

# **Part I: Pre-trained Language Models**

**EDBT 2023 Tutorial: Mining Structures from Massive Texts by Exploring the Power of Pre-trained Language Models**

**Yu Zhang, Yunyi Zhang, Jiawei Han**

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

**Mar 29, 2023**

# Outline

---

- Static word embeddings 
- Euclidean space: Word2Vec, GloVe
- Hyperbolic space: Poincaré embeddings
- Spherical space: JoSE
- 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?

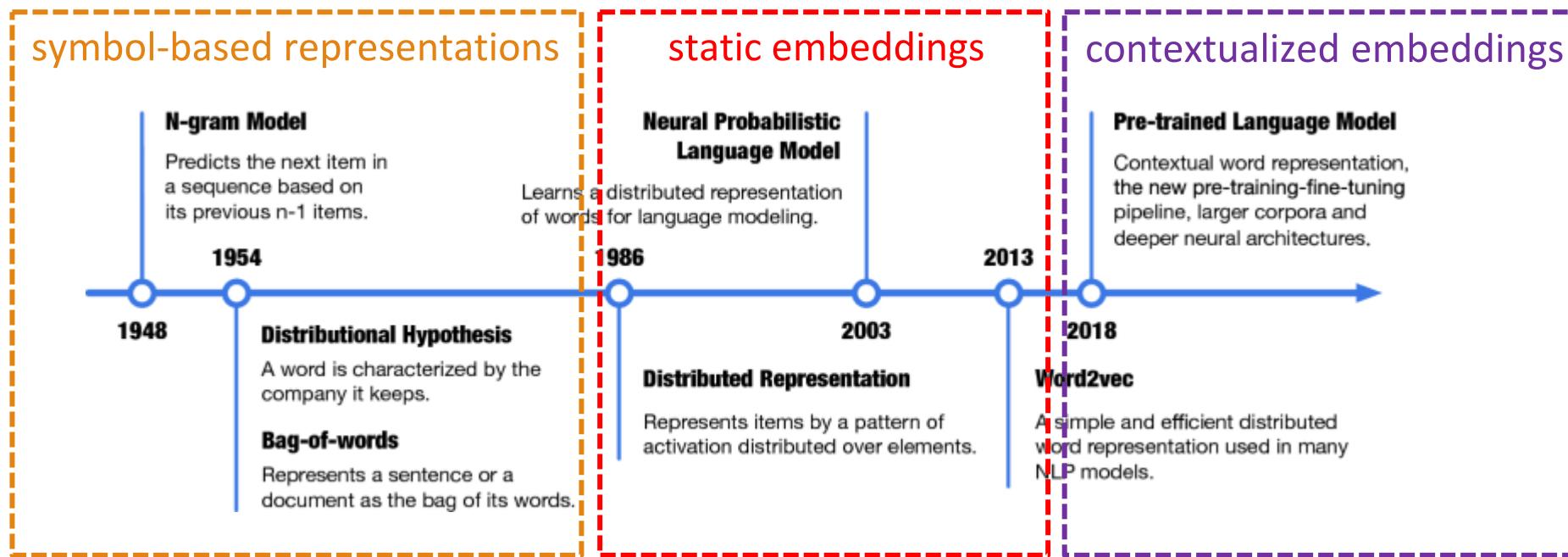


Figure from: Liu Z., Lin Y., Sun M. (2020) Representation Learning and NLP. In: Representation Learning for Natural Language Processing. Springer.

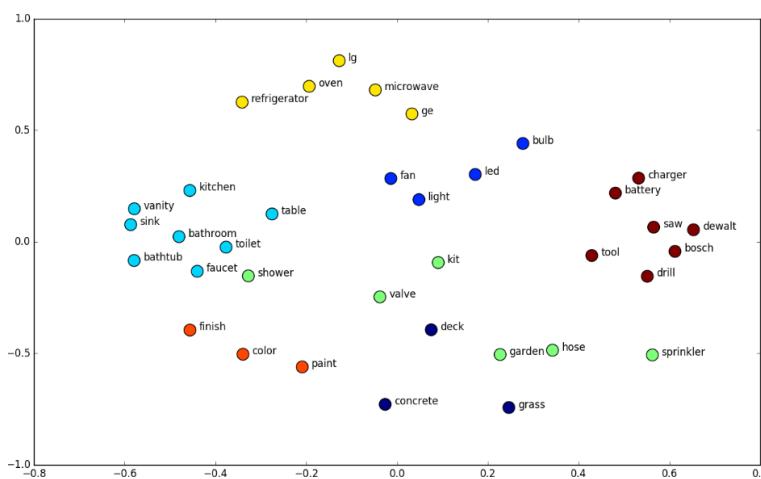
# Symbol-Based Text Representations

---

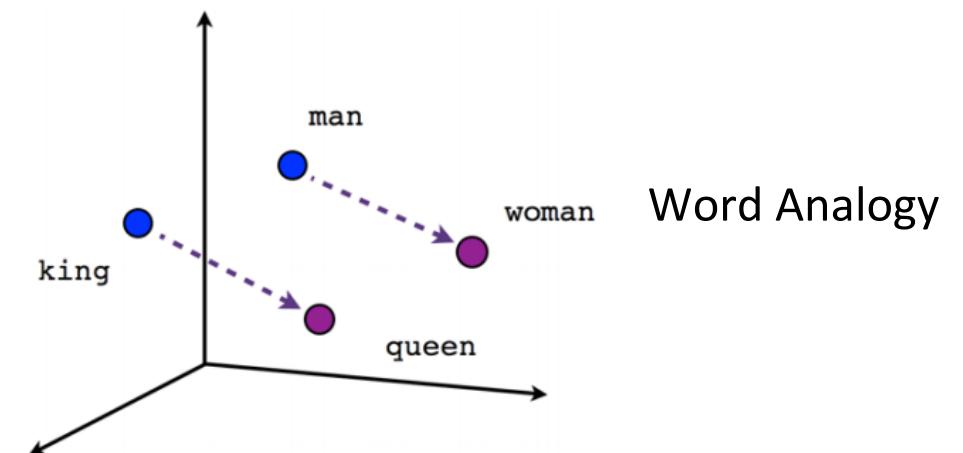
- One-to-one correspondence between text units and representation elements
  - E.g., “dogs” = [1, 0, 0, 0, 0]; “cats” = [0, 1, 0, 0, 0]; “cars” = [0, 0, 1, 0, 0]; “like” = [0, 0, 0, 1, 0]; “I” = [0, 0, 0, 0, 1]
- Bag-of-words representation of documents: Describe a document according to which words are 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

