HIT-SCIR

NLPCC 2020 Zhengzhou, China October, 14-18

Revisiting Pre-Trained Models for Natural Language Processing

Yiming Cui

Joint Laboratory of HIT and iFLYTEK Research (HFL)

October 14, 2020

Roadmap

- Before Tutorial
- Introduction
- Traditional Approaches for Text Representations
    - word2vec, GloVe
- Contextualized Language Models
    - CoVe, ELMo
- Deep Contextualized Language Models
    - GPT, BERT, XLNet, RoBERTa, ALBERT, ELECTRA

Roadmap

- Chinese Pre-trained Language Models
  - Chinese BERT-wwm, ERNIE, NEZHA, ZEN
  - **MacBERT**
- Recent Research on PLM
  - Trending: GPT-2, GPT-3, T5
  - Distillation: DistilBERT, TinyBERT, MobileBERT, **TextBrewer**
  - Multi-lingual: mBERT, XLM, XLM-R
- Summary

Source Code

Paper / Resource QR

Special Tips

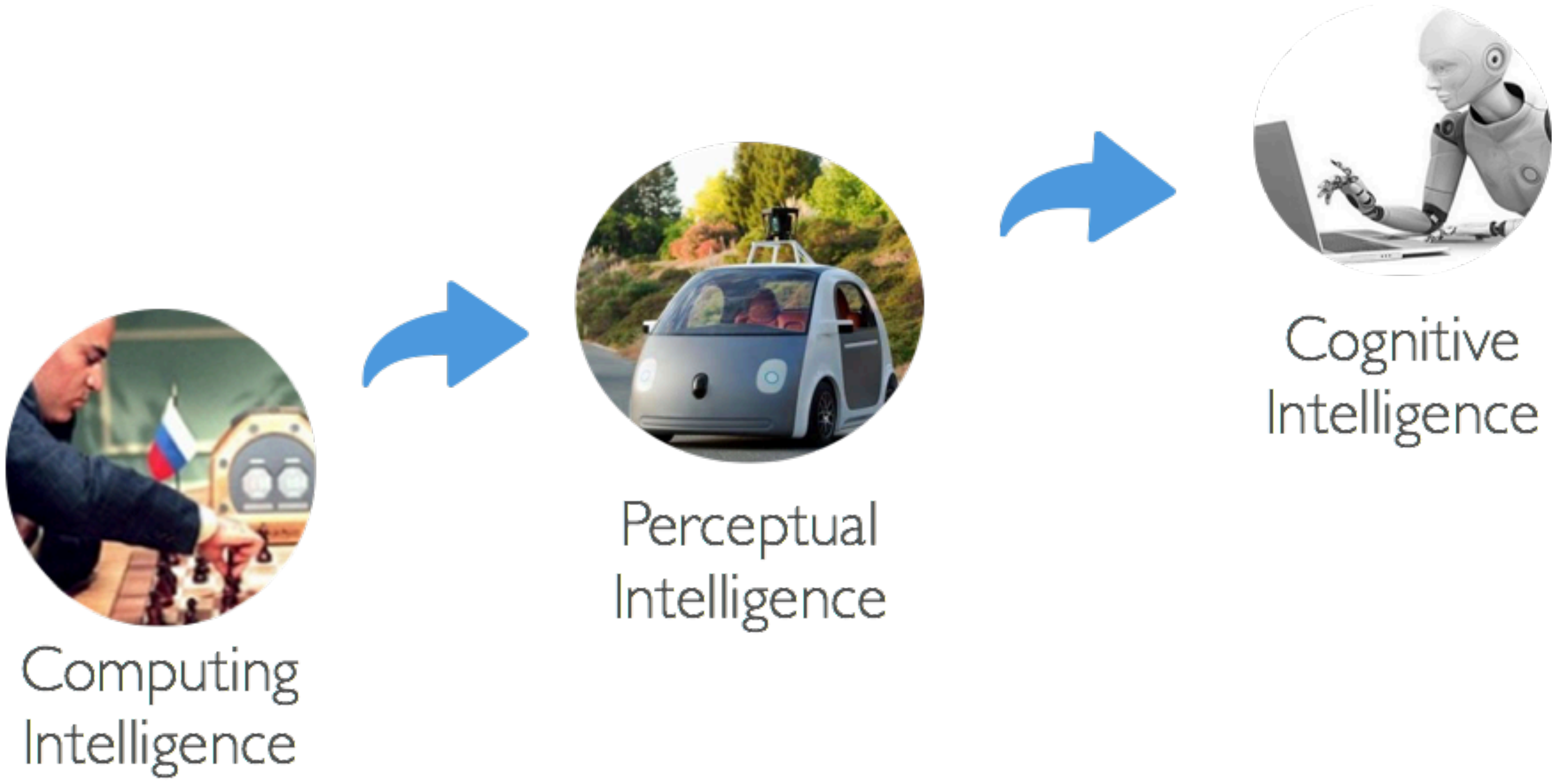
Open Source

简介

INTRODUCTION

Introduction

- **NLP is One of The Most Challenging Tasks in A.I.**
    - Understanding natural language is key to achieve strong A.I.



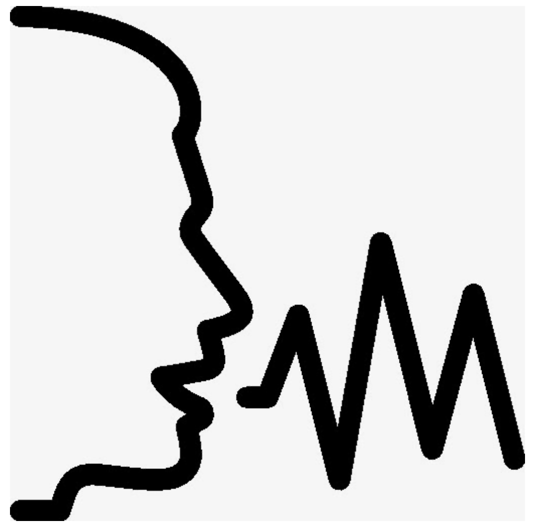
Introduction

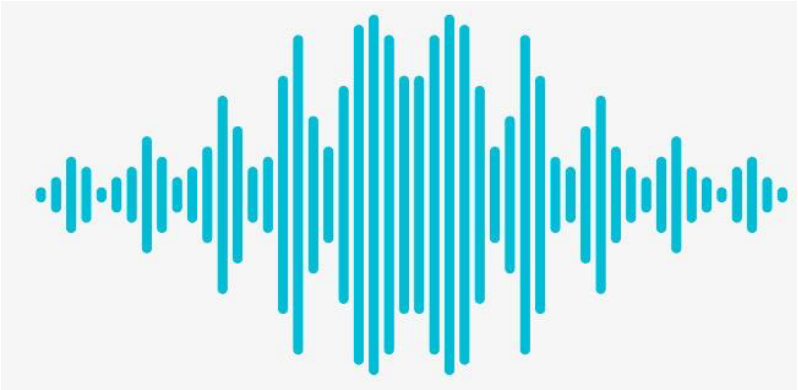
- **Why NLP is Hard?**

Vision

Pixels

Speech





Waves

Language

?

