Part I: Pre-Trained Language Models

AAAI 2022 Tutorial

Pre-Trained Language Representations for Text Mining

Yu Meng, Jiaxin Huang, Yu Zhang, Jiawei Han

Computer Science, University of Illinois at Urbana-Champaign

February 23, 2022

Outline

- Introduction to text representations
- Context-free 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?

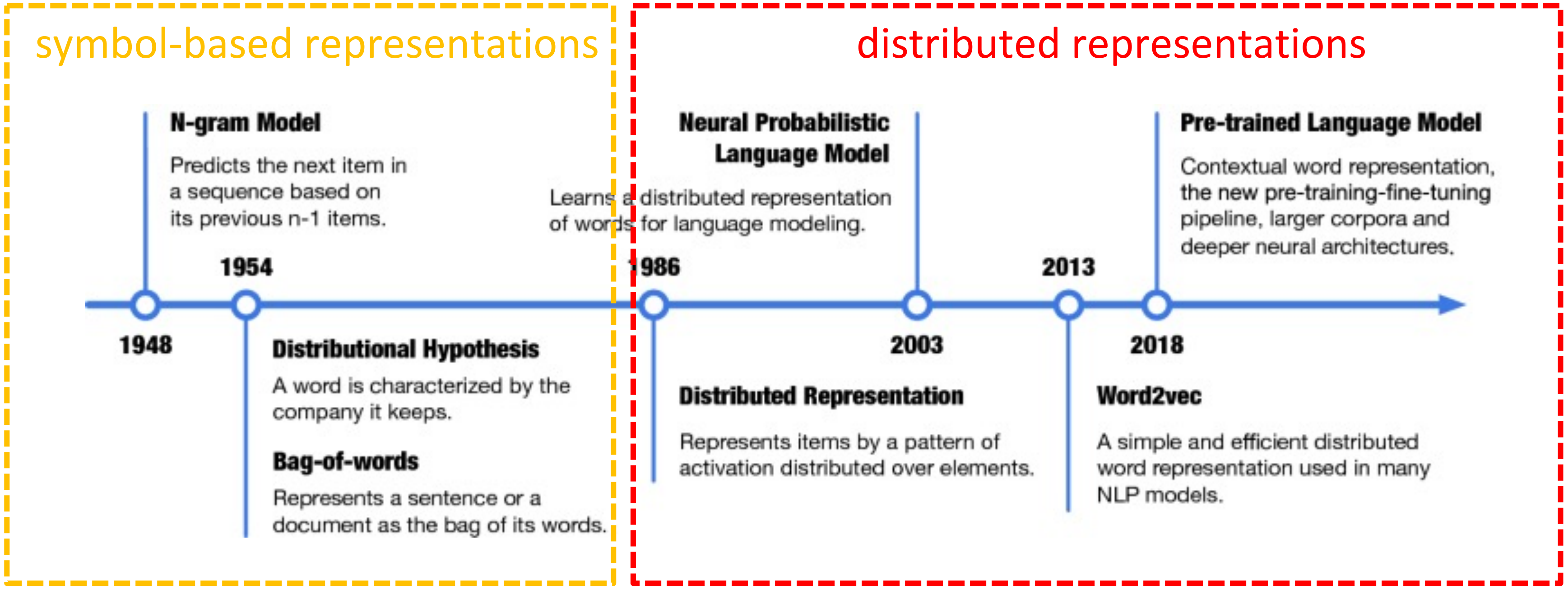


Figure from: Liu Z., Lin Y., Sun M. (2020) Representation Learning and NLP. In: Representation Learning for Natural Language Processing. Springer, Singapore.

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

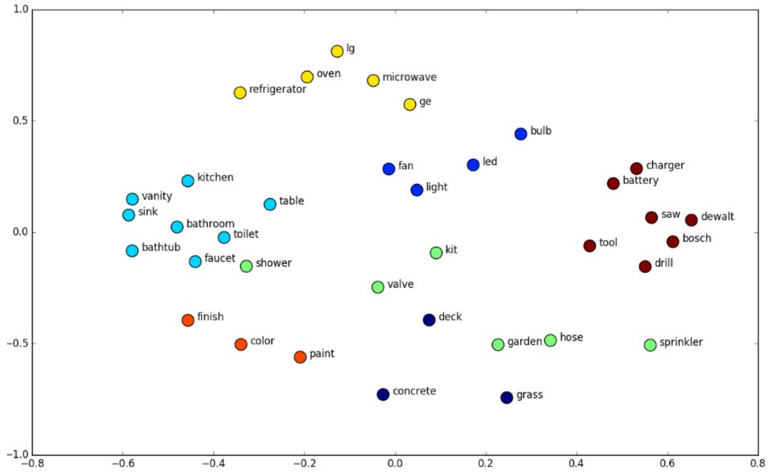
- The Distributional Hypothesis: “a word is characterized by the company it keeps”
  - words that are used and occur in the same contexts tend to purport similar meanings
- Distributed representations (i.e., embeddings)
  - The representation of any text unit is distributed over all vector dimensions 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

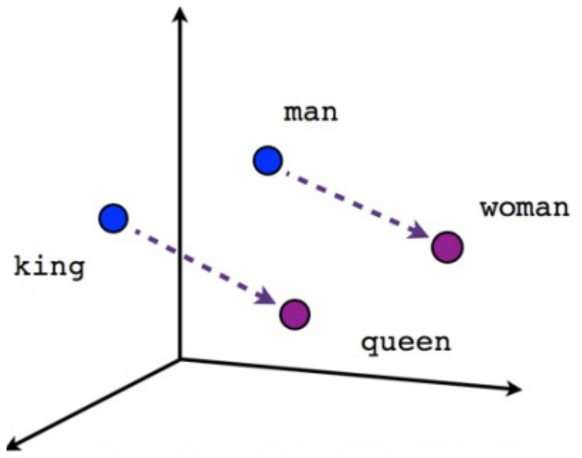
- Introduction to text representations
- Context-free embeddings
- Deep contextualized embeddings via neural language models

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)



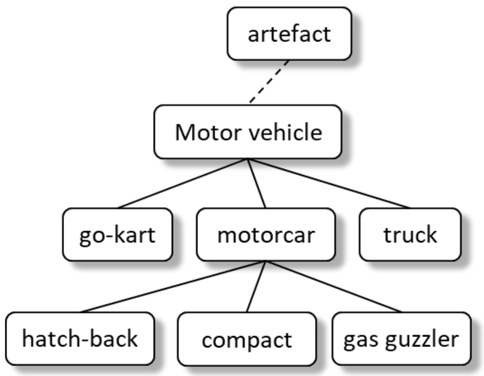
Word Similarity



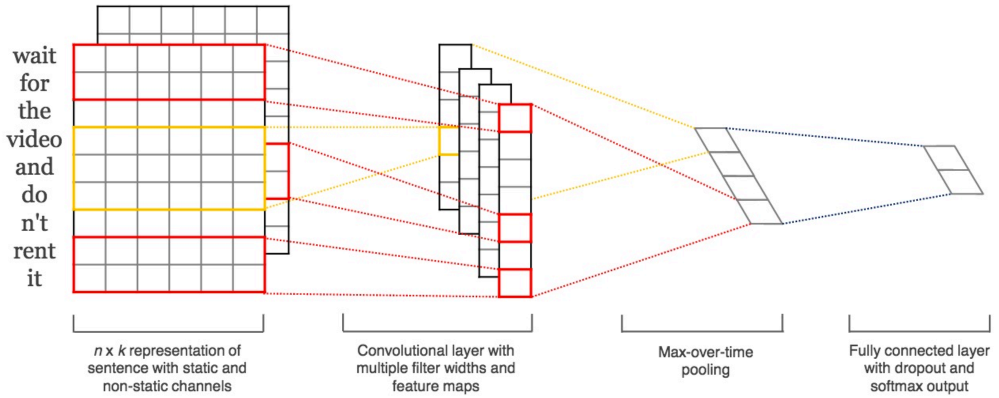
Word Analogy

Applications of Text Embeddings

- Text embeddings can be used in a lot of downstream applications
  - Word/token/entity-level tasks
    - Keyword extraction/clustering
    - Taxonomy construction
  - Document/paragraph-level tasks
    - Document classification/clustering/retrieval
    - Question answering/text summarization



Taxonomy Construction



Document Classification

Outline

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

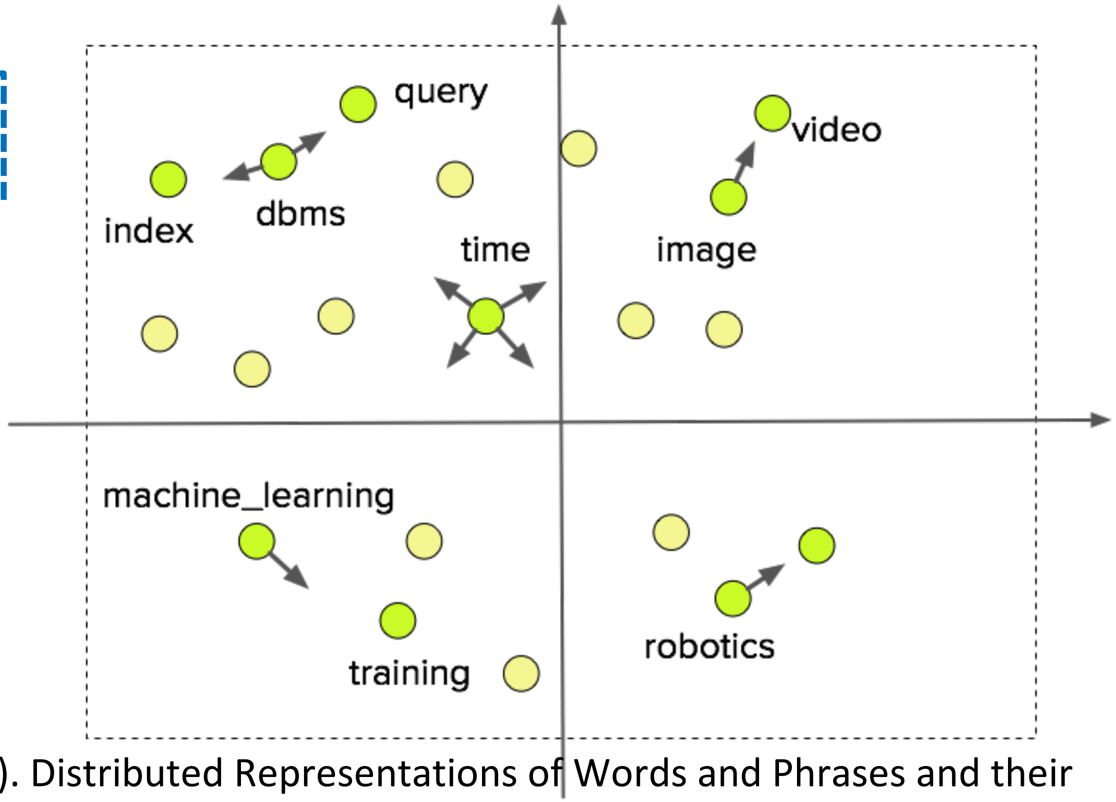
Word2Vec

- 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 coherent terms are more likely to have close embeddings

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

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

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

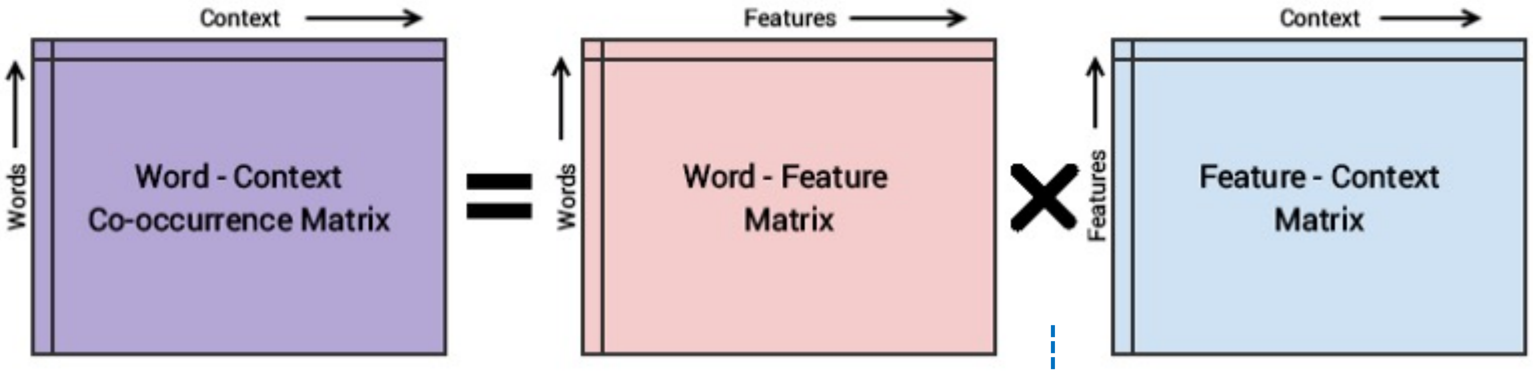


Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS.