# 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

---

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



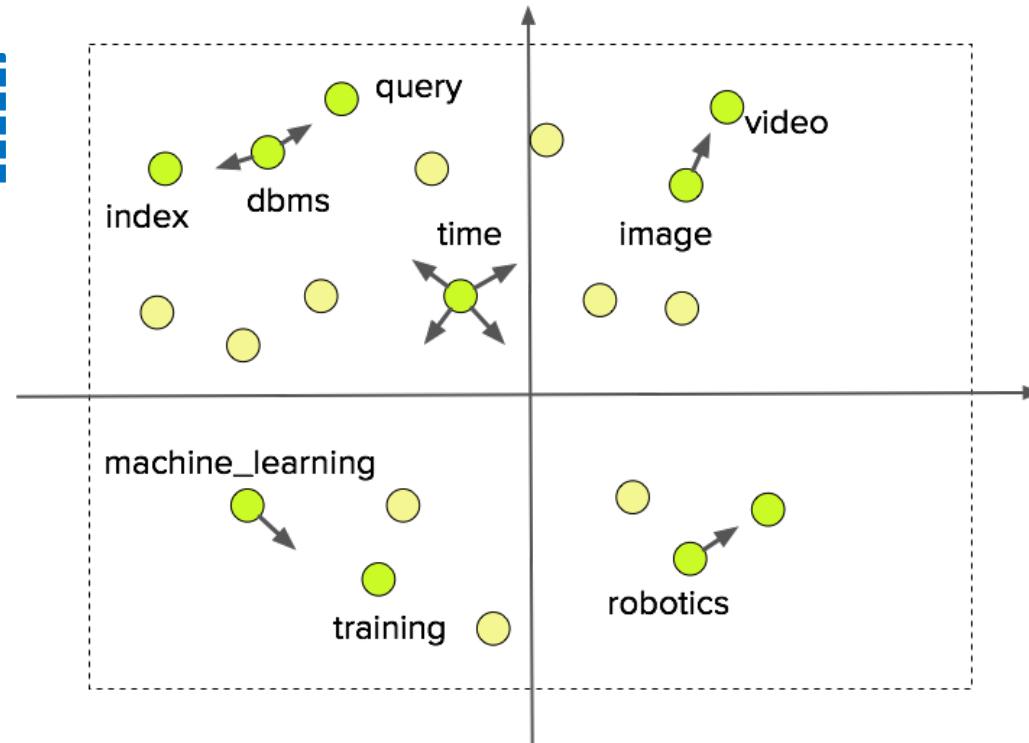
# Word2Vec

- 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**

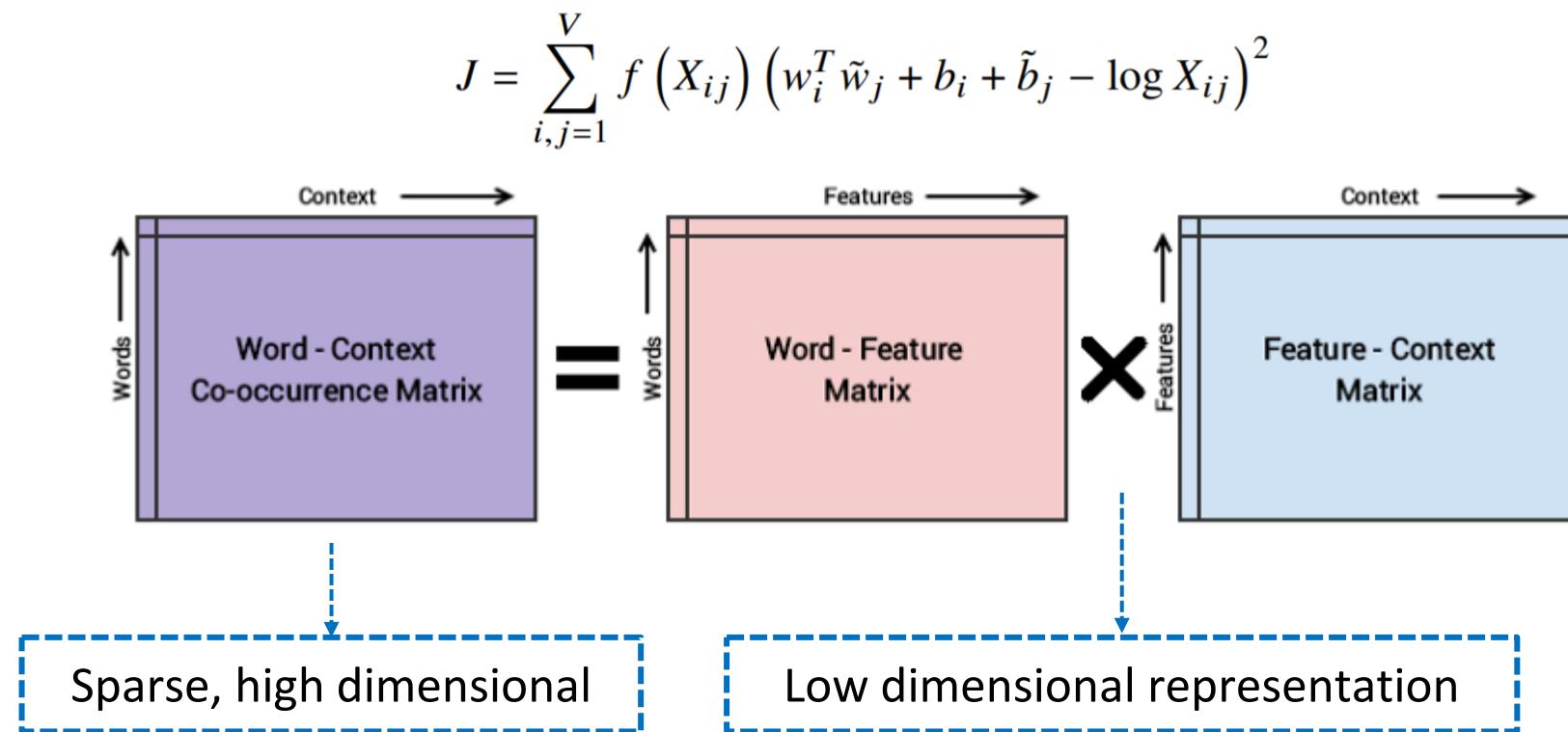
$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$$