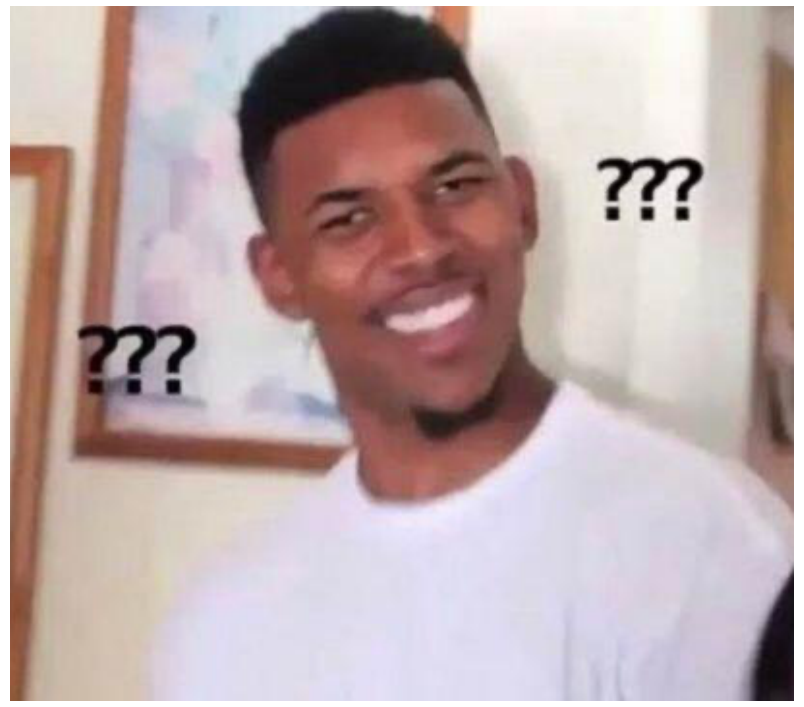
Introduction

- **Why NLP is Hard?**
  - Language is highly **abstracted** without determined physical representation
  - Requires deep understanding and sometimes needs logical inference / commonsense

他一把把把把住了

The man could't lift his son because he was so [weak/heavy]. Who was [weak/heavy]?

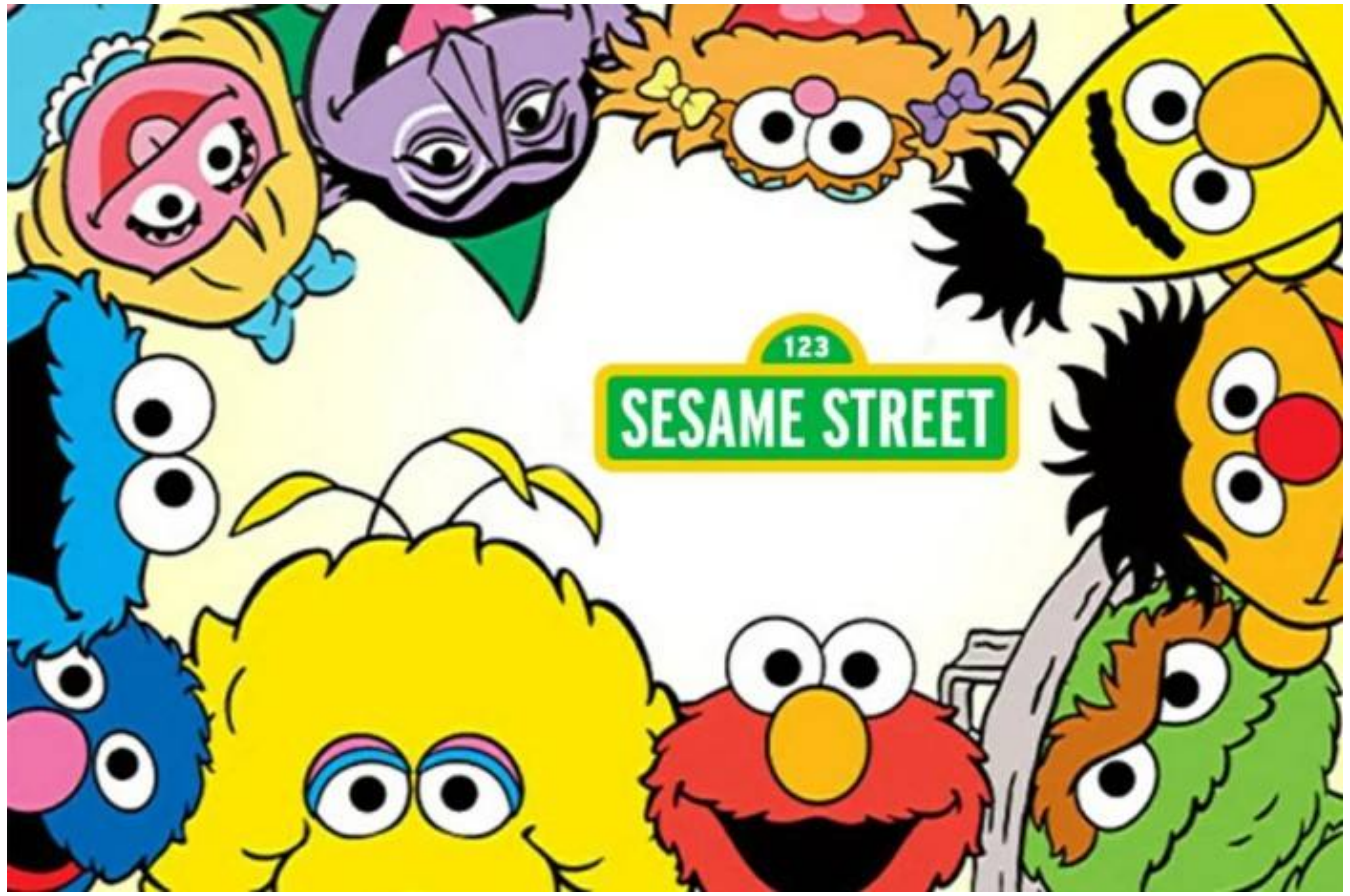
货拉拉拉不拉拉布拉多?