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

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



Sparse, high dimensional

Low dimensional representation

Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.

fastText

- fastText improves upon Word2Vec by incorporating subword information into word embedding

Tri-gram extraction

<where> $\Rightarrow$ <wh, whe, her, ere, re>

- fastText allows sharing subword representations across words, since words are represented by the aggregation of their n-grams

Word2Vec probability expression

$$p(w_O|w_I) = \frac{\exp\left(\boxed{{v'_{w_O}}^\top v_{w_I}}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)} \rightarrow \sum_{g \in \mathcal{G}_w} \boxed{\mathbf{z}_g^\top} \mathbf{v}_c.$$

Represent a word by the sum of the vector representations of its n-grams

N-gram embedding

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.

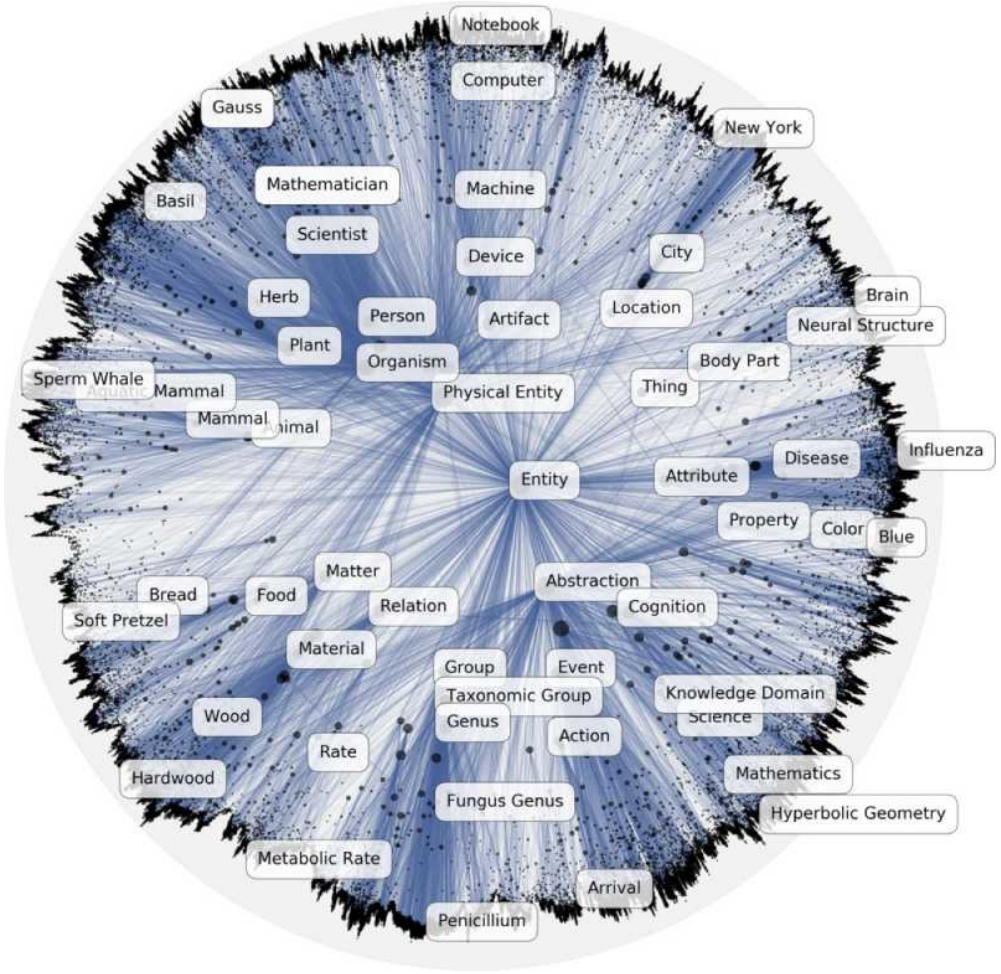
Outline

- Introduction to text representations
- Context-free embeddings
  - Euclidean space: Word2Vec, GloVe, fastText
  - 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

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



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/specificity

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

Hyperbolic metric

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

General

Specific

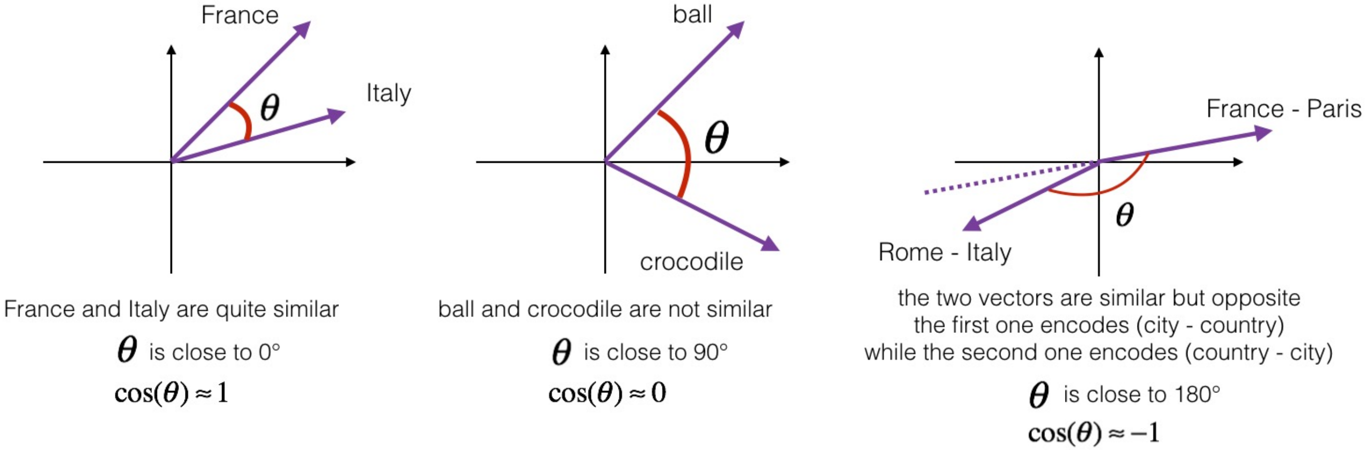
Tifrea, A., Bécigneul, G., & Ganea, O. (2019). Poincaré GloVe: Hyperbolic Word Embeddings. ICLR.

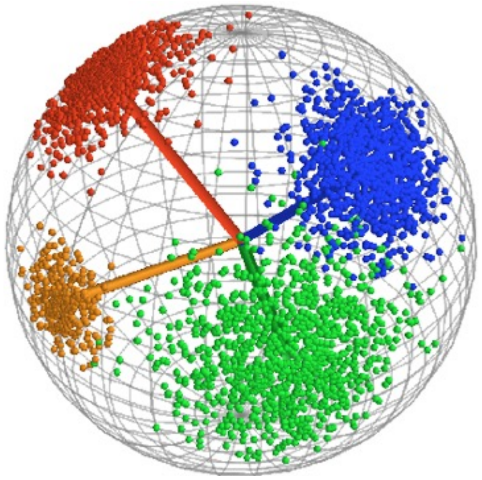
Outline

- Introduction to text representations
- Context-free embeddings
  - Euclidean space: Word2Vec, GloVe, fastText
  - 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

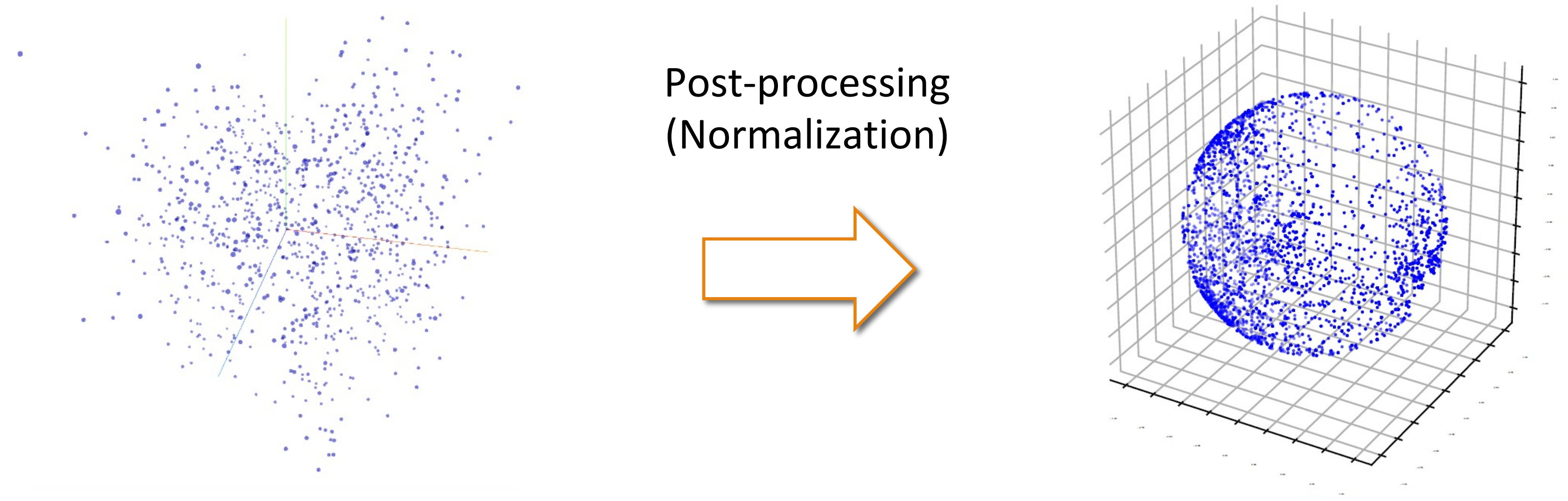




  - Better clustering performances when embeddings are normalized, and spherical clustering algorithms are used (Spherical K-means)
  - Vector direction is what actually matters!