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



# 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



# Outline

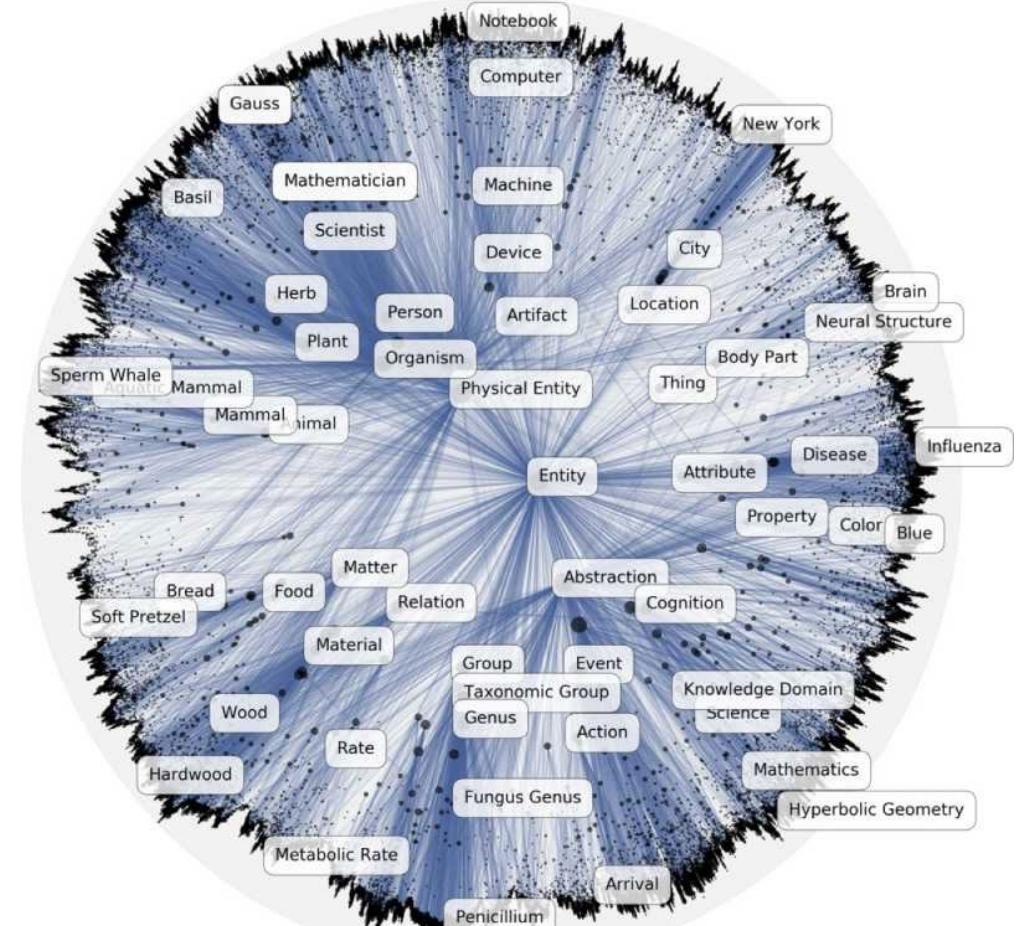
---

- 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

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



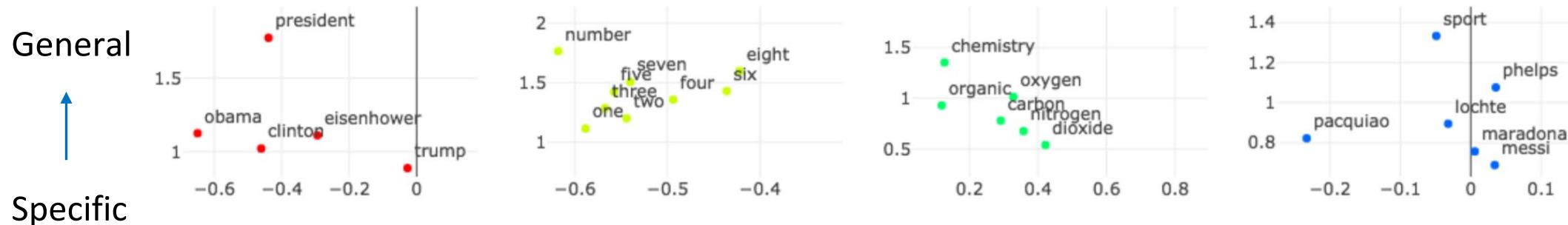
# Texts in Hyperbolic Space: Poincaré GloVe

- 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

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

Hyperbolic metric

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



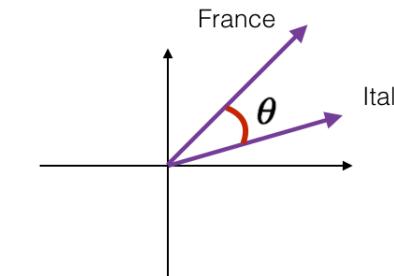
# Outline

---

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

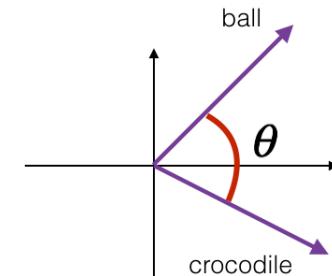
# Directional Analysis for Text Embeddings

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



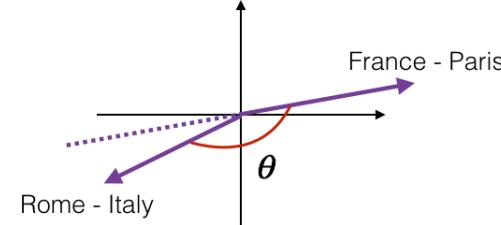
France and Italy are quite similar

$$\theta \text{ is close to } 0^\circ \\ \cos(\theta) \approx 1$$



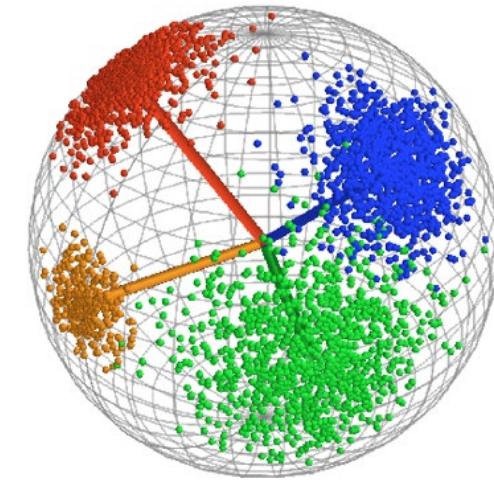
ball and crocodile are not similar

$$\theta \text{ is close to } 90^\circ \\ \cos(\theta) \approx 0$$



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

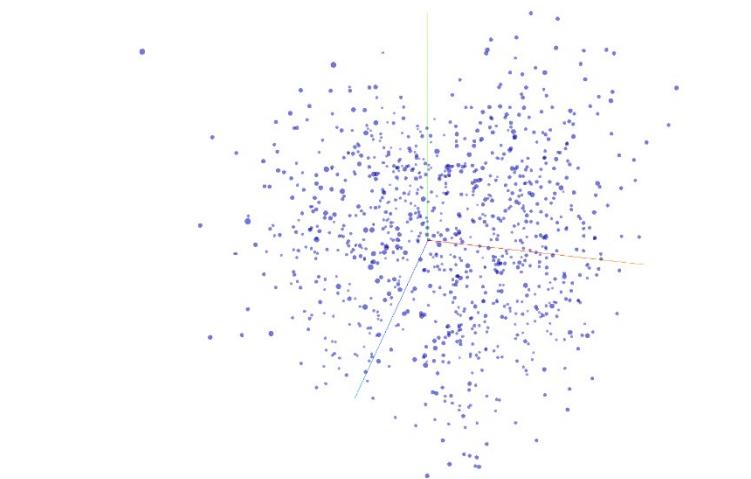
$$\theta \text{ is close to } 180^\circ \\ \cos(\theta) \approx -1$$