Learning Good Text Representations is the Foundation in NLP

Introduction

- **Pre-trained Models in NLP**
  - ‘Sesame Street’ family and OpenAI GPT family



BERT

ERNIE

Grover

BigBird

ELMo

Oscar

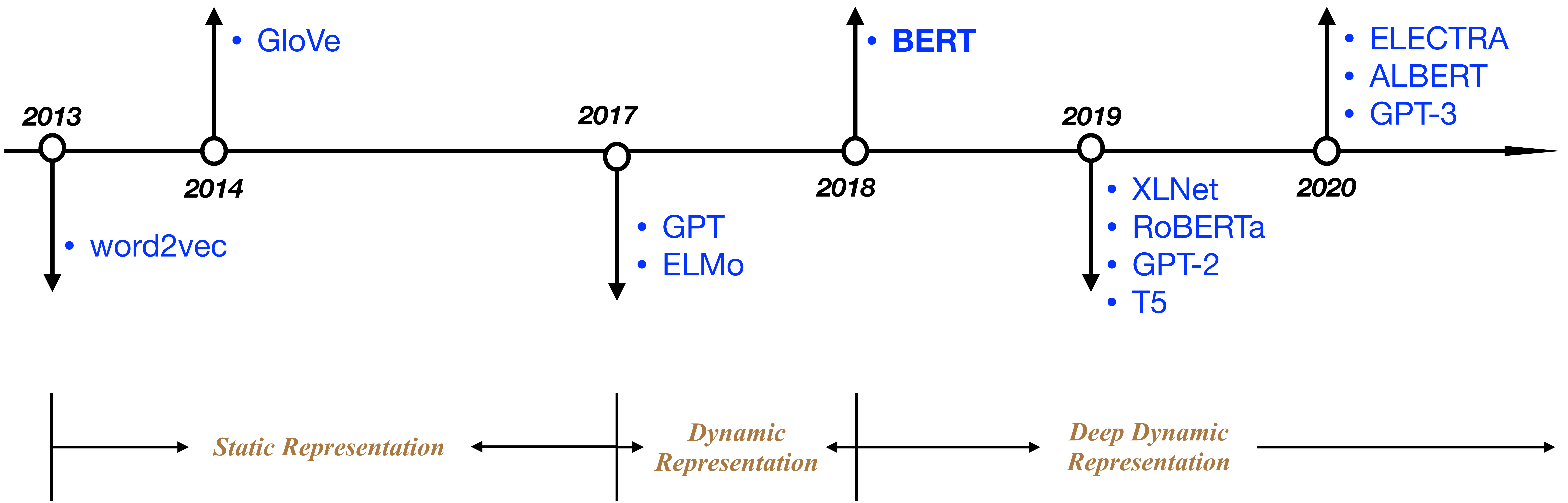
OpenAI

GPT

GPT-2

GPT-3

Introduction



传统文本表示方法

TRADITIONAL APPROACHES FOR TEXT REPRESENTATION

1. One-hot
2. word2vec
3. GloVe

One-Hot

- **Why should we use vector representations for text?**
  - Easy and eligible for calculation (similarity, distance, etc.)
  - Training neural models
- **Traditional Approaches for Text Representation**
  - One-hot Representation
  - word2vec
  - GloVe
  - NNLM, RNNLM, … (not covered in this talk)

One-Hot

- **One-hot Representations**
  - A binary vector with all zero values except for the index of the word is set to one

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

  - Drawbacks
    - failed to capture the similarity of the words
    - Can not express highly abstract meaning

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

word2vec

- **Distributed Representation of Words and Phrases and Their Compositionality**
- **Efficient Estimation of Word Representation in Vector Space**
  - Instead of capturing word co-occurrences, predict surrounding words of every word
  - Famous CBOW and Skip-gram model

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov Google Inc. Mountain View mikolov@google.com
Ilya Sutskever Google Inc. Mountain View ilyasu@google.com
Kai Chen Google Inc. Mountain View kai@google.com
Greg Corrado Google Inc. Mountain View gcorrado@google.com
Jeffrey Dean Google Inc. Mountain View jeff@google.com

Efficient Estimation of Word Representations in Vector Space

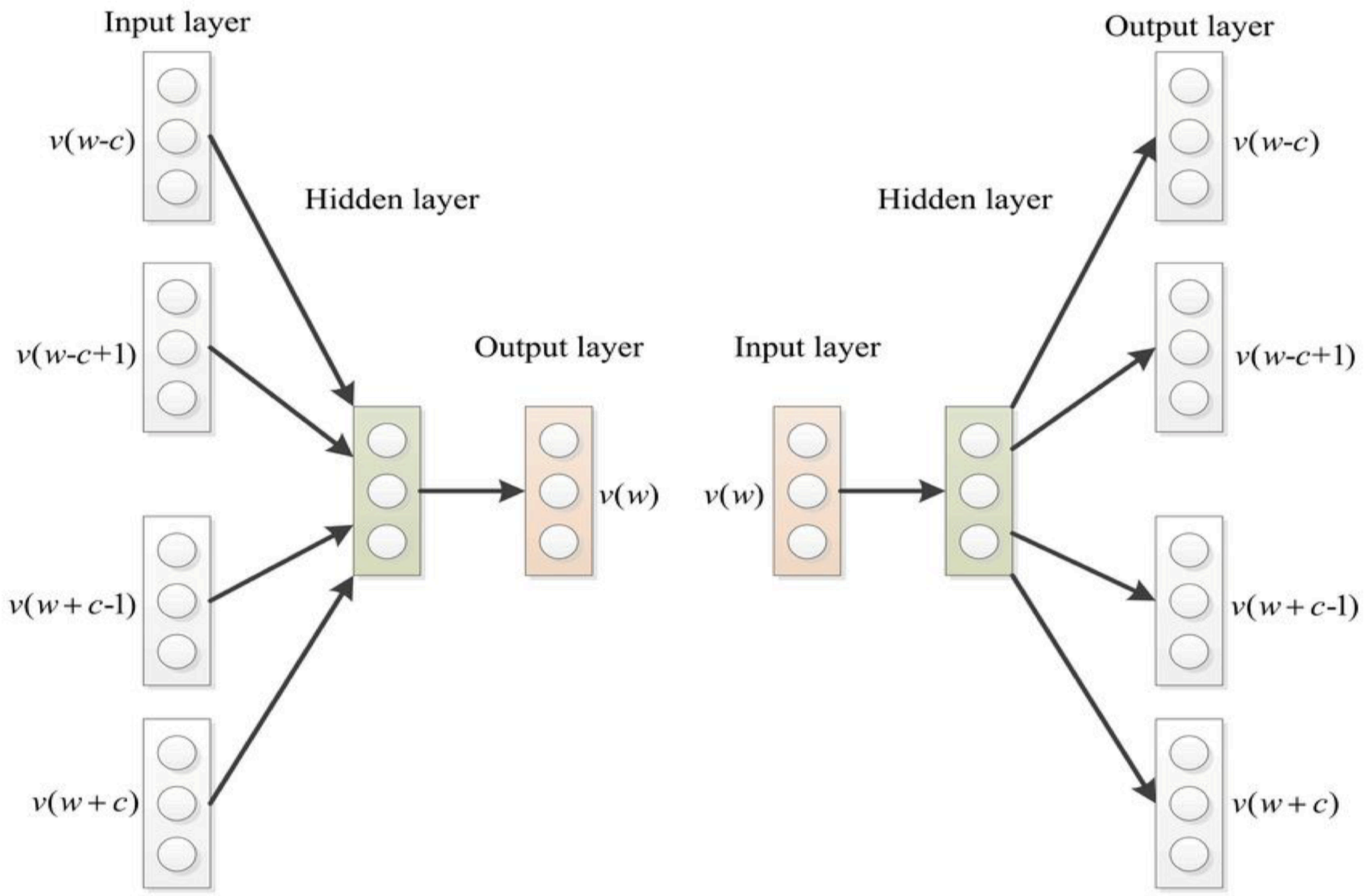
Tomas Mikolov Google Inc., Mountain View, CA tmikolov@google.com
Kai Chen Google Inc., Mountain View, CA kaichen@google.com
Greg Corrado Google Inc., Mountain View, CA gcorrado@google.com
Jeffrey Dean Google Inc., Mountain View, CA jeff@google.com