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

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

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

Word2Vec

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

GloVe

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

fastText

Spherical Text Embedding: Generative Model

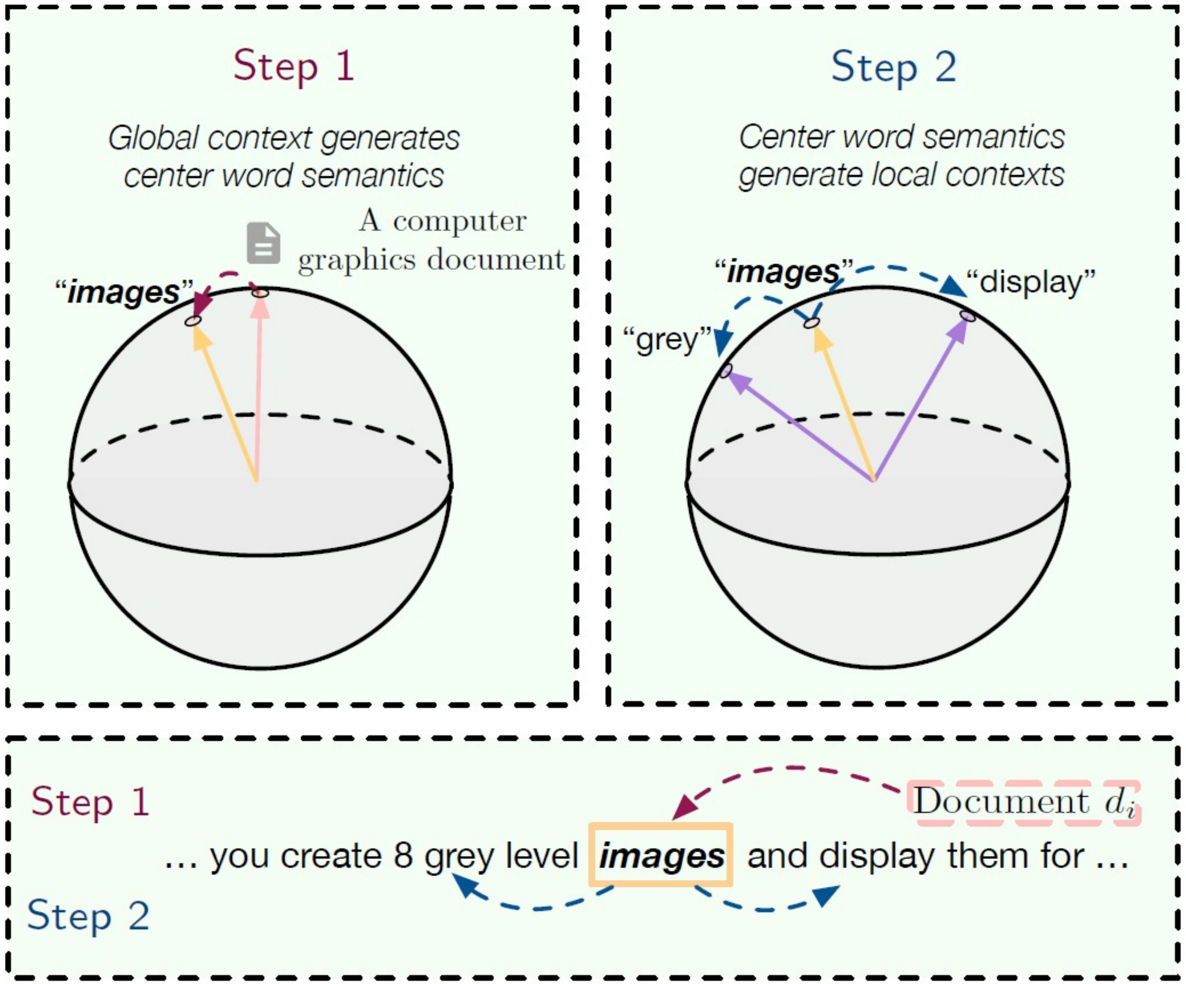
- We design a generative model on the sphere that follows how humans write articles:
  - We first have a general idea of the paragraph/document, and then start to write down each word in consistent with not only the paragraph/document, but also the surrounding words
  - Assume a two-step generation process:

$$\text{Document/ Paragraph } (d) \xrightarrow{p(u \mid d) \propto \exp(\cos(\boldsymbol{u}, \boldsymbol{d}))} \text{Center Word } (u) \xrightarrow{p(v \mid u) \propto \exp(\cos(\boldsymbol{v}, \boldsymbol{u}))} \text{Surrounding Word } (v)$$

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



Spherical Text Embedding: Objective

- The final generation probability:

$$p(v, u \mid d) = p(v \mid u) \cdot p(u \mid d) = \mathrm{vMF}_p(\boldsymbol{v}; \boldsymbol{u}, 1) \cdot \mathrm{vMF}_p(\boldsymbol{u}; \boldsymbol{d}, 1)$$

- Maximize the log-probability of a real co-occurred tuple (v, u, d), while minimize that of a negative sample (v, u', d), with a max-margin loss:

$$\begin{aligned}\mathcal{L}_{\text{joint}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{d}) = \max \Big(0, m - & \underbrace{\log \left(c_p(1)\exp(\cos(\boldsymbol{v}, \boldsymbol{u})) \cdot c_p(1)\exp(\cos(\boldsymbol{u}, \boldsymbol{d}))\right)}_{\text{Positive Sample}} \\ & + \underbrace{\log \left(c_p(1)\exp(\cos(\boldsymbol{v}, \boldsymbol{u}')) \cdot c_p(1)\exp(\cos(\boldsymbol{u}', \boldsymbol{d}))\right)}_{\text{Negative Sample}}\Big) \\ = \max \left(0, m - \cos(\boldsymbol{v}, \boldsymbol{u}) - \right. & \left. \cos(\boldsymbol{u}, \boldsymbol{d}) + \cos(\boldsymbol{v}, \boldsymbol{u}') + \cos(\boldsymbol{u}', \boldsymbol{d})\right),\end{aligned}$$

Optimization on the Sphere

- Riemannian optimization with Riemannian SGD:
  - Riemannian gradient:

$$\operatorname{grad} f(\boldsymbol{x}) := \left(I - \boldsymbol{x}\boldsymbol{x}^\top\right) \nabla f(\boldsymbol{x})$$

  - Exponential mapping (maps from the tangent plane to the sphere):

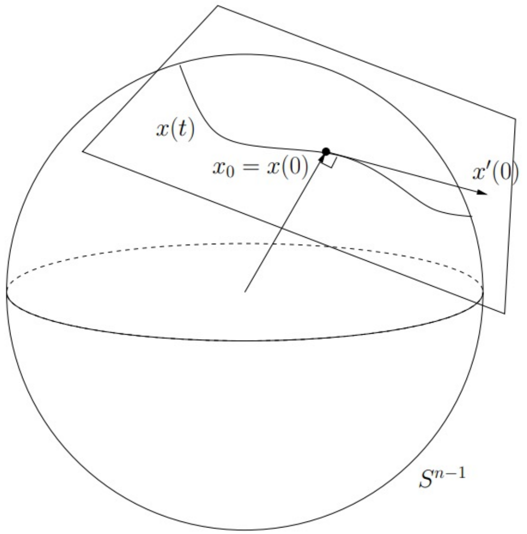
$$\exp_{\boldsymbol{x}}(\boldsymbol{z}) := \begin{cases} \cos(\|\boldsymbol{z}\|)\boldsymbol{x} + \sin(\|\boldsymbol{z}\|)\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, & \boldsymbol{z} \in T_{\boldsymbol{x}}\mathbb{S}^{p-1} \backslash \{\mathbf{0}\}, \\ \boldsymbol{x}, & \boldsymbol{z} = \mathbf{0}. \end{cases}$$

  - Riemannian SGD:

$$\boldsymbol{x}_{t+1} = \exp_{\boldsymbol{x}_t}\left(-\eta_t \operatorname{grad} f(\boldsymbol{x}_t)\right)$$

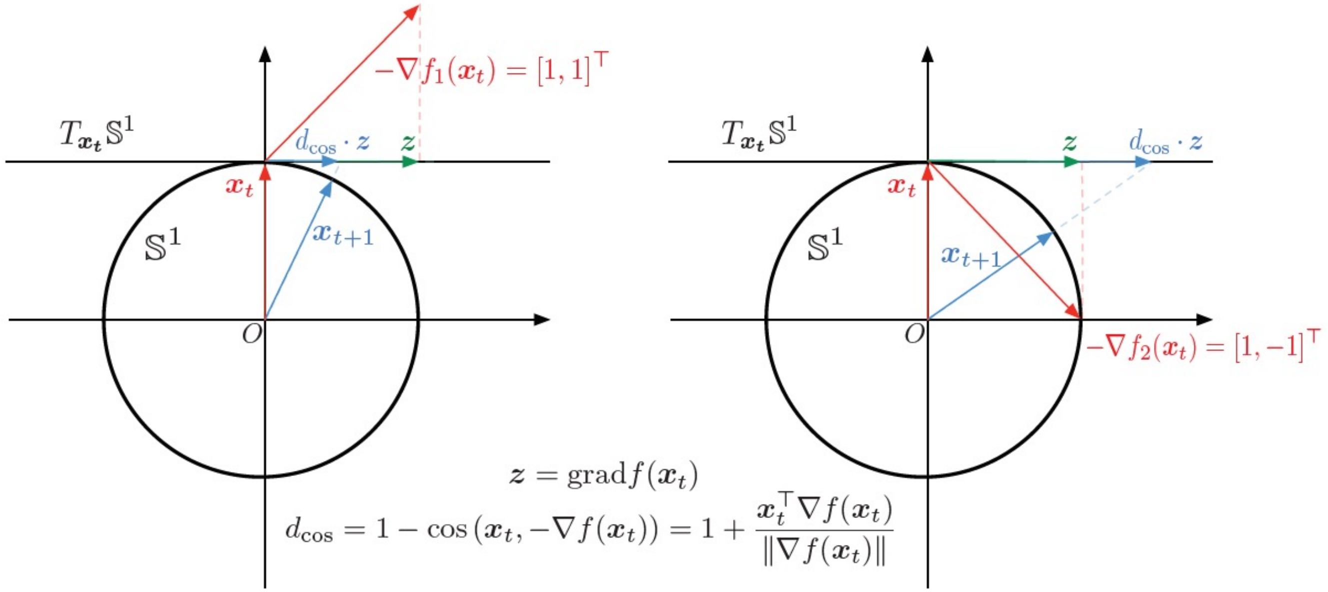
  - Retraction (first-order approximation of the exponential mapping):

$$R_{\boldsymbol{x}}(\boldsymbol{z}) := \frac{\boldsymbol{x} + \boldsymbol{z}}{\|\boldsymbol{x} + \boldsymbol{z}\|}$$



Optimization on the Sphere

- Training details:
  - Incorporate angular distances into Riemannian optimization



  - Multiply the Euclidean gradient with its angular distance from the current point

$$\boldsymbol{x}_{t+1} = R_{\boldsymbol{x}_t}\left(-\eta_t\left(1 + \frac{\boldsymbol{x}_t^\top \nabla f(\boldsymbol{x}_t)}{\|\nabla f(\boldsymbol{x}_t)\|}\right)\left(I - \boldsymbol{x}_t\boldsymbol{x}_t^\top\right)\nabla f(\boldsymbol{x}_t)\right).$$

Experiments

- Word similarity results:

Table 1: Spearman rank correlation on word similarity evaluation.

Embedding Space	Model	WordSim353	MEN	SimLex999
Euclidean	Word2Vec	0.711	0.726	0.311
	GloVe	0.598	0.690	0.321
	fastText	0.697	0.722	0.303
	BERT	0.477	0.594	0.287
Poincaré	Poincaré GloVe	0.623	0.652	0.321
Spherical	**JoSE**	**0.739**	**0.748**	**0.339**

- Why does BERT fall behind on this task?
  - BERT learns contextualized representations, but word similarity is conducted in a context-free manner
  - BERT is optimized on specific pre-training tasks like predicting masked words and sentence relationships, which have no direct relation to word similarity

Experiments

- Document clustering results:

Table 2: Document clustering evaluation on the 20 Newsgroup dataset.

