <sub>0.0</sub>	49.7 <sub>0.0</sub>	50.8 <sub>0.0</sub>	83.6 <sub>0.0</sub>	2.0 <sub>0.0</sub>	51.3 <sub>0.0</sub>	61.9 <sub>0.0</sub>	50.1
SuperGen	<b>72.3</b> <sub>0.5</sub> / <b>73.8</b> <sub>0.5</sub>	<b>66.1</b> <sub>1.1</sub>	<b>73.3</b> <sub>1.9</sub>	<b>92.8</b> <sub>0.6</sub>	<b>32.7</b> <sub>5.5</sub>	<b>65.3</b> <sub>1.2</sub>	82.2 <sub>0.5</sub>	<b>69.4</b>
- data selection	63.7 <sub>1.5</sub> /64.2 <sub>1.6</sub>	62.3 <sub>2.2</sub>	63.9 <sub>3.2</sub>	91.3 <sub>2.0</sub>	30.5 <sub>8.8</sub>	62.4 <sub>1.5</sub>	81.6 <sub>0.2</sub>	65.1
- label smooth	70.7 <sub>0.8</sub> /72.1 <sub>0.7</sub>	65.1 <sub>0.9</sub>	71.4 <sub>2.5</sub>	91.0 <sub>0.9</sub>	9.5 <sub>1.0</sub>	64.8 <sub>1.1</sub>	<b>83.0</b> <sub>0.7</sub>	65.2
- temporal ensemble	62.0 <sub>4.6</sub> /63.6 <sub>4.8</sub>	63.9 <sub>0.3</sub>	72.4 <sub>2.0</sub>	92.5 <sub>0.9</sub>	23.5 <sub>7.0</sub>	63.5 <sub>1.0</sub>	78.8 <sub>2.2</sub>	65.3
<b>Few-Shot Setting:</b> Use 32 labeled samples/class (half for training and half for development).								
Fine-tuning <sup>†</sup>	45.8 <sub>6.4</sub> /47.8 <sub>6.8</sub>	60.7 <sub>4.3</sub>	60.2 <sub>6.5</sub>	81.4 <sub>3.8</sub>	<b>33.9</b> <sub>14.3</sub>	54.4 <sub>3.9</sub>	76.6 <sub>2.5</sub>	59.1
Manual prompt <sup>†</sup>	68.3 <sub>2.3</sub> /70.5 <sub>1.9</sub>	65.5 <sub>5.3</sub>	64.5 <sub>4.2</sub>	92.7 <sub>0.9</sub>	9.3 <sub>7.3</sub>	69.1 <sub>3.6</sub>	74.5 <sub>5.3</sub>	63.6
+ demonstration <sup>†</sup>	<b>70.7</b> <sub>1.3</sub> / <b>72.0</b> <sub>1.2</sub>	<b>69.8</b> <sub>1.8</sub>	<b>69.2</b> <sub>1.9</sub>	92.6 <sub>0.5</sub>	18.7 <sub>8.8</sub>	68.7 <sub>2.3</sub>	77.8 <sub>2.0</sub>	66.9
Auto prompt <sup>†</sup>	68.3 <sub>2.5</sub> /70.1 <sub>2.6</sub>	67.0 <sub>3.0</sub>	68.3 <sub>7.4</sub>	92.3 <sub>1.0</sub>	14.0 <sub>14.1</sub>	<b>73.9</b> <sub>2.2</sub>	76.2 <sub>2.3</sub>	65.8
+ demonstration <sup>†</sup>	70.0 <sub>3.6</sub> /72.0 <sub>3.1</sub>	67.7 <sub>5.8</sub>	68.5 <sub>5.4</sub>	<b>93.0</b> <sub>0.6</sub>	21.8 <sub>15.9</sub>	71.1 <sub>5.3</sub>	<b>78.1</b> <sub>3.4</sub>	<b>67.3</b>

# Outline

---

- ❑ Introduction to text representations
- ❑ Static word embeddings
- ❑ Deep contextualized embeddings via neural language models
  - ❑ Language Model Pretraining
  - ❑ Language Model Deployment
    - ❑ Standard fine-tuning
    - ❑ Prompt-based methods
    - ❑ Parameter-efficient tuning