__Mikolov et al., NeurIPS 2013.__ Distributed Representations of Words and Phrases and Their Compositionality
__Mikolov et al., 2013.__ Efficient Estimation of Word Representations in Vector Space

word2vec

- **CBOW:** Using context to predict the central token
- **Skip-Gram:** Using central token to predict its context

CBOW



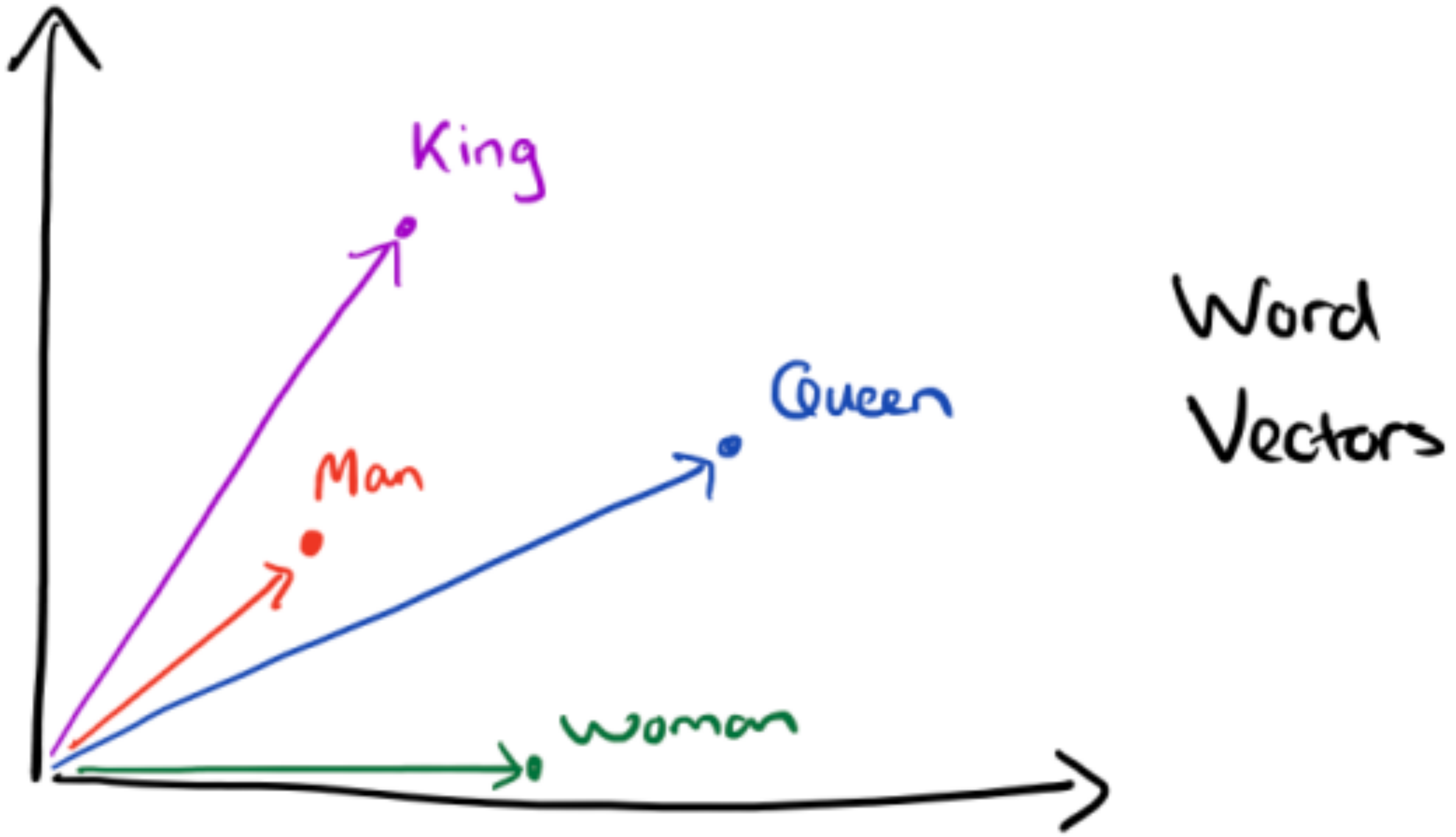
Skip-gram

Mikolov et al., NeurIPS 2013. *Distributed Representations of Words and Phrases and Their Compositionality*

Mikolov et al., 2013. *Efficient Estimation of Word Representations in Vector Space*

word2vec

King - Man + Woman = Queen



Mikolov et al., NeurIPS 2013. *Distributed Representations of Words and Phrases and Their Compositionality*
Mikolov et al., 2013. *Efficient Estimation of Word Representations in Vector Space*

GloVe

- **GloVe: Glo**bal **Ve**ctors for Word Representation
  - word2vec only considers LOCAL context
  - GloVe incorporates global information during word vector training
  - Other advantages: Fast training; Scalable to huge corpora; Good performance even with small corpus/vectors

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

Pennington et al., EMNLP 2014. *GloVe: Global Vectors for Word Representation*

GloVe

- **Main Advantage over word2vec: Introduce GLOBAL Information**
  - Observation: word-word co-occurrence probabilities have the potential for encoding some form of meaning
  - Training objective is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