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

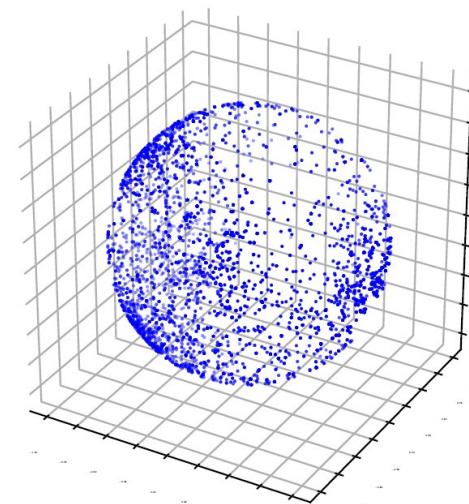
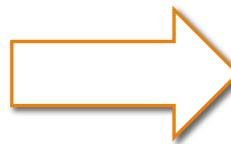
# Issues with Previous Embedding Frameworks

- Although directional similarity has shown effective for various applications, previous embeddings (e.g., Word2Vec, GloVe) 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

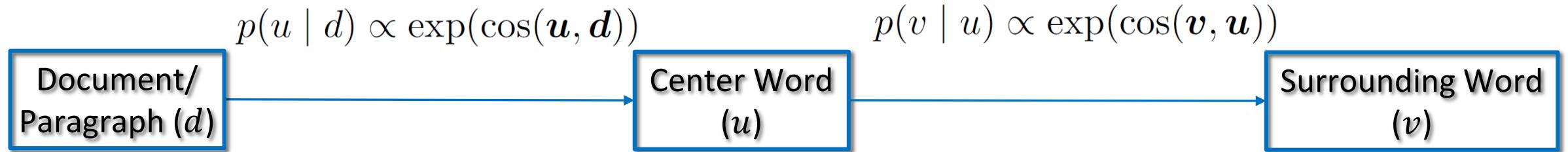
Post-processing  
(Normalization)



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

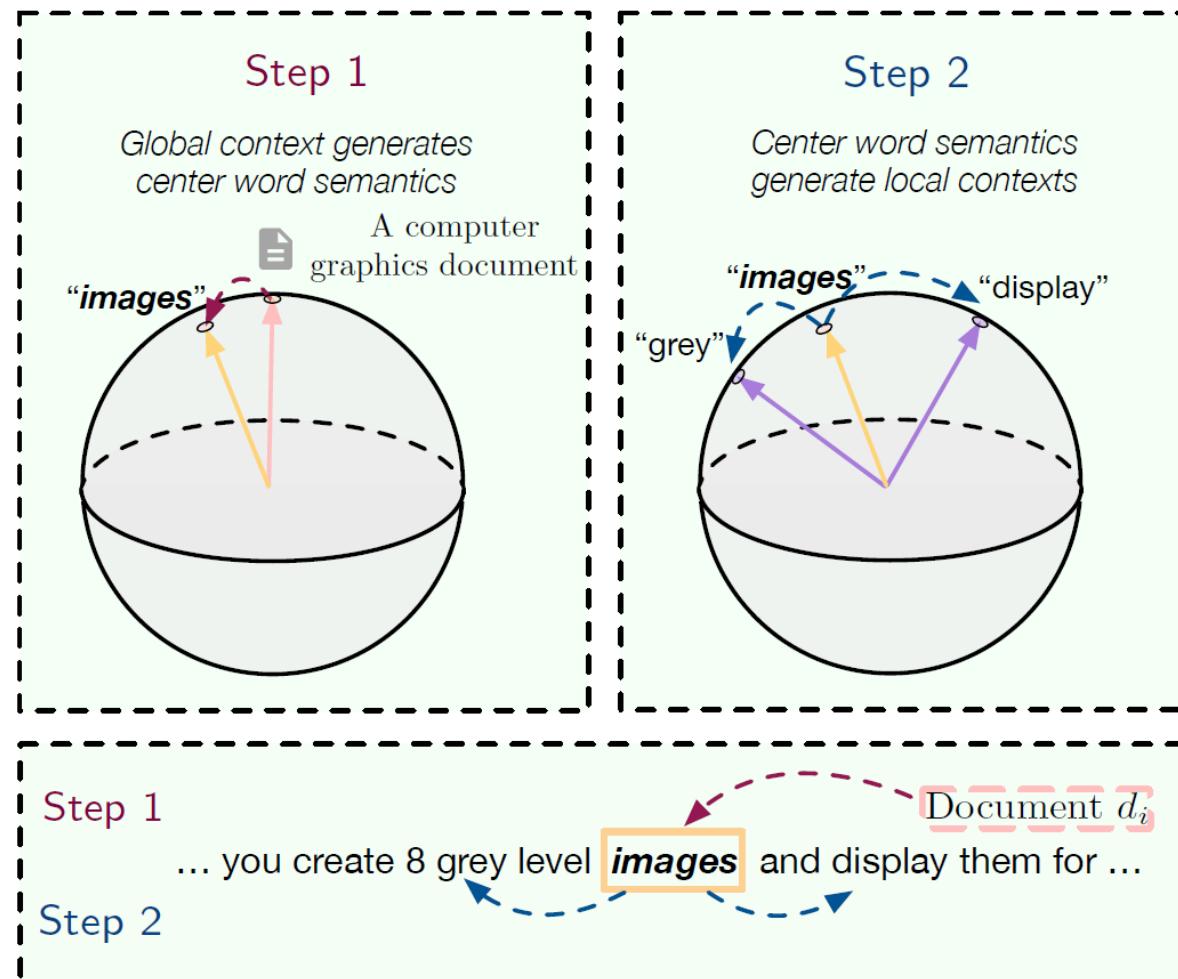
# Spherical Text Embedding: Generative Model

- 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:



# Spherical Text Embedding: Illustration

## □ Understanding the spherical generative model



# Outline

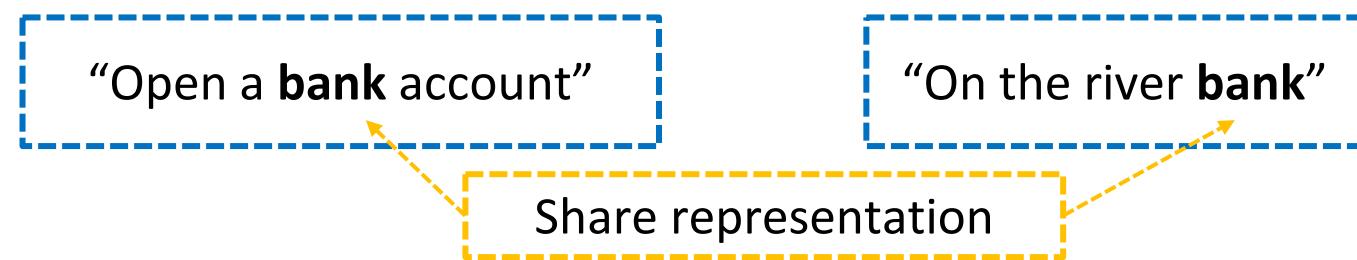
---

- Static word embeddings
- Deep contextualized embeddings via neural language models 
- Decoder-only (unidirectional) PLM
- Encoder-only (bidirectional) PLM
- Encoder-decoder (sequence-to-sequence) PLM
- How to use PLMs: Fine-tuning and prompting