Embedding	Clus. Alg.	MI	NMI	ARI	Purity
Avg. W2V	K-Means	1.299 ± 0.031	0.445 ± 0.009	0.247 ± 0.008	0.408 ± 0.014
	SK-Means	1.328 ± 0.024	0.453 ± 0.009	0.250 ± 0.008	0.419 ± 0.012
SIF	K-Means	0.893 ± 0.028	0.308 ± 0.009	0.137 ± 0.006	0.285 ± 0.011
	SK-Means	0.958 ± 0.012	0.322 ± 0.004	0.164 ± 0.004	0.331 ± 0.005
BERT	K-Means	0.719 ± 0.013	0.248 ± 0.004	0.100 ± 0.003	0.233 ± 0.005
	SK-Means	0.854 ± 0.022	0.289 ± 0.008	0.127 ± 0.003	0.281 ± 0.010
Doc2Vec	K-Means	1.856 ± 0.020	0.626 ± 0.006	0.469 ± 0.015	0.640 ± 0.016
	SK-Means	1.876 ± 0.020	0.630 ± 0.007	0.494 ± 0.012	0.648 ± 0.017
JoSE	K-Means	1.975 ± 0.026	0.663 ± 0.008	0.556 ± 0.018	0.711 ± 0.020
	SK-Means	$\mathbf{1.982} \pm 0.034$	$\mathbf{0.664} \pm 0.010$	$\mathbf{0.568} \pm 0.020$	$\mathbf{0.721} \pm 0.029$

- Embedding quality is generally more important than clustering algorithms:
  - Using spherical K-Means only gives marginal performance boost over K-Means
  - JoSE embedding remains optimal regardless of clustering algorithms

Outline

- Introduction to text representations
- Context-free embeddings
- Deep contextualized embeddings via neural language models

From Context-Free Embedding to Contextualized Embedding

- Previous unsupervised word embeddings like Word2Vec and GloVe learn **context-free** 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

“Open a **bank** account”

“On the river **bank**”

Share representation

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

Outline

- Introduction to text representations
- Context-free embeddings
- Deep contextualized embeddings via neural language models
  - Language Model Pre-Training
  - Language Model Deployment

Pre-Training Deep Language Models

- The "pretrain-finetune" paradigm has become the prominent practice in a wide variety of text applications
- First pre-train language models (PLMs, often based on the Transformer architecture) via self-supervised objectives on large-scale general-domain corpora, then fine-tune them on task-specific data
- Based on the pre-training objective/task, PLMs can be generally categorized into two types:
  - Unidirectional (or autoregressive) PLM: Predict the next token based on all previous tokens, leveraging single-directional (left-to-right) contexts (e.g., GPT)
  - Bidirectional (or autoencoding) PLM: Predict masked/corrupted tokens based on all other (uncorrupted) tokens, leveraging bidirectional contexts (e.g., BERT, XLNet, ELECTRA)

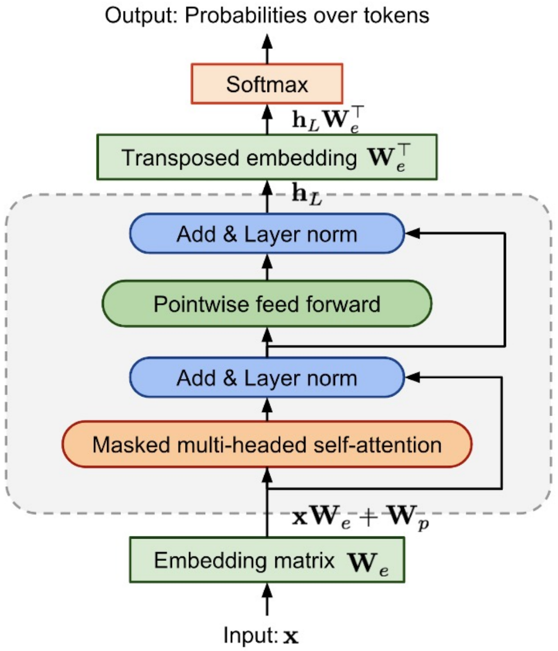
GPT-Style Pre-Training: Introduction

- Generative Pre-Training (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

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

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



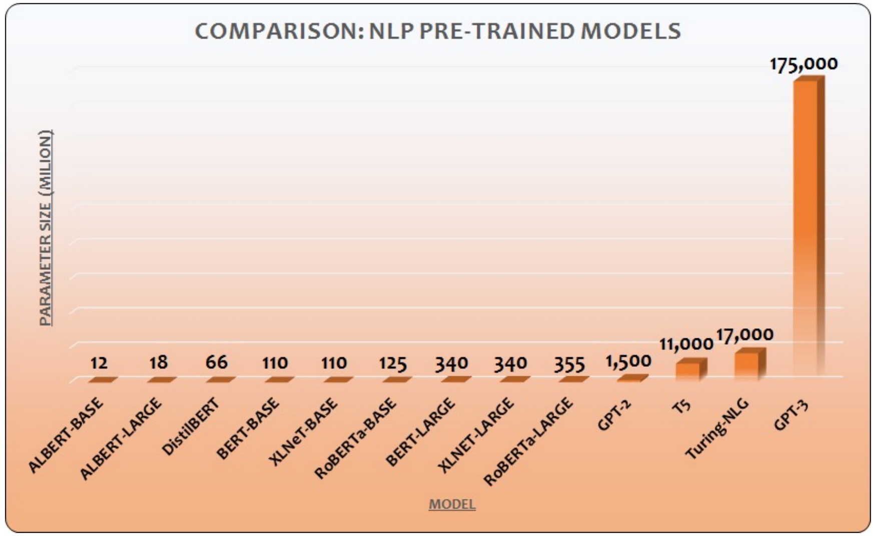
[1] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog

[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.

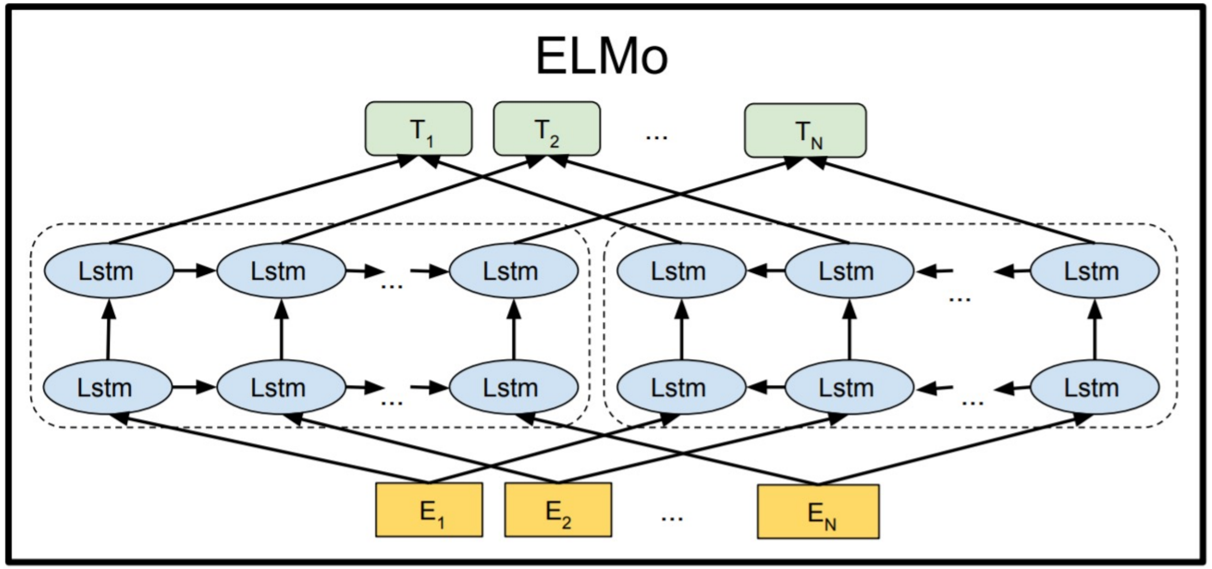
GPT-Style Pre-Training: Text Generation

- 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/



ELMo: Deep contextualized word representations

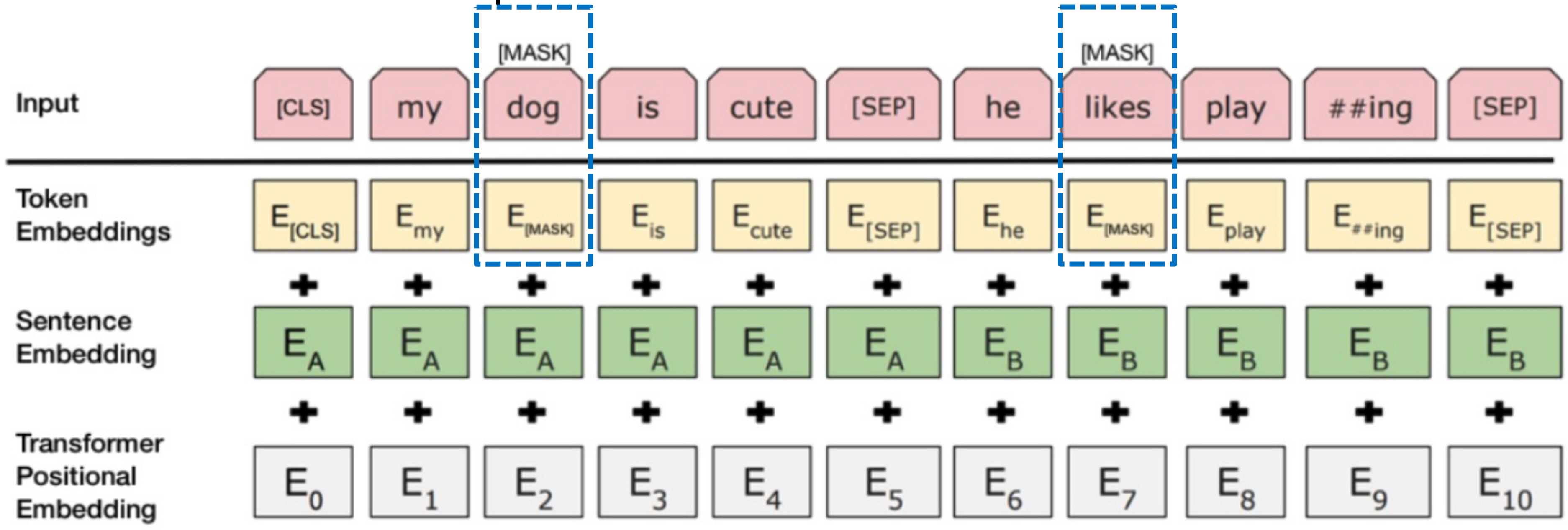
- Word representations are learned functions of the internal states of a deep bi-directional LSTMs
- Results in a pre-trained network that benefits several downstream tasks (e.g., Sentiment analysis, Named entity extraction, Question answering)
- However, left-to-right and right-to-left LSTMs are **independently** trained and concatenated



Peters, M.E., Neumann, M., Iyyer, M., Gardner, M.P., Clark, C., Lee, K., & Zettlemoyer, L.S. (2018). Deep contextualized word representations. NAACL.