$P(solid \mid ice) > P(gas \mid ice)$

$P(solid \mid steam) < P(gas \mid steam)$

Both *ice* and *steam* are less related to *fashion*

Ratio much greater/less than 1 matters

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

Pennington et al., EMNLP 2014. *GloVe: Global Vectors for Word Representation*

GloVe

- **Highlights of GloVe**

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

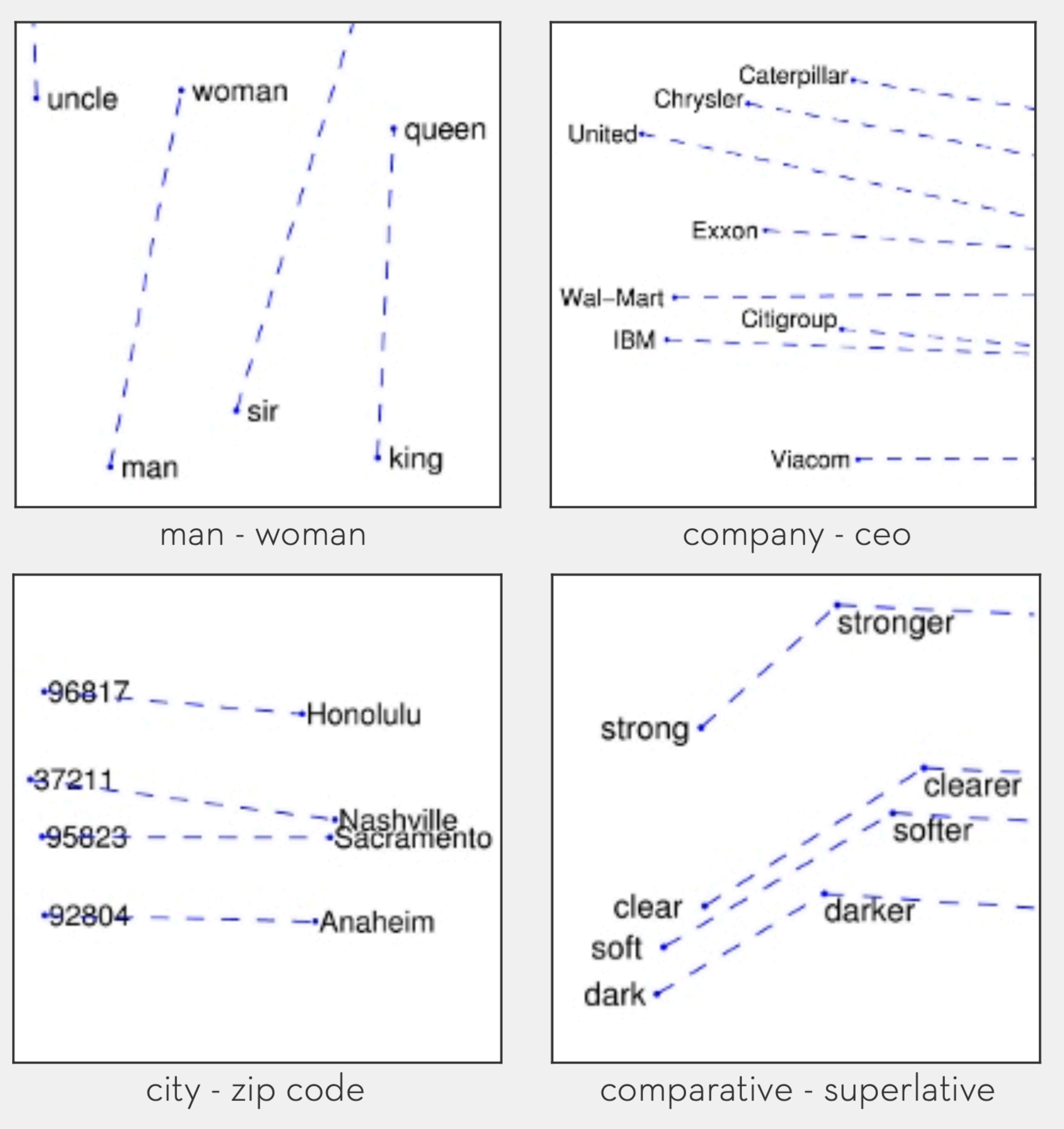
litoria

leptodactylidae

rana

eleutherodactylus

Nearest Neighbors



Linear Substructures

Pennington et al., EMNLP 2014. *GloVe: Global Vectors for Word Representation*

GloVe

- Pre-trained GloVe embeddings have been used in neural network models on a regular basis

> - Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
>   - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
>   - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
>   - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
>   - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip

- **Tips**
  - Choose pre-trained GloVe vectors based on your TOPIC
  - Performance: High Dimension > Low Dimension? Not always
  - If you are using a small dataset, it's better to freeze the embedding in case of overfitting

Pennington et al., EMNLP 2014. *GloVe: Global Vectors for Word Representation*

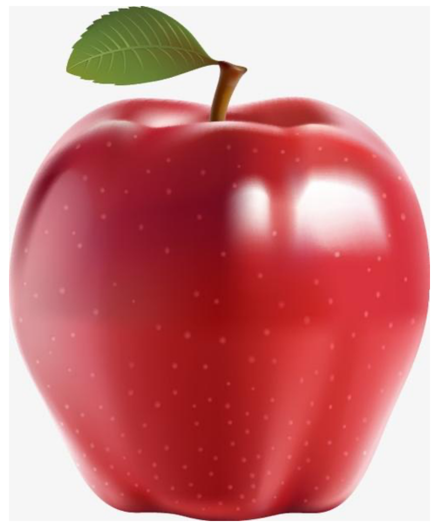
基于上下文的语言模型

CONTEXTUALIZED LANGUAGE MODELS

Pre-PLMs

- **Disadvantages of Static Embeddings**
  - Many words have different meanings

SAME!

Apple →  → | 0.1 | 0.2 | 0.8 | 0.3 |

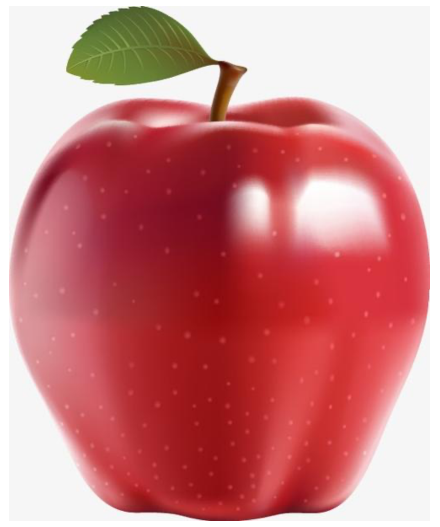
Apple → (Apple logo) → | 0.1 | 0.2 | 0.8 | 0.3 |

Apple → ... *the apple of my eye* → | 0.1 | 0.2 | 0.8 | 0.3 |

Pre-PLMs

- **Disadvantages of Static Embeddings**
  - The word vector should be adjusted according to its **context**

Apple



0.1	0.2	0.8	0.3

0.1	0.2	0.8	**0.5**

...

the apple of my eye

0.9	0.2	0.8	0.3

1 CoVe

2 ELMo

CoVe

- **CoVe: Co**ntextualized Word **Ve**ctors
  - The first paper that proposes a **contextualized** text representation approach
  - Transfer the knowledge in NMT to general NLP tasks