# 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

# Pre-trained Language Models: Overview

---

- The “pre-train & finetune” paradigm has become the prominent practice in a wide variety of text applications
- “Pre-training”: Train deep language models (usually Transformer models) via **self-supervised objectives** on **large-scale general-domain corpora**
  - All Wikipedia articles, all PubMed papers, billions of tweets, ...
- “Fine-tuning”: Adapt the PLMs to downstream tasks using task-specific data
  - Text Classification, question answering, entity recognition, ...
- The power of PLMs: Encode generic linguistic features and knowledge learned through large-scale pre-training, which can be effectively transferred to the target applications

# Categorization of Pre-trained Language Models

---

- ❑ 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

---

- Static word embeddings
- Deep contextualized embeddings via neural language models
  - Decoder-only (unidirectional) PLM 
  - Encoder-only (bidirectional) PLM
  - Encoder-decoder (sequence-to-sequence) PLM
  - How to use PLMs: Fine-tuning and prompting

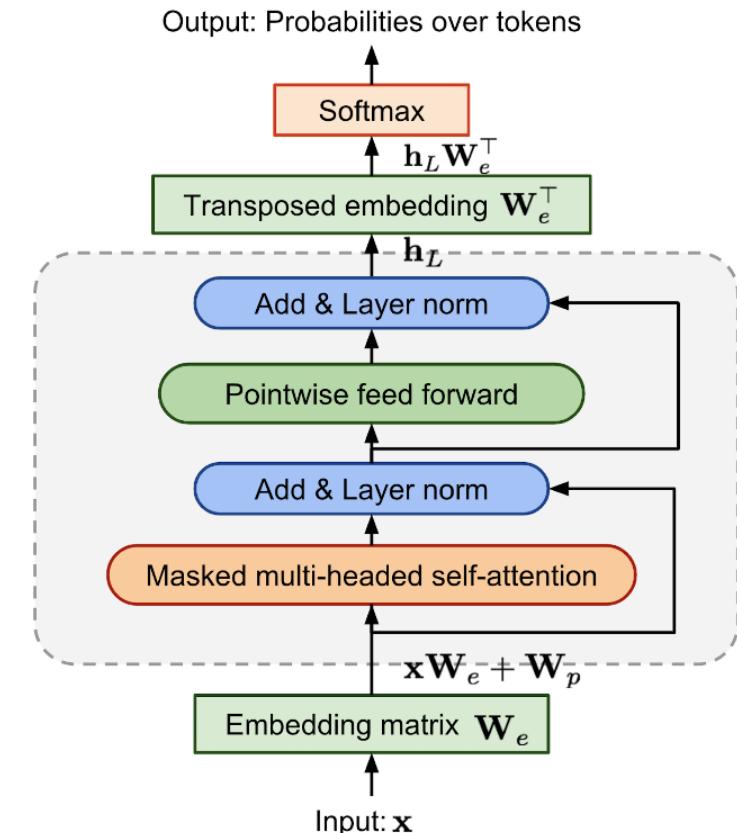
# GPT-Style Pre-training: Introduction

- Generative Pre-training (GPT [1], GPT-2 [2], GPT-3 [3], GPT-4 [4]):
- 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)
- Unidirectional PLMs 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).



- [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.
- [4] OpenAI. (2023). GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.

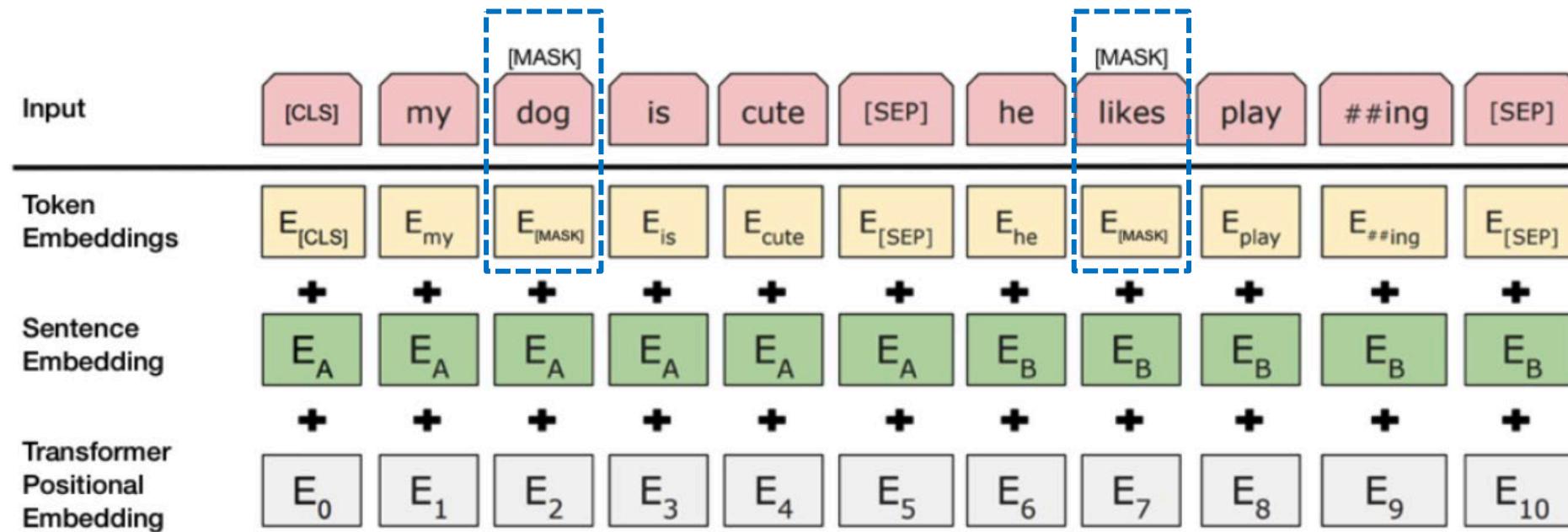
# Outline

---

- Static word embeddings
- Deep contextualized embeddings via neural language models
  - Decoder-only (unidirectional) PLM
  - Encoder-only (bidirectional) PLM 
  - Encoder-decoder (sequence-to-sequence) PLM
  - How to use PLMs: Fine-tuning and prompting