BERT: Masked Language Modeling

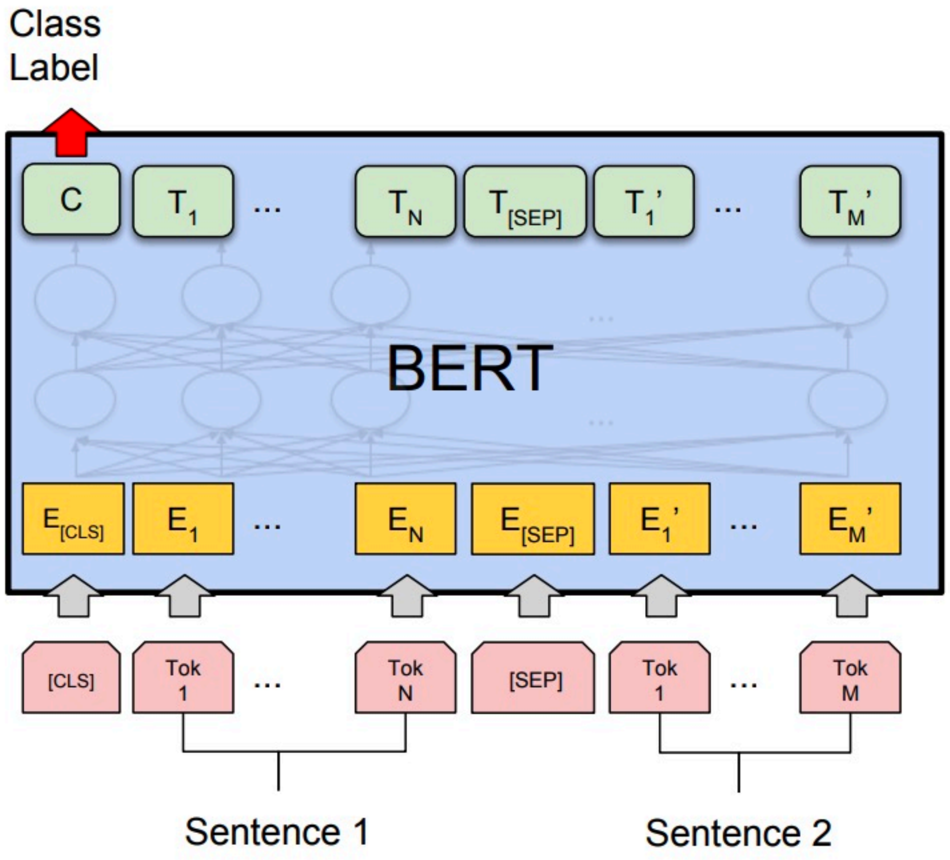
- Bidirectional: BERT leverages a Masked LM learning to introduce **real bidirectionality** training
- Masked LM: With 15% words randomly masked, the model learns bidirectional contextual information to predict the masked words



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	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
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

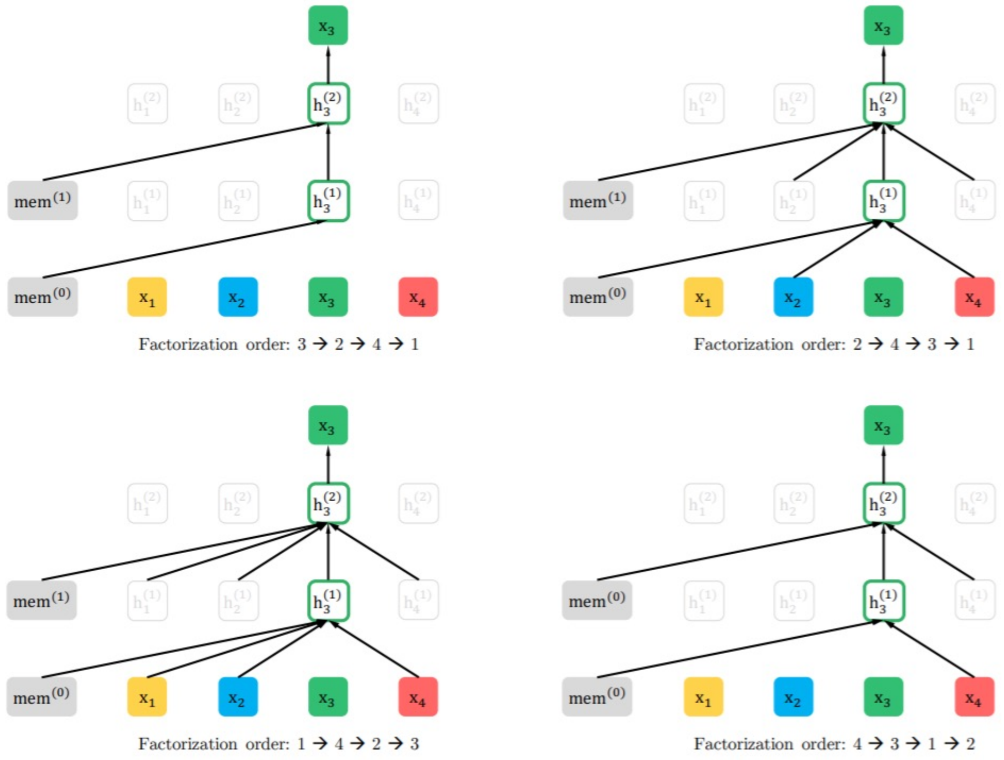
- Simple modifications that make BERT more **efficient**:
  - Factorized embedding parameterization: use lower-dimensional token embeddings; project token embeddings to hidden layer dimension
  - Cross-layer parameter sharing: Share feed-forward network parameters/attention parameters across layers
  - 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	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	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 pre-training 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

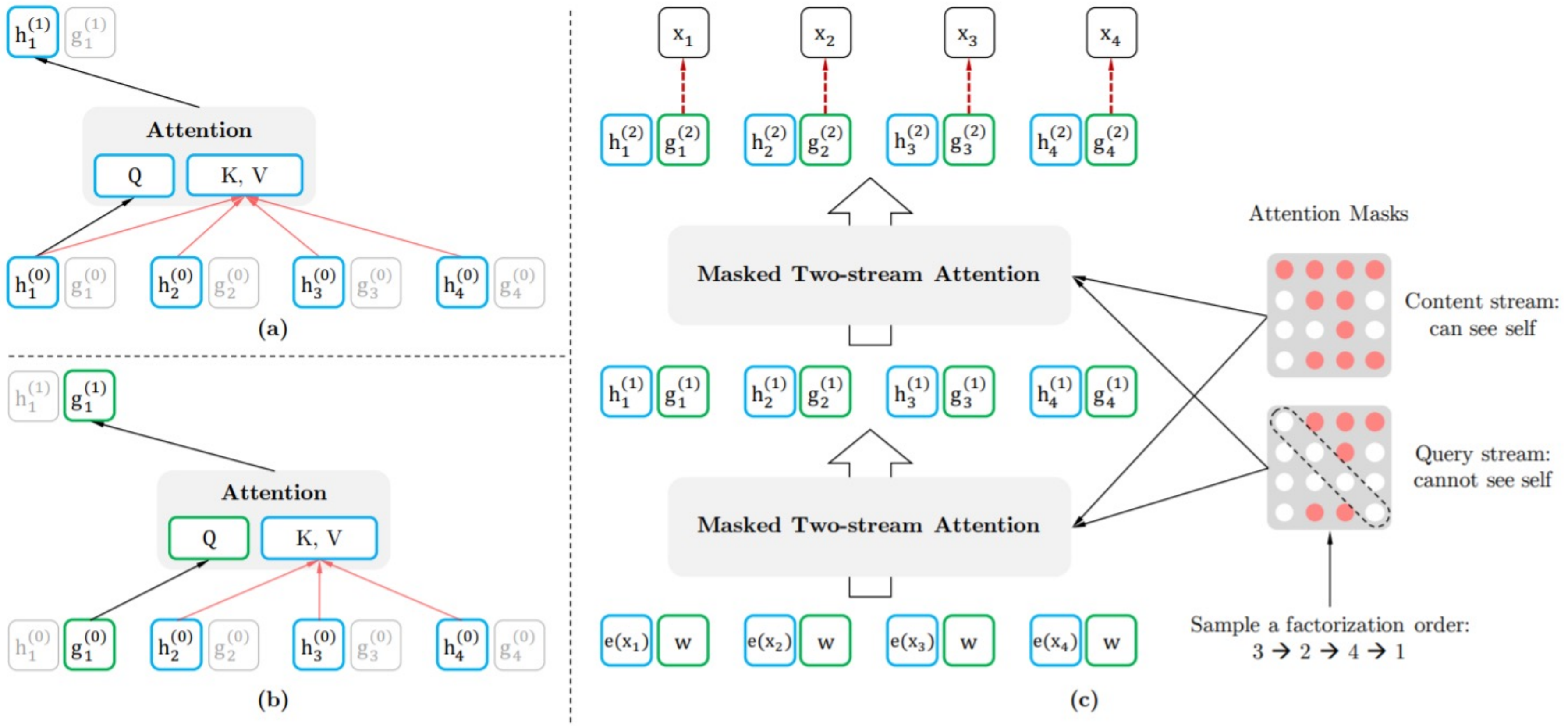
Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. NeurIPS.

XLNet: Two-Stream Self-Attention

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

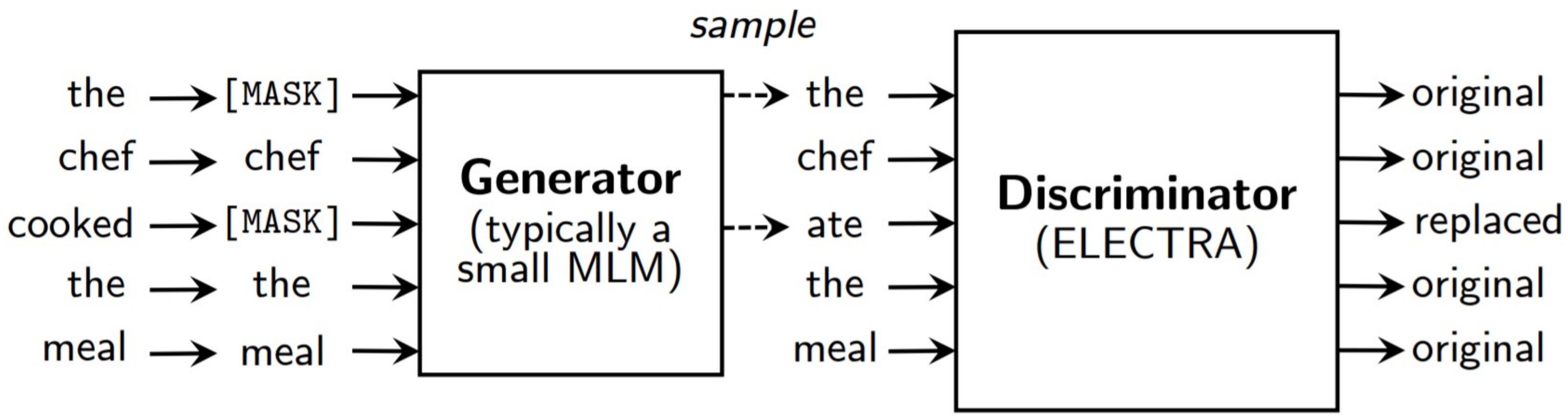
Content representation

Query representation



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)