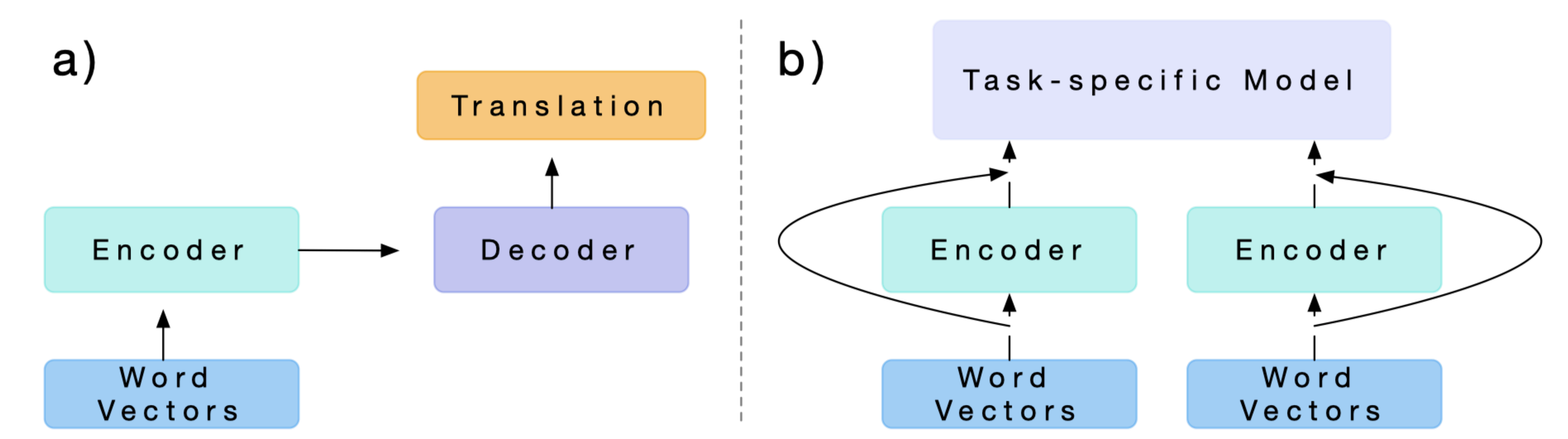
Learned in Translation: Contextualized Word Vectors

Bryan McCann
bmccann@salesforce.com

James Bradbury
james.bradbury@salesforce.com

Caiming Xiong
cxiong@salesforce.com

Richard Socher
rsocher@salesforce.com



McCann et al., NeurIPS 2017. *Learned in Translation: Contextualized Word Vectors*

CoVe

- **Training Phase: train a machine translation model**
  - Given a source sentence w^x, target sentence w^z

two-layer bidirectional-LSTM

$$h = \text{MT-LSTM}(\text{GloVe}(w^x))$$

  - Attentional Decoder

$$\alpha_t = \text{softmax}\left(H(W_1 h_t^{\text{dec}} + b_1)\right) \qquad \tilde{h}_t = \left[\tanh\left(W_2 H^\top \alpha_t + b_2; h_t^{\text{dec}}\right)\right]$$

$$h_t^{\text{dec}} = \text{LSTM}\left([z_{t-1}; \tilde{h}_{t-1}], h_{t-1}^{\text{dec}}\right)$$

  - Output

$$p(\hat{w}_t^z | X, w_1^z, \ldots, w_{t-1}^z) = \text{softmax}\left(W_{\text{out}} \tilde{h}_t + b_{\text{out}}\right)$$

McCann et al., NeurIPS 2017. *Learned in Translation: Contextualized Word Vectors*

CoVe

- **Inference Phrase**
  - Given a source sentence w

$$\text{CoVe}(w) = \text{MT-LSTM}(\text{GloVe}(w))$$

  - How to use CoVe in a downstream task?
    - Requires the SAME dimension of GloVe and CoVe

$$\tilde{w} = [\text{GloVe}(w); \text{CoVe}(w)]$$

Easy!

McCann et al., NeurIPS 2017. *Learned in Translation: Contextualized Word Vectors*

CoVe

- **Experimental Results**
  - Training: En-De 30K (small), 209K (medium), 7M (large)
  - Moderate improvements over traditional word representations, more seems like a remedy for word/char embedding

			GloVe+				
Dataset	Random	GloVe	Char	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	89.0	90.9	91.1	**91.2**
SST-5	48.6	53.5	52.2	54.0	54.7	54.5	**55.2**
IMDb	88.4	91.1	91.3	90.6	91.6	91.7	**92.1**
TREC-6	88.9	94.9	94.7	94.7	95.1	95.8	**95.8**
TREC-50	81.9	89.2	89.8	89.6	89.6	90.5	**91.2**
SNLI	82.3	87.7	87.7	87.3	87.5	87.9	**88.1**
SQuAD	65.4	76.0	78.1	76.5	77.1	79.5	**79.9**

*S=Small, M=Medium, L=Large

McCann et al., NeurIPS 2017. *Learned in Translation: Contextualized Word Vectors*

ELMo

- **ELMo: E**mbeddings from **L**anguage **Mo**dels (NAACL 2018 Best Paper)
  - Pre-training a deep bidirectional LM on a large corpus
  - ELMo can be easily added to the existing models

Deep contextualized word representations

Matthew E. Peters†, **Mark Neumann**†, **Mohit Iyyer**†, **Matt Gardner**†,
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark*, **Kenton Lee***, **Luke Zettlemoyer**†*
{csquared,kentonl,lsz}@cs.washington.edu

†Allen Institute for Artificial Intelligence
*Paul G. Allen School of Computer Science & Engineering, University of Washington

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

让世界聆听我们的声音

- **Training Phase: Bidirectional Language Model (BiLM)**
  - Given a sequence of N tokens $(t_1, t_2, \ldots, t_N)$
  - Forward LM $$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$
  - Backward LM $$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$
  - BiLM $$\sum_{k=1}^{N} \big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \big).$$

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **Training Phase: Bidirectional Language Model (BiLM)**
  - For each token t_k, a L-layer BiLM computes a set of $2L+1$ representations

Embedding *BiLM output*

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \boxed{\overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM}} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

  - Collapse all layers in R into a single vector

$$\mathbf{ELMo}_k = E(R_k; \mathbf{\Theta}_e)$$

  - Weighting of all BiLM layers

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **Inference Phase**
  - Just add one layer of ELMo at the same location as pre-trained word representations

$$X_{final} = concat[X_{char}, X_{word}, X_{ELMo}]$$

  - Useful Tips
    - For some tasks (such as SQuAD), adding another ELMo representation at RNN output could give slight improvements
    - Add some dropout (0.5 is a good default value) to the ELMo output
    - **Fine-tune the pre-trained ELMo if necessary**

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **Experimental Results**
  - Significant improvements over various NLP tasks
  - Moderate improvements when biLM is fine-tuned for the downstream task

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 7 lists the development set perplexities for the considered tasks. In every case except CoNLL 2012, fine tuning results in a large improvement in perplexity, e.g., from 72.1 to 16.8 for SNLI.

The impact of fine tuning on supervised performance is task dependent. In the case of SNLI, fine tuning the biLM increased development accuracy 0.6% from 88.9% to 89.5% for our single best model. However, for sentiment classification development set accuracy is approximately the same regardless whether a fine tuned biLM was used.