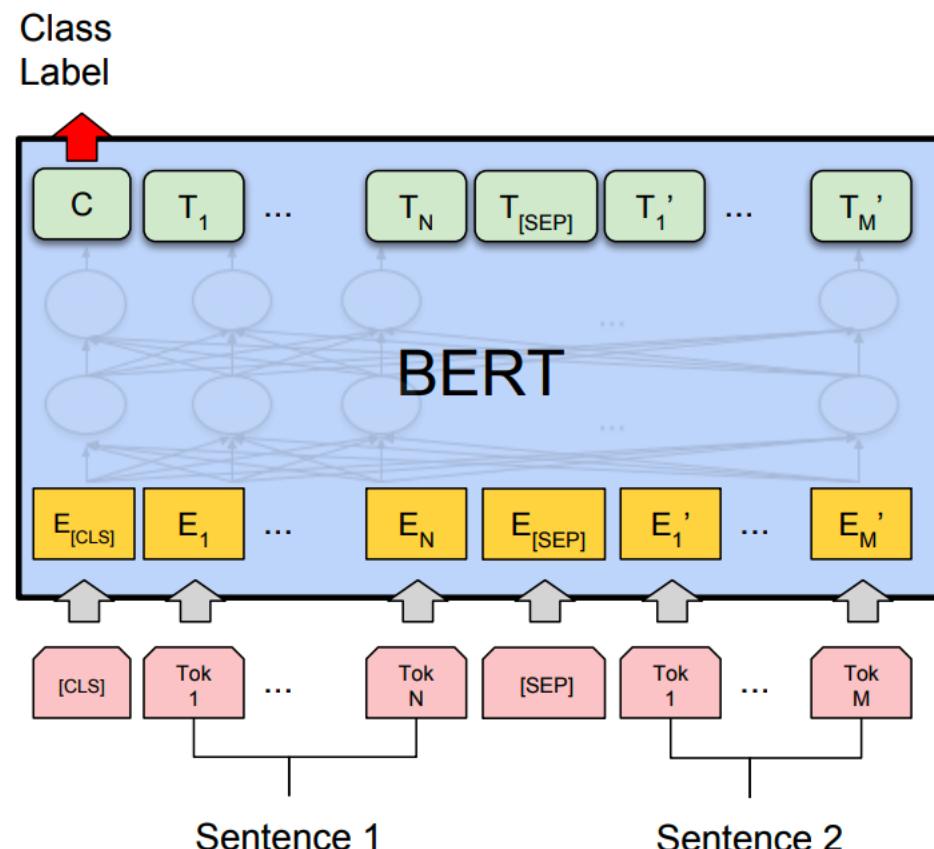
# BERT: Masked Language Modeling

- 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



# 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



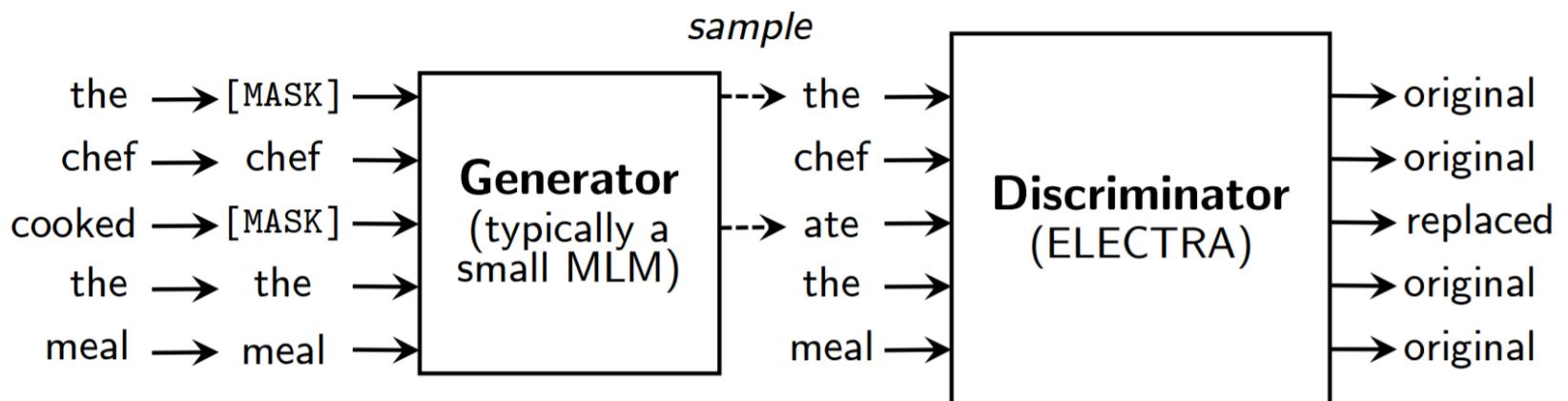
# RoBERTa

- 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

# ELECTRA

- Change masked language modeling to a more sample-efficient pre-training 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)



# 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>

# Outline

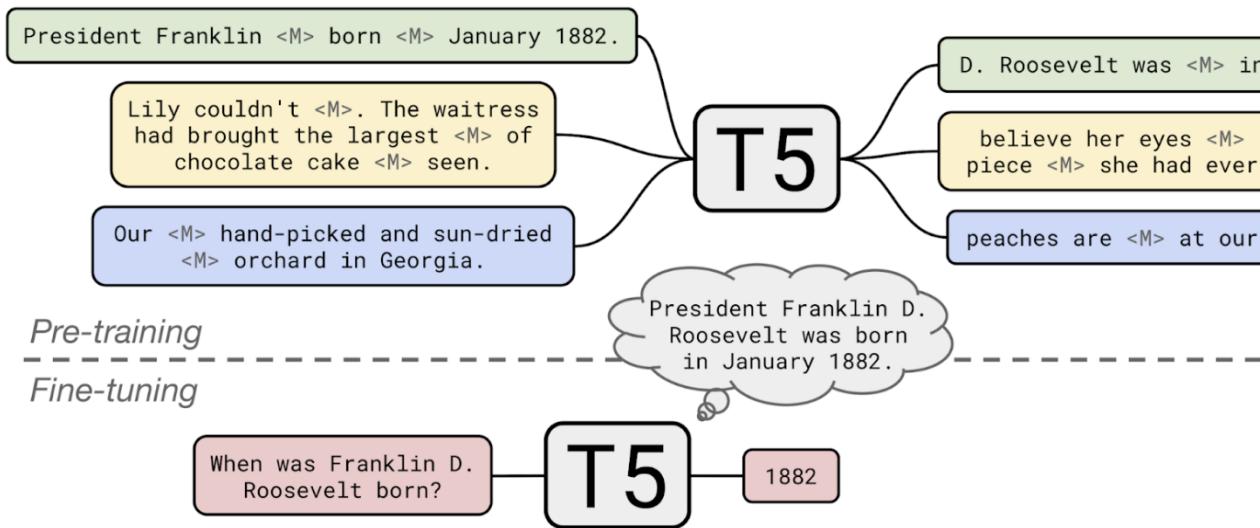
---

- Static word embeddings
- Deep contextualized embeddings via neural language models
  - Decoder-only (unidirectional) PLM
  - Encoder-only (bidirectional) PLM
  - Encoder-decoder (sequence-to-sequence) PLM
  - How to use PLMs: Fine-tuning and prompting