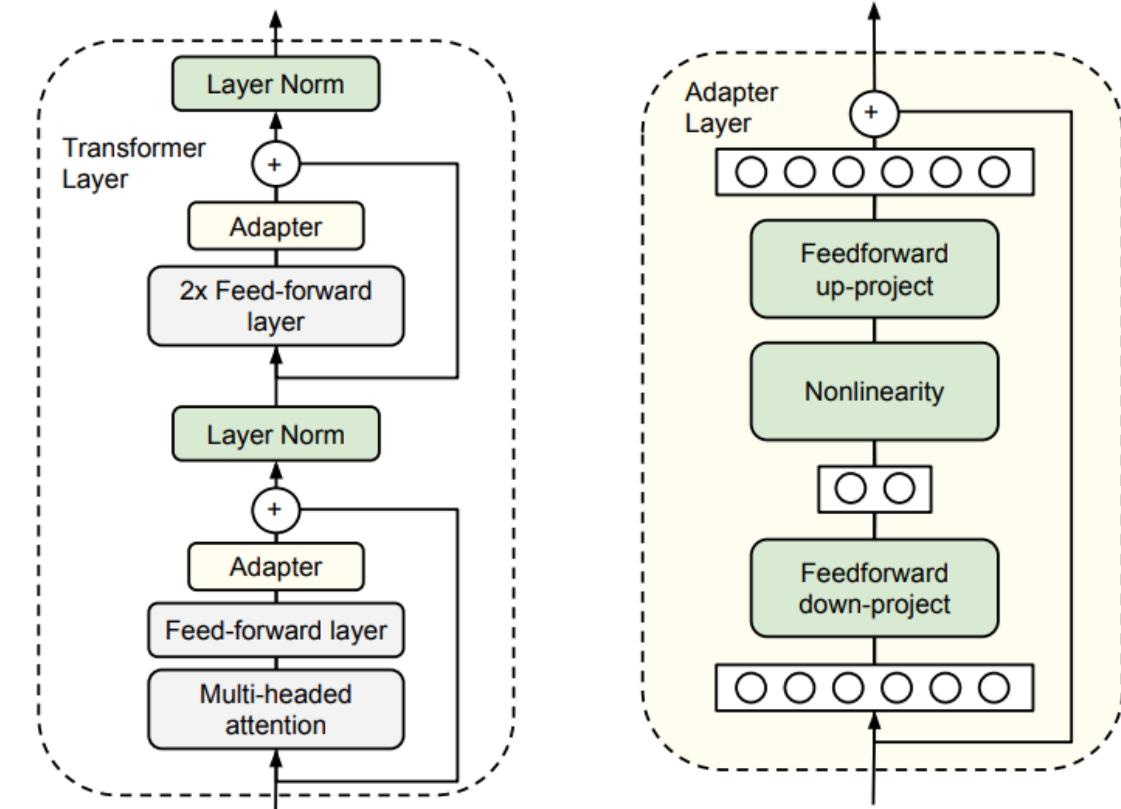
# Parameter-Efficient Tuning of PLMs

---

- ❑ Fine-tuning updates all PLM parameters at the same time
- ❑ Large PLMs can have an enormous amount of parameters that are costly to optimize
- ❑ Can we optimize only a small set of parameters in PLMs while still achieving comparable performance to fine-tuning?
- ❑ A few strategies:
  - ❑ Adapter: Insert small bottleneck modules and only update adapter + layer norm parameters
  - ❑ Prefix Tuning: Prepend tunable prefix vectors to every Transformer layer and keep other parameters unchanged
  - ❑ Low-Rank Adaptation: Use trainable low-rank matrices to approximate weight updates

# Adapter for Parameter-Efficient Tuning

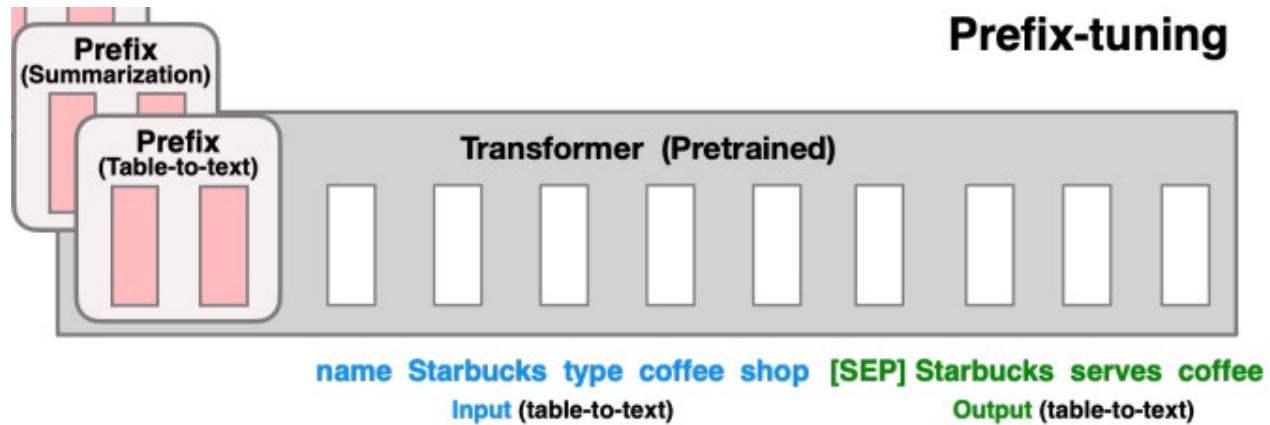
- ❑ Adapters are added twice to each Transformer layer
- ❑ Consist of a bottleneck structure (down-project + up-project)
- ❑ Only adapter parameters + layer norm parameters are updated during tuning



Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. ICML

# Prefix Tuning

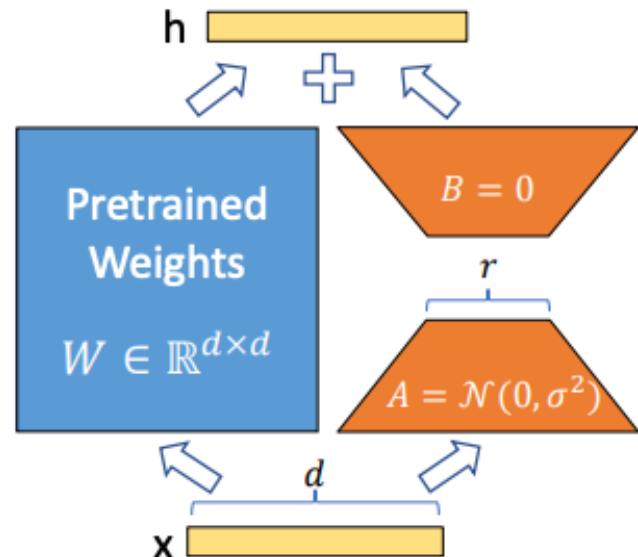
- ❑ Prefix tuning prepends trainable vectors to each Transformer layer
- ❑ Only update prefix vectors and keep other pretrained parameters unchanged
- ❑ Similar to prompt-based fine-tuning except that the prefix vectors are continuous parameters instead of natural language words



Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. ACL.

# Low-Rank Adaptation

- Inject trainable low-rank matrices into transformer layers to approximate the weight updates
- Since low-rank matrices have far less parameters than full-rank ones, training them is much more efficient than standard fine-tuning



$$W_0 + \Delta W = W_0 + [BA]$$

A and B are low-rank matrices

# References I

---

- Abu-El-Haija, S., Perozzi, B., Al-Rfou', R., & Alemi, A.A. (2018). Watch Your Step: Learning Node Embeddings via Graph Attention. NeurIPS.
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135-146.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. NeurIPS.
- Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. ICLR.
- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.
- Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. ACL
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. ICML
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. ICLR.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. ICLR.
- Le Scao, T., & Rush, A. M. (2021). How many data points is a prompt worth? NAACL.

# References II

---

- 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.
- Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. ACL.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS.
- Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781.
- Meng, Y., Huang, J., Wang, G., Zhang, C., Zhuang, H., Kaplan, L.M., & Han, J. (2019). Spherical Text Embedding. NeurIPS.
- Meng, Y., Xiong, C., Bajaj, P., Bennett, P., Han, J., & Song, X. (2021). COCO-LM: Correcting and contrasting text sequences for language model pretraining. NeurIPS.
- Meng, Y., Xiong, C., Bajaj, P., Bennett, P. N., Han, J., & Song, X. (2022). Pretraining Text Encoders with Adversarial Mixture of Training Signal Generators. ICLR.
- Meng, Y., Huang, J., Zhang, Y., & Han, J. (2022). Generating Training Data with Language Models: Towards Zero-Shot Language Understanding. arXiv preprint arXiv:2202.04538.
- Nickel, M., & Kiela, D. (2017). Poincaré Embeddings for Learning Hierarchical Representations. NIPS.
- Nickel, M., & Kiela, D. (2018). Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. ICML.

# References III

---

- Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.
- Peters, M.E., Neumann, M., Iyyer, M., Gardner, M.P., Clark, C., Lee, K., & Zettlemoyer, L.S. (2018). Deep contextualized word representations. NAACL.
- Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases? EMNLP.
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- 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.
- Schick, T., & Schütze, H. (2021). Exploiting cloze questions for few shot text classification and natural language inference. EACL.
- Tifrea, A., Bécigneul, G., & Ganea, O. (2019). Poincare Glove: Hyperbolic Word Embeddings. ICLR.
- Turian, J.P., Ratinov, L., & Bengio, Y. (2010). Word Representations: A Simple and General Method for Semi-Supervised Learning. ACL.
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. NeurIPS.

# Q&A