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

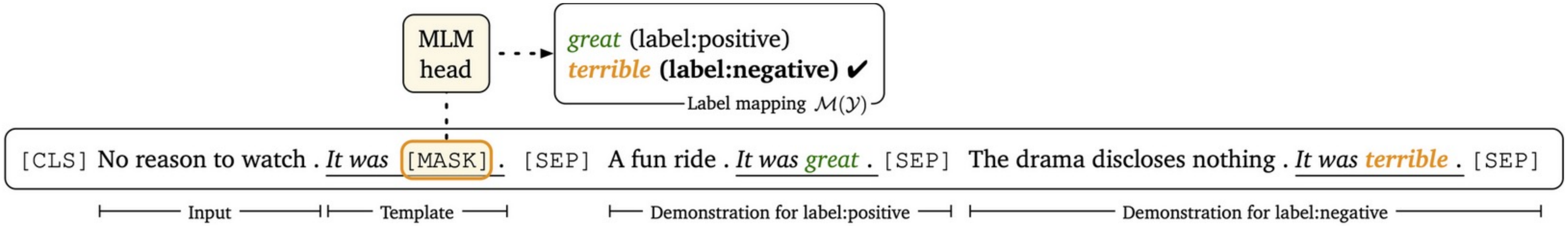
ELECTRA

- 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	**97.1**	**91.2**	92.0	90.5	**91.3**	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	**97.1**	90.5	**92.6**	90.4	90.9	–	88.5	**92.5**	89.1	–
ELECTRA	3.1e21 (1x)	**71.7**	**97.1**	90.7	92.5	**90.8**	**91.3**	**95.8**	**89.8**	**92.5**	**89.5**	**89.4**

Challenges with ELECTRA-Style Pre-Training

- 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

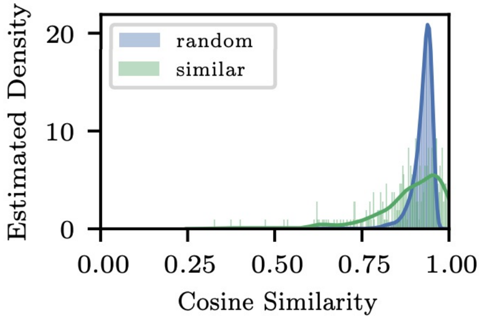


Prompt-based fine-tuning transfers the PLMs' language modeling ability to downstream tasks

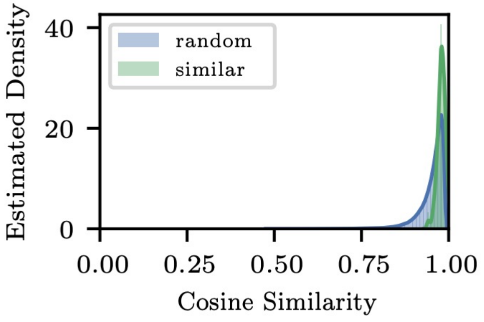
Challenges with ELECTRA-Style Pre-Training

- 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:

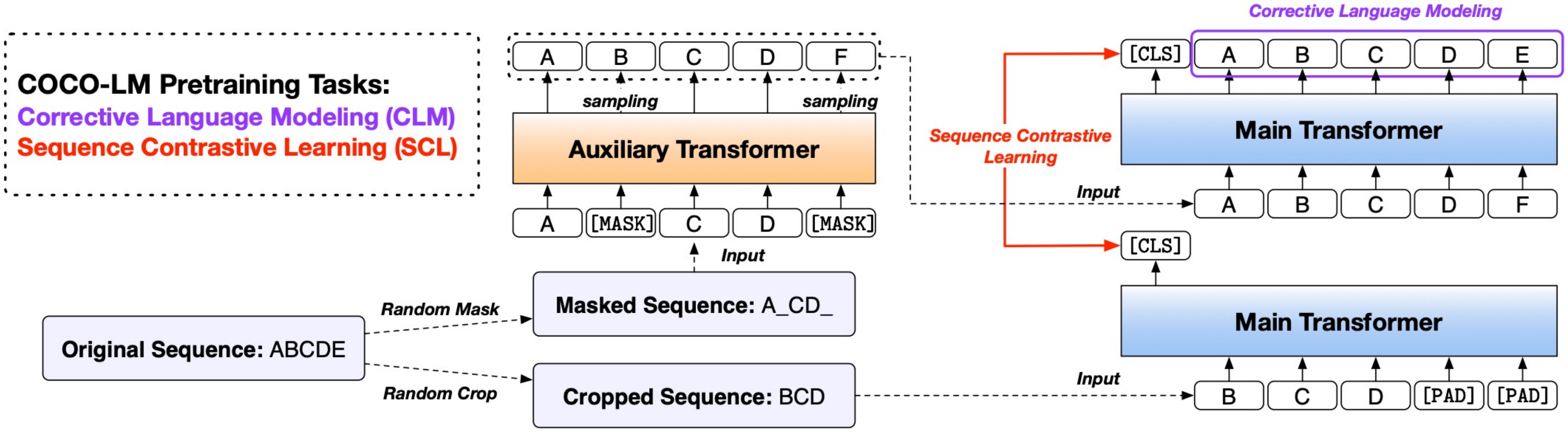


ELECTRA sequence embedding space:



COCO-LM: Method

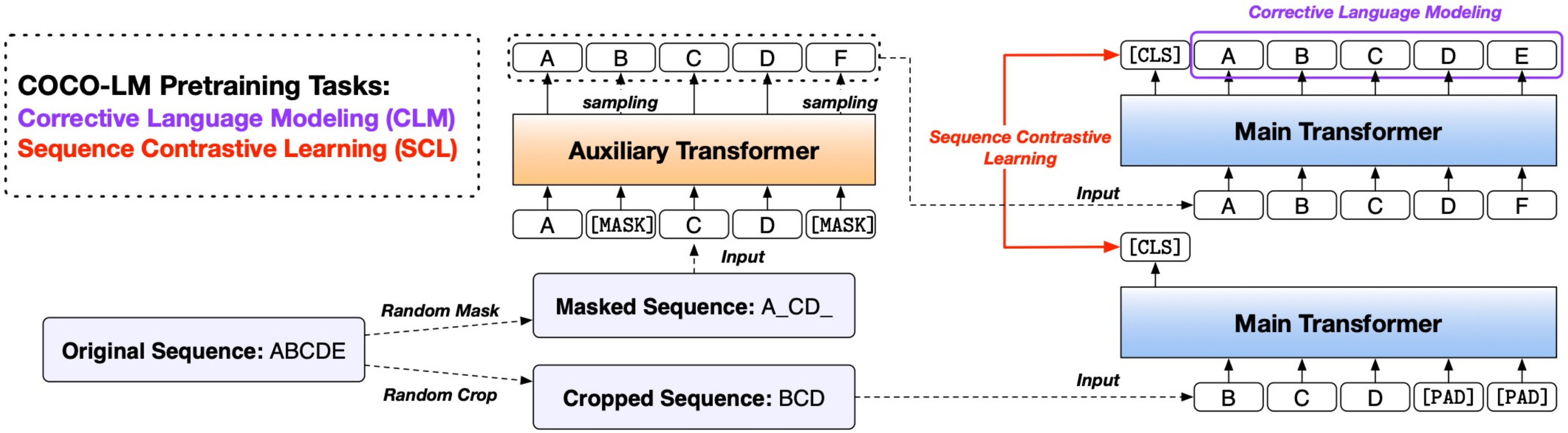
- COCO-LM has two new pre-training tasks upon the corrupted sequences that address the challenges in ELECTRA-style pretraining
  - Corrective Language Modeling (CLM)
  - Sequence Contrastive Learning (SCL)



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: Method

- Corrective Language Modeling (CLM) trains the main Transformer to recover the original tokens
  - The main Transformer needs to not only detect replaced tokens, but also output the original ones if the tokens are replaced
- Sequence Contrastive Learning (SCL) trains the sequence embeddings (i.e., [CLS] embedding) of a positive pair to be close and negative pairs to be apart
  - Using token replaced sequence and cropped sequence as the positive pair
  - Making the sequence representations robust to token-level and sequence-level alterations



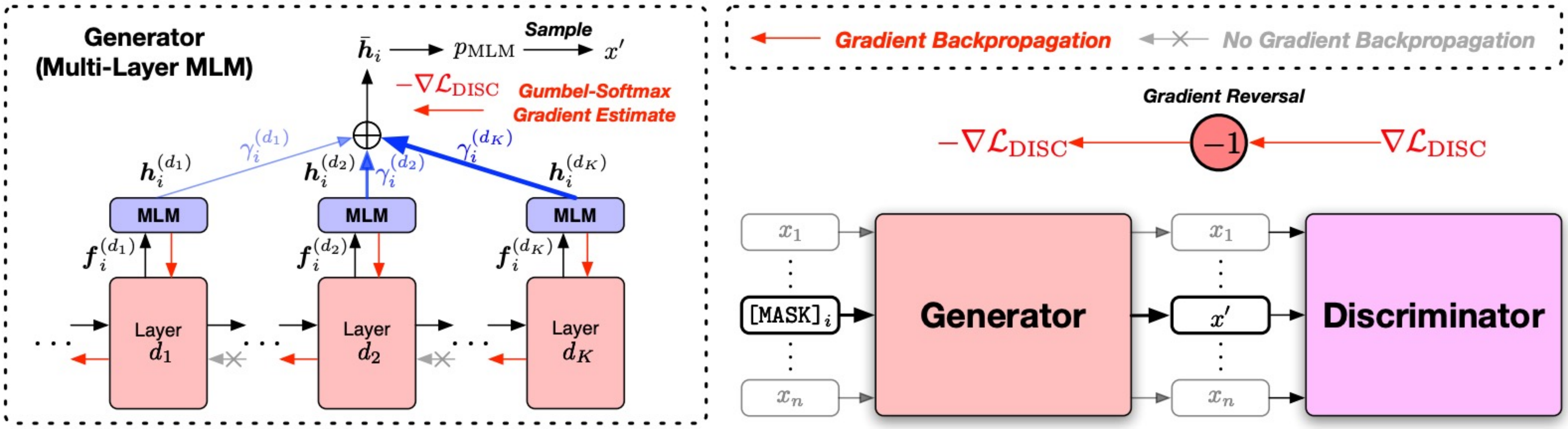
COCO-LM: Results

- 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 MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	AVG	SQuAD 2.0 DEV EM	F1
Base Setting: 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	**93.7**	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	**66.2**	75.1	88.2	89.7	85.5	79.7	82.6
COCO-LM	110M	**88.5/88.3**	**92.0**	**93.1**	93.2	63.9	**84.8**	**91.4**	**90.3**	**87.2**	**82.4**	**85.2**
Base++ Setting: 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	**95.1**	65.2	81.3	**91.8**	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	–	–
COCO-LM	134M	**90.2/90.0**	**92.2**	**94.2**	94.6	**67.3**	**87.4**	91.2	**91.8**	**88.6**	**85.4**	**88.1**
Large++ Setting: BERT Large Size, Bigger Training Data, and More Training Steps												
XLNet [62]	360M	90.8/90.8	92.3	94.9	**97.0**	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
COCO-LM	367M	**91.4/91.6**	**92.8**	**95.7**	96.9	**73.9**	**91.0**	**92.2**	**92.7**	**90.8**	**88.2**	**91.0**
$\text{Megatron}_{1.3B}$ [48]	1.3B	90.9/91.0	92.6	–	–	–	–	–	–	–	87.1	90.2
$\text{Megatron}_{3.9B}$ [48]	3.9B	91.4/91.4	92.7	–	–	–	–	–	–	–	88.5	91.2