From Appendix A.1

Peters et al., NAACL 2018. *Deep contextualized word representations*

经典预训练语言模型

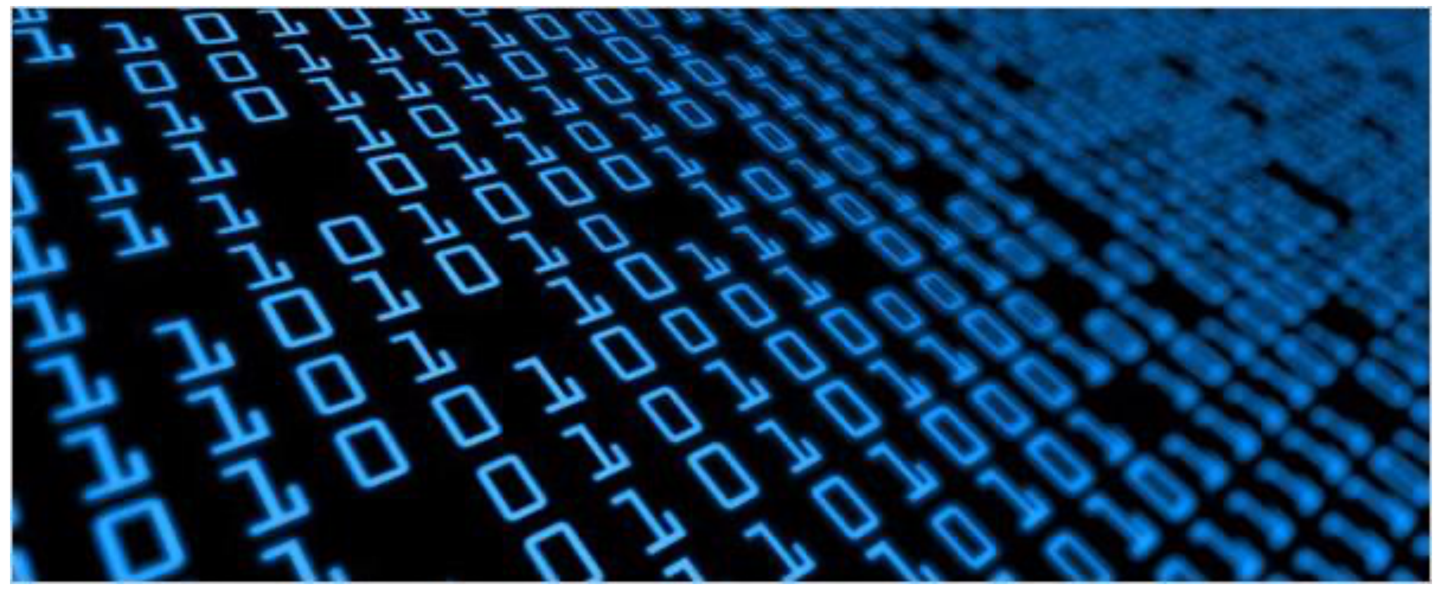
PRE-TRAINED LANGUAGE MODELS

PLMs

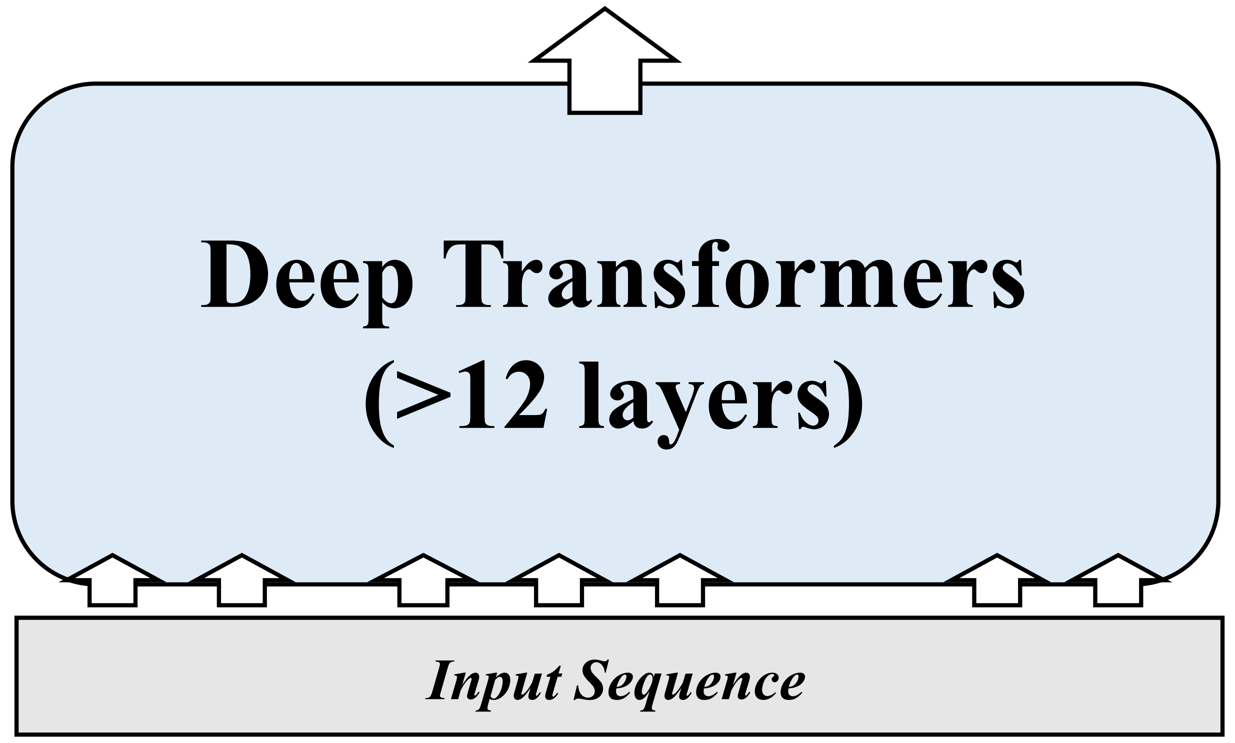
- **Main Disadvantages of CoVe/ELMo**
  - Data
    - Training data is either restricted to parallel corpus or relatively small
  - Model
    - Training parameters are relatively less (compared to PLMs)
  - Usage
    - Representations remains FIXED once the LMs are trained
    - Unable to unleash the power of LARGE PARAMETERS