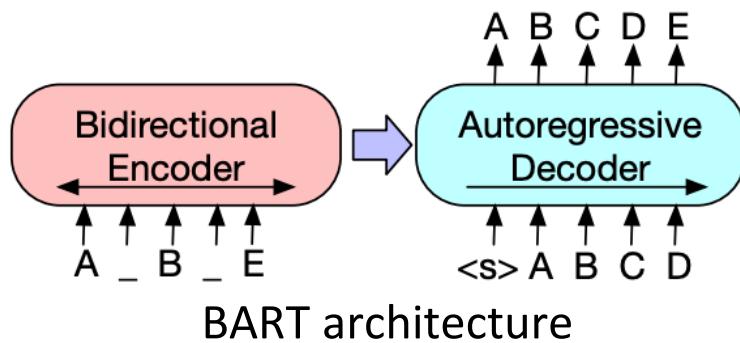
# T5

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

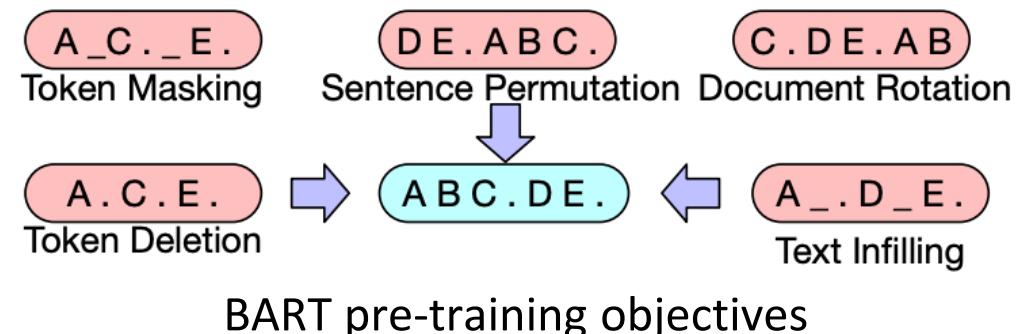


# BART

- ❑ BART: Denoising autoencoder for pre-training sequence-to-sequence models
- ❑ Pre-training: 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

---

- Static word embeddings
- Deep contextualized embeddings via neural language models
  - Decoder-only (unidirectional) PLM
  - Encoder-only (bidirectional) PLM
  - Encoder-decoder (sequence-to-sequence) PLM
  - How to use PLMs: Fine-tuning and prompting



# How to use PLMs?

---

- Pre-trained 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** (will not be covered due to time limit): Only update a small portion of PLM parameters and keep other (majority) parameters unchanged

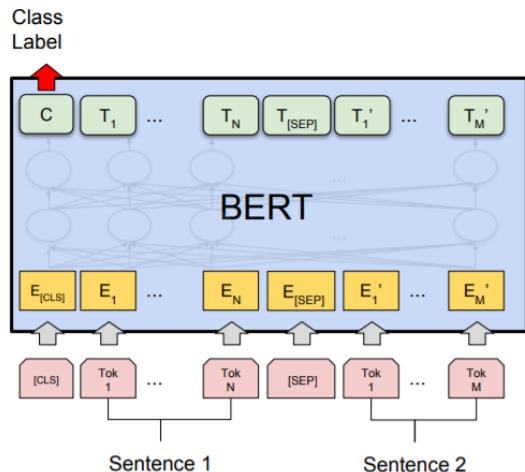
Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. ACL.

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

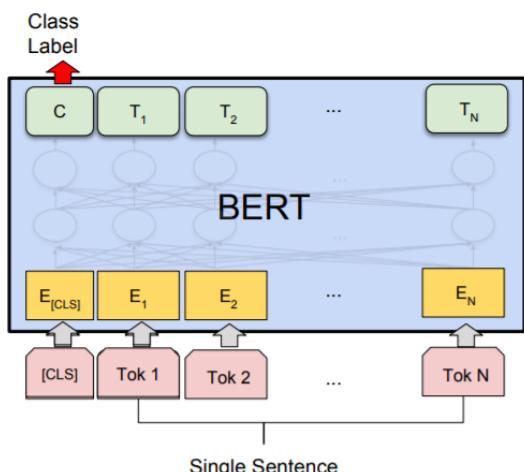
Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. ICLR.

# Standard Fine-Tuning of PLMs

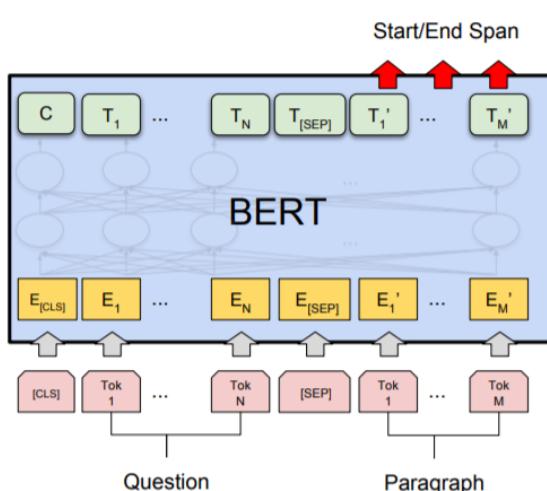
- Add task-specific layers (usually one or two linear layers) on top of the embeddings produced by the PLMs (sequence-level tasks use [CLS] token embeddings; token-level tasks use real token embeddings)
- Task-specific layers and the PLMs are jointly fine-tuned with task-specific training data



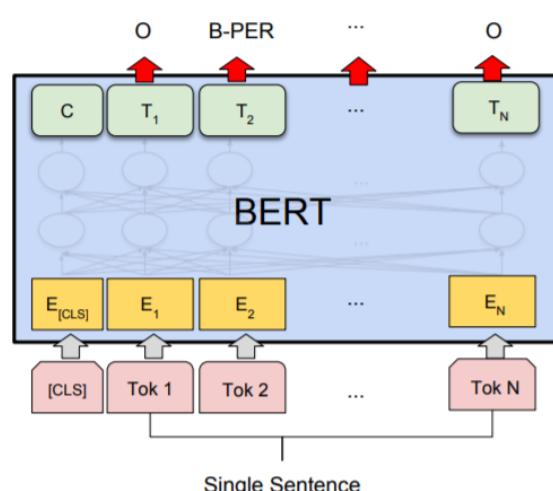
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