AMOS: Adversarial Curriculum for Pre-Training

- 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 MNLI	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	AVG	SQuAD 2.0 EM	F1
Base Setting: 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_{Loss}+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	**91.4**	90.3	87.2	82.4	85.2
AMOS	110M	**88.9/88.7**	**92.3**	**93.6**	**94.2**	**70.7**	**86.6**	90.9	**91.6**	**88.6**	**84.2**	**87.2**
Base++ Setting: 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	**91.8**	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	**87.4**	91.2	91.8	88.6	**85.4**	**88.1**
AMOS	134M	**90.5/90.4**	**92.4**	**94.4**	**95.5**	**71.8**	86.6	91.7	**92.0**	**89.4**	85.0	87.9

Outline

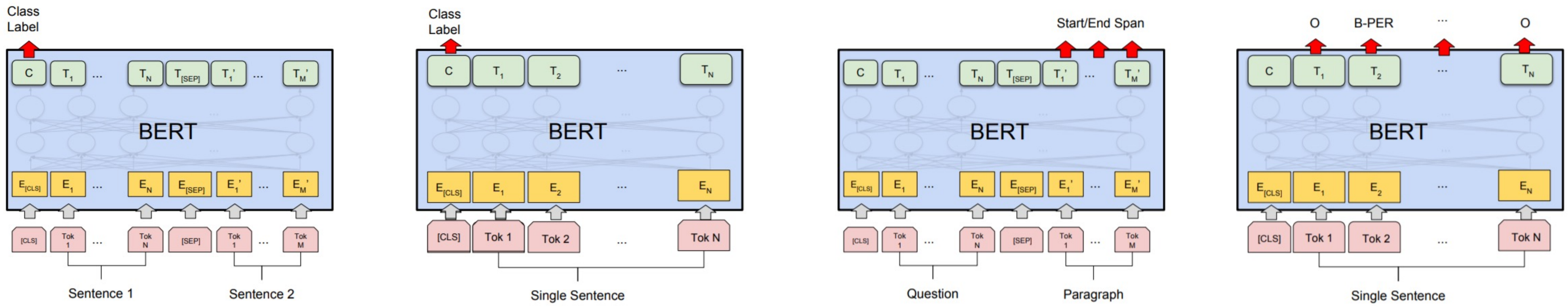
- Introduction to text representations
- Context-free embeddings
- Deep contextualized embeddings via neural language models
  - Language Model Pre-Training
  - Language Model Deployment

Deployment of Pre-Trained Language Models

- 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
  - Parameter-efficient tuning: Only update a small portion of PLM parameters and keep other (majority) parameters unchanged
  - Prompt-based inference: Directly use PLMs to make predictions on cloze-type token prediction tasks without parameter updates

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

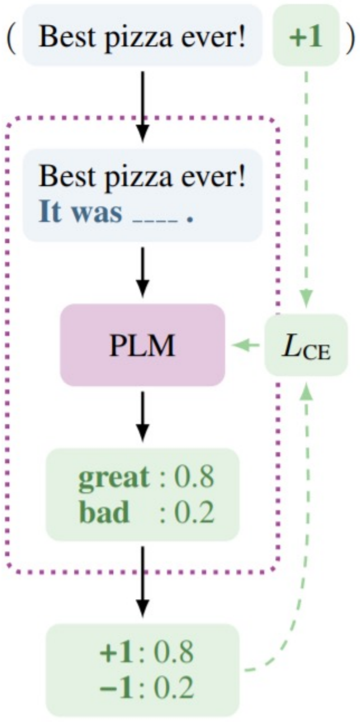
(b) Single Sentence Classification Tasks: SST-2, CoLA

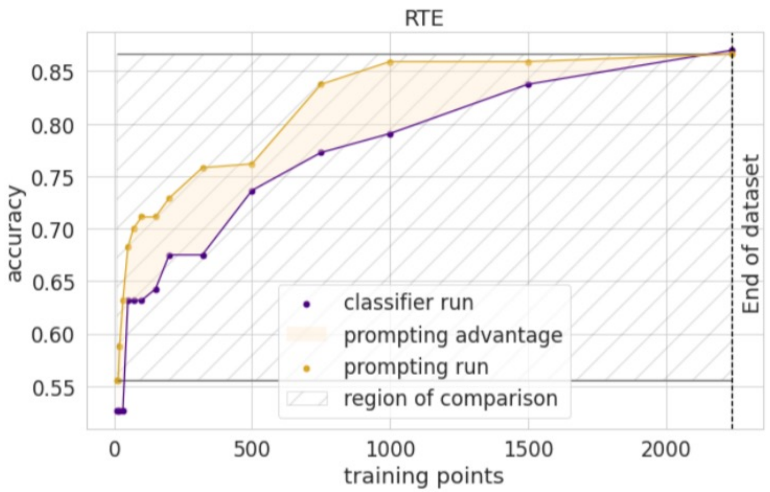
(c) Question Answering Tasks: SQuAD v1.1

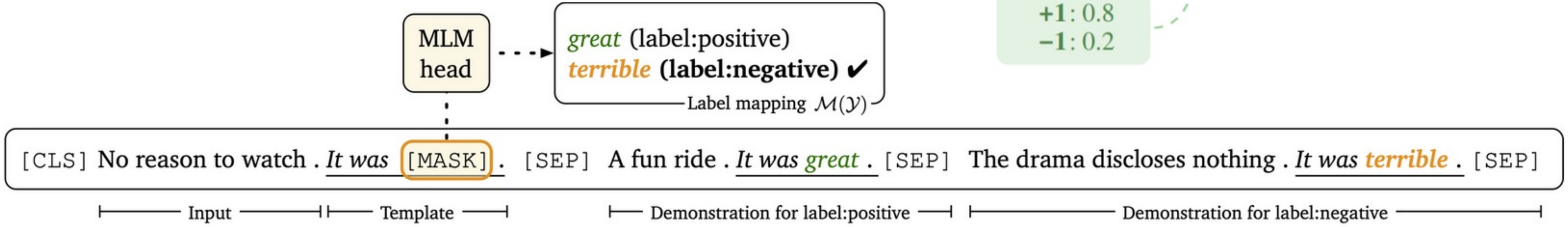
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

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†	*50.9*	*23.1*	*50.0*	*50.0*	*50.0*	*50.0*	*18.8*	*0.0*
Prompt-based zero-shot‡	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)	**33.9** (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)	**50.6** (1.4)	86.6 (2.2)	90.2 (1.2)	**87.0** (1.1)	**92.3** (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	**93.0** (0.6)	49.5 (1.7)	**87.7** (1.4)	**91.0** (0.9)	86.5 (2.6)	91.4 (1.8)	**89.4** (1.7)	21.8 (15.9)
Fine-tuning (full)†	*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†	*32.7*	*33.0*	*33.8*	*49.5*	*52.7*	*81.2*	*0.0*	-
Prompt-based zero-shot‡	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	**70.7** (1.3)	**72.0** (1.2)	**79.7** (1.5)	**69.2** (1.9)	68.7 (2.3)	77.8 (2.0)	**69.8** (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)	**73.9** (2.2)	76.2 (2.3)	67.0 (3.0)	75.0 (3.3)
+ demonstrations	70.0 (3.6)	**72.0** (3.1)	77.5 (3.5)	68.5 (5.4)	71.1 (5.3)	**78.1** (3.4)	67.7 (5.8)	**76.4** (6.2)
Fine-tuning (full)†	*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

Zero-Shot Fine-Tuning of PLMs

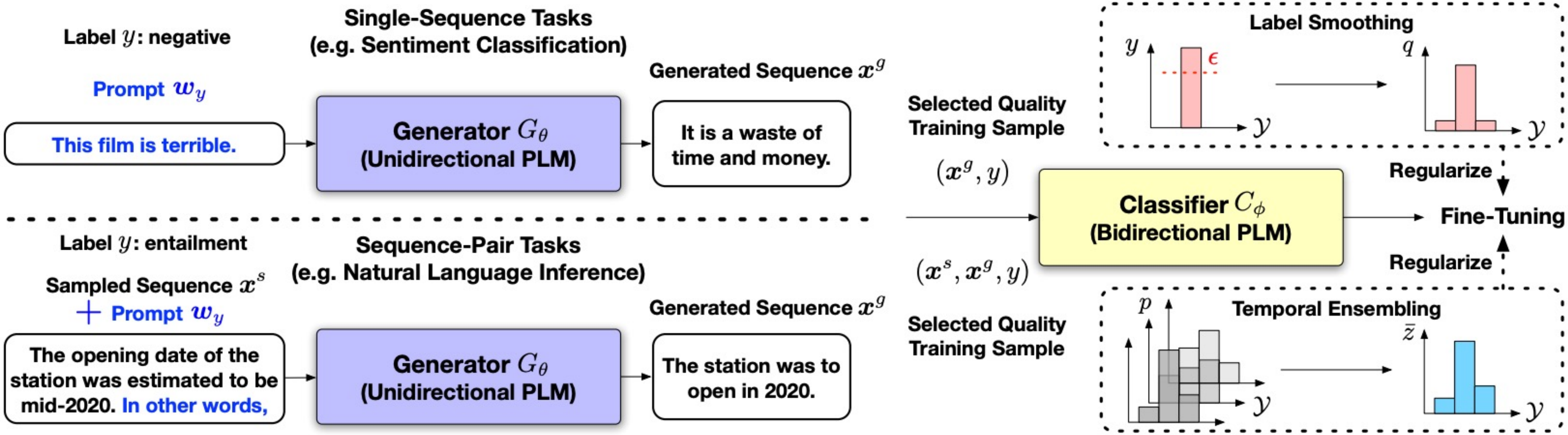
- Prompt-based approaches have remarkable few-shot fine-tuning performance, but their zero-shot performance is significantly worse
- Without any task-specific samples, it is challenging for PLMs to interpret the prompts that come in different formats and are unseen in the pretraining data
- The current 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

Generating Training Data with PLMs

- SuperGen: A **Super**vision **Gen**eration 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 Fine-Tuning 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)

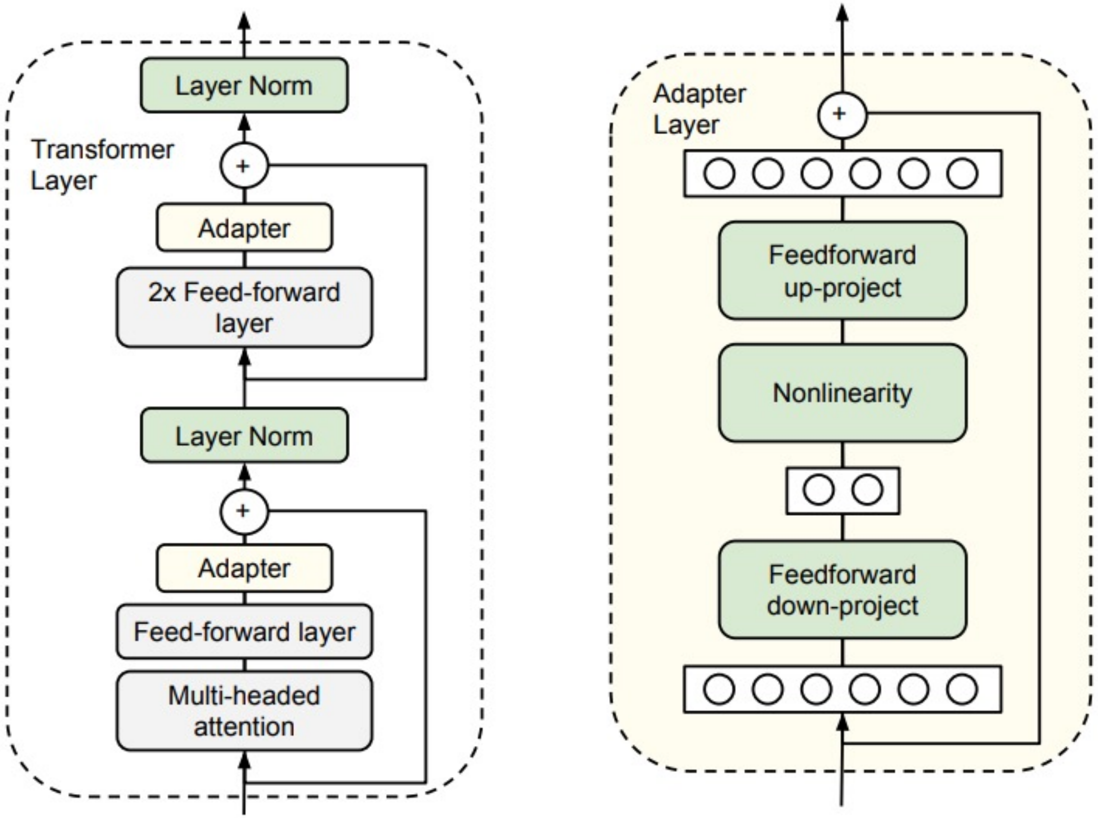
Method	MNLI-(m/mm) (Acc.)	QQP (F1)	QNLI (Acc.)	SST-2 (Acc.)	CoLA (Matt.)	RTE (Acc.)	MRPC (F1)	AVG
Zero-Shot Setting: No task-specific data (neither labeled nor unlabeled).								
Prompting†	$50.8_{0.0}/51.7_{0.0}$	$49.7_{0.0}$	$50.8_{0.0}$	$83.6_{0.0}$	$2.0_{0.0}$	$51.3_{0.0}$	$61.9_{0.0}$	50.1
SuperGen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

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

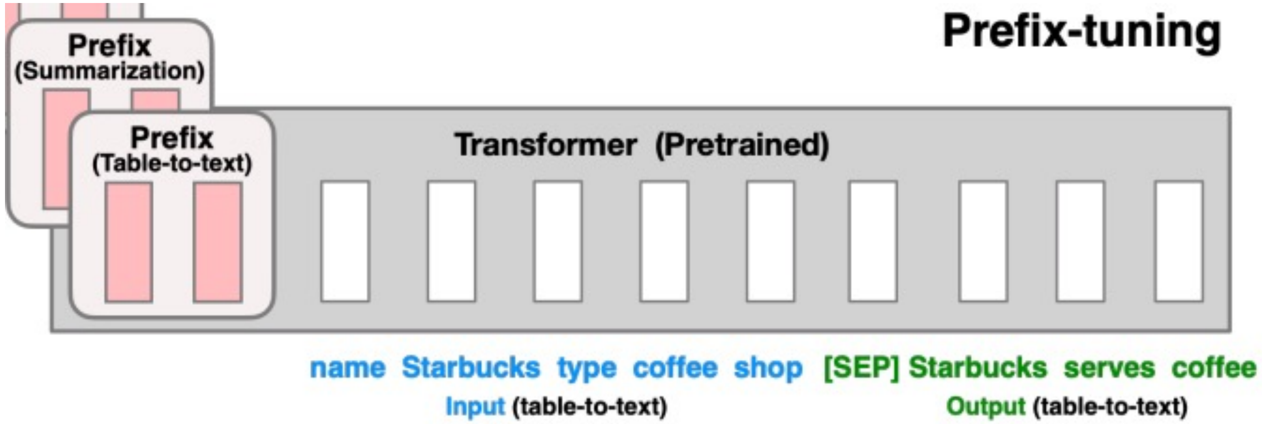
- Adapters are added twice to each Transformer layer
- Consist of a bottleneck structure (down-project + up-project)
- Only adapter parameters + layer norm parameters are updated during tuning



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

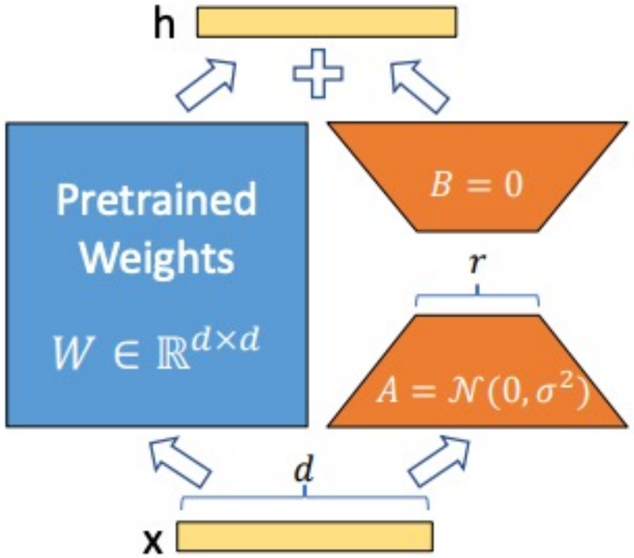
- Prefix tuning prepends trainable vectors to each Transformer layer
- Only update prefix vectors and keep other pretrained parameters unchanged
- Similar to prompt-based fine-tuning except that the prefix vectors are continuous parameters instead of natural language words



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



$$W_0 + \Delta W = W_0 + \boxed{BA}$$

A and B are low-rank matrices

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.

Prompt-Based Inference Without Tuning

- 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

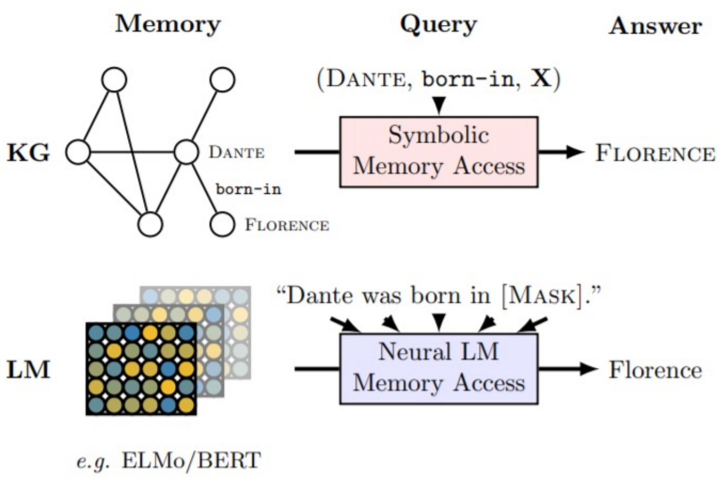
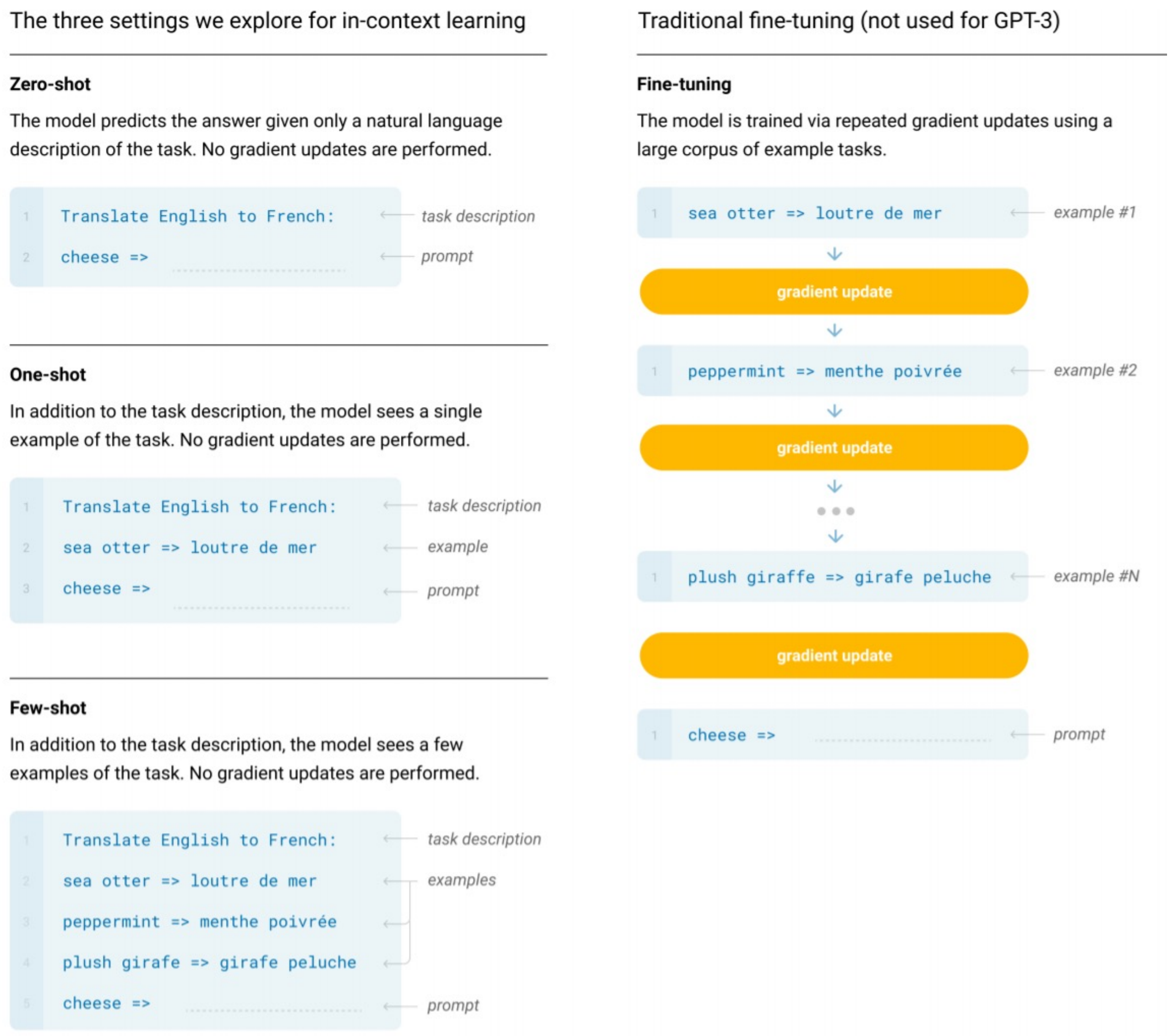


Figure 1: Querying knowledge bases (KB) and language models (LM) for factual knowledge.

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

Prompt-Based Inference Without Tuning

- 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



References I

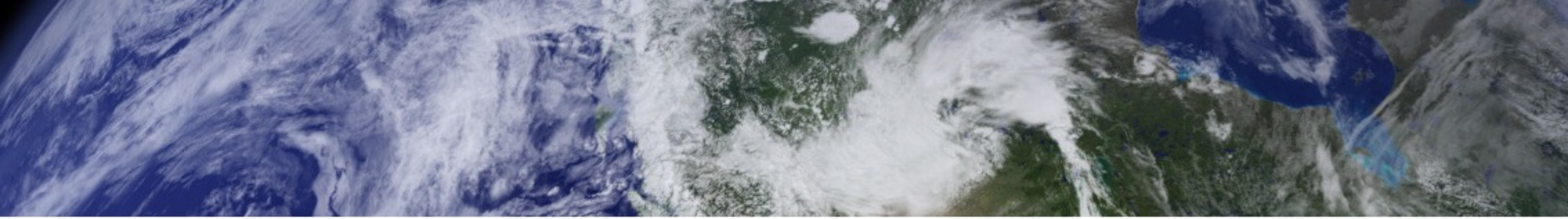
- Abu-El-Haija, S., Perozzi, B., Al-Rfou', R., & Alemi, A.A. (2018). Watch Your Step: Learning Node Embeddings via Graph Attention. NeurIPS.
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational 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

- 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.
- 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