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



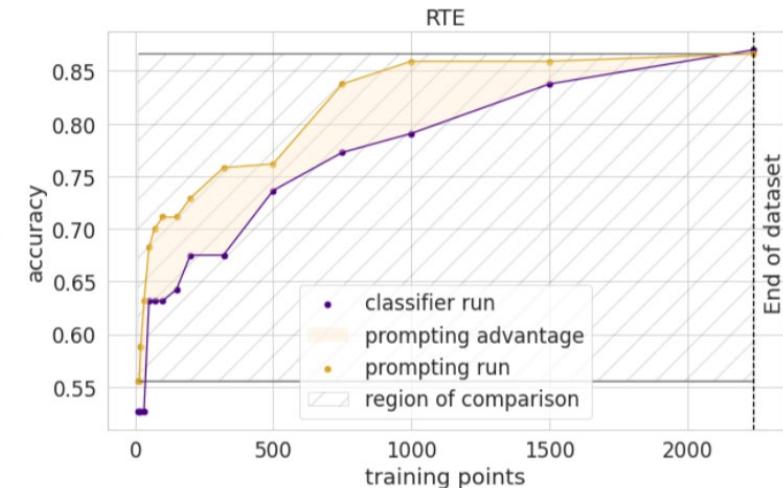
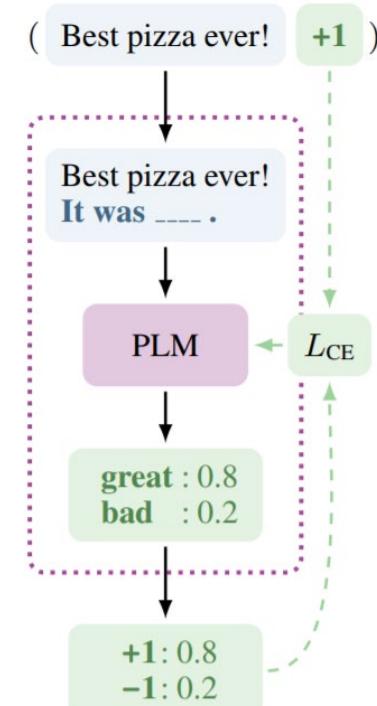
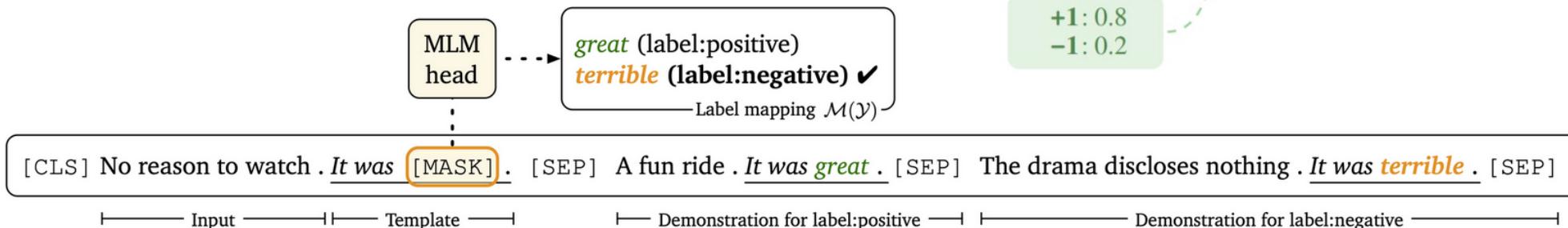
(c) Question Answering Tasks:  
SQuAD v1.1



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

# Prompt-Based Fine-Tuning of PLMs

- Task descriptions are created to convert training examples to cloze questions
- Highly resemble the pre-training tasks (MLM) so that pre-training 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

# 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

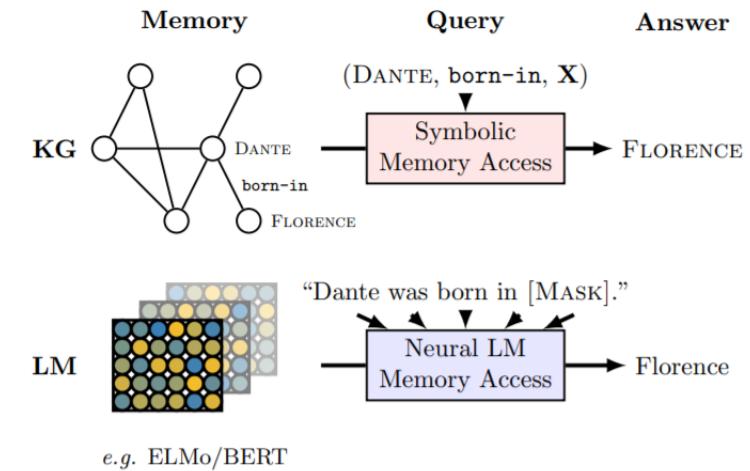


Figure 1: Querying knowledge bases (KB) and language models (LM) for factual knowledge.

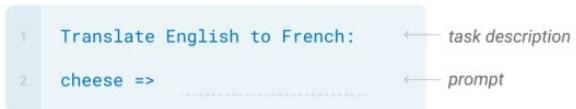
# Prompt-Based Few-Shot Inference

- 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

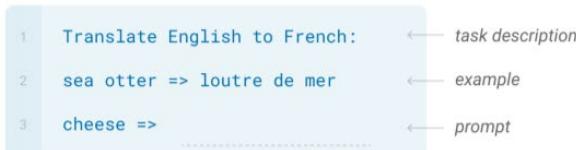
## Zero-shot

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.



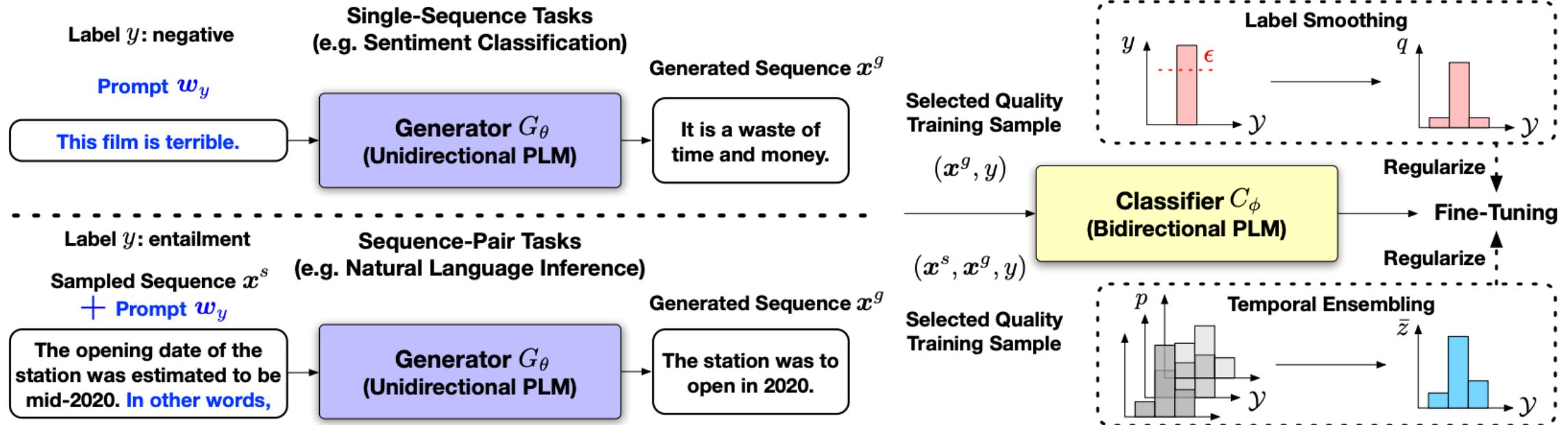
# Zero-Shot Fine-Tuning of PLMs

---

- ❑ 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 pre-training 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
- ❑ 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



# 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>

# References

---

- 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.
- Le Scao, T., & Rush, A. M. (2021). How many data points is a prompt worth? NAACL.
- 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.
- 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.
- Meng, Y., Huang, J., Wang, G., Zhang, C., Zhuang, H., Kaplan, L.M., & Han, J. (2019). Spherical Text Embedding. NeurIPS.

# References

---

- Meng, Y., Huang, J., Zhang, Y., & Han, J. (2022). Generating Training Data with Language Models: Towards Zero-Shot Language Understanding. NeurIPS.
- Nickel, M., & Kiela, D. (2017). Poincaré Embeddings for Learning Hierarchical Representations. NIPS.
- OpenAI. (2023). GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.
- Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.
- 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.

# Q&A