PLMs

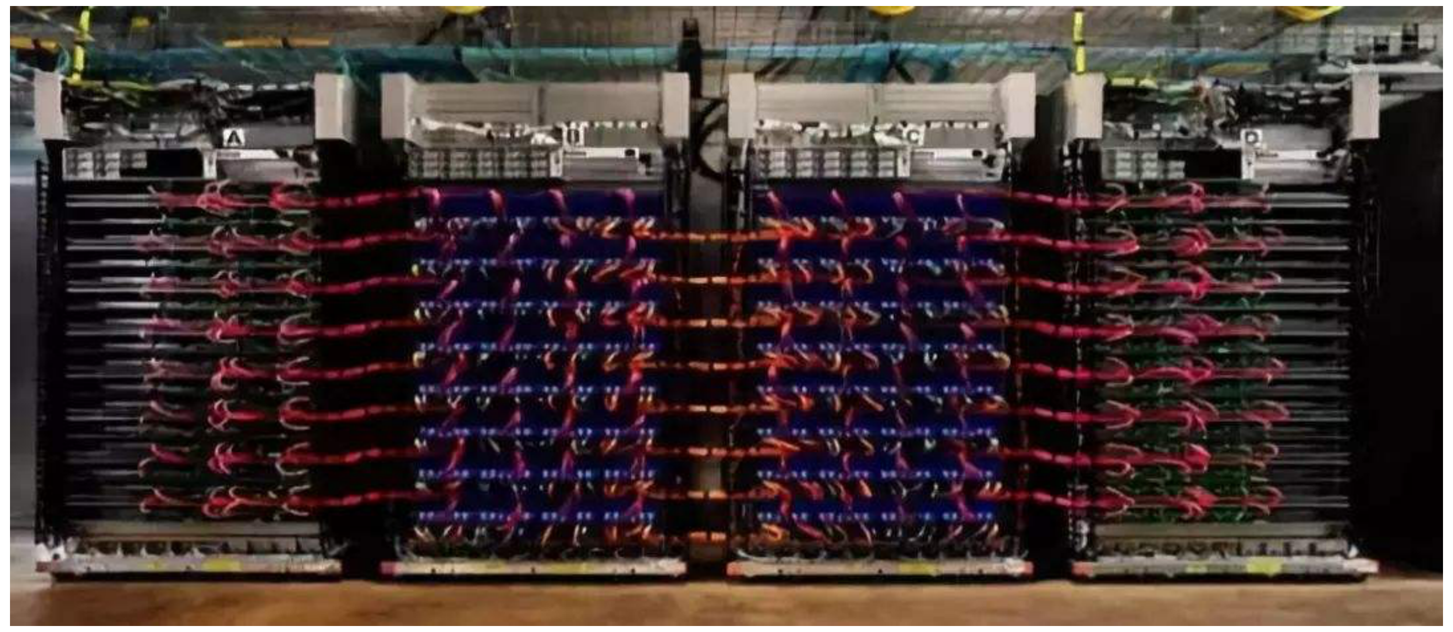
Big (Unsupervised) Data

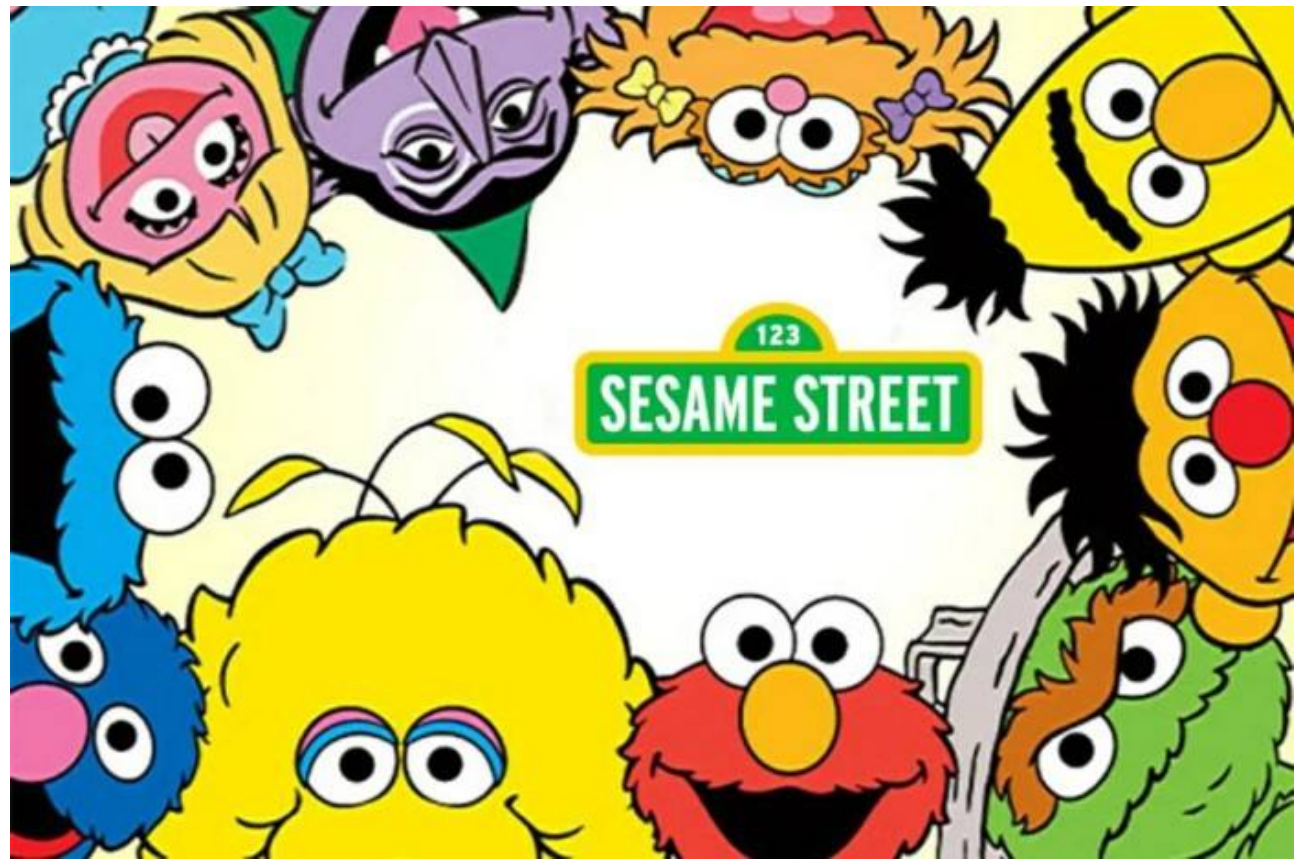


Deeper Neural Networks



Bigger and Faster Clusters





OpenAI

GPT

GPT-2

GPT-3

And more money

1 GPT

2 BERT

3 XLNet

4 RoBERTa

5 ALBERT

6 ELECTRA

GPT

- **GPT: G**enerative **P**re-**T**raining
  - Generative pre-training + discriminative fine-tuning scheme
  - Pre-training data size: 800M words (BooksCorpus)

Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI alec@openai.com
Karthik Narasimhan OpenAI karthikn@openai.com
Tim Salimans OpenAI tim@openai.com
Ilya Sutskever OpenAI ilyasu@openai.com

Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

- **Training Phase**
  - Learning a high-capacity LM on a large corpus
  - Training a standard left-to-right Transformer-based LM
  - Using a Transformer Decoder

Context Vector of Tokens ***Token Embedding Matrix***

$$h_0 = UW_e + W_p$$ ***Position Embedding Matrix***

$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$

$$P(u) = \texttt{softmax}(h_n W_e^T)$$

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

- **Inference Phase**
  - Fine-tune the model to a discriminative task with labeled data
  - Given a labeled dataset C, input tokens $x^1, \ldots, x^m$, label y

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

Transformer block's activation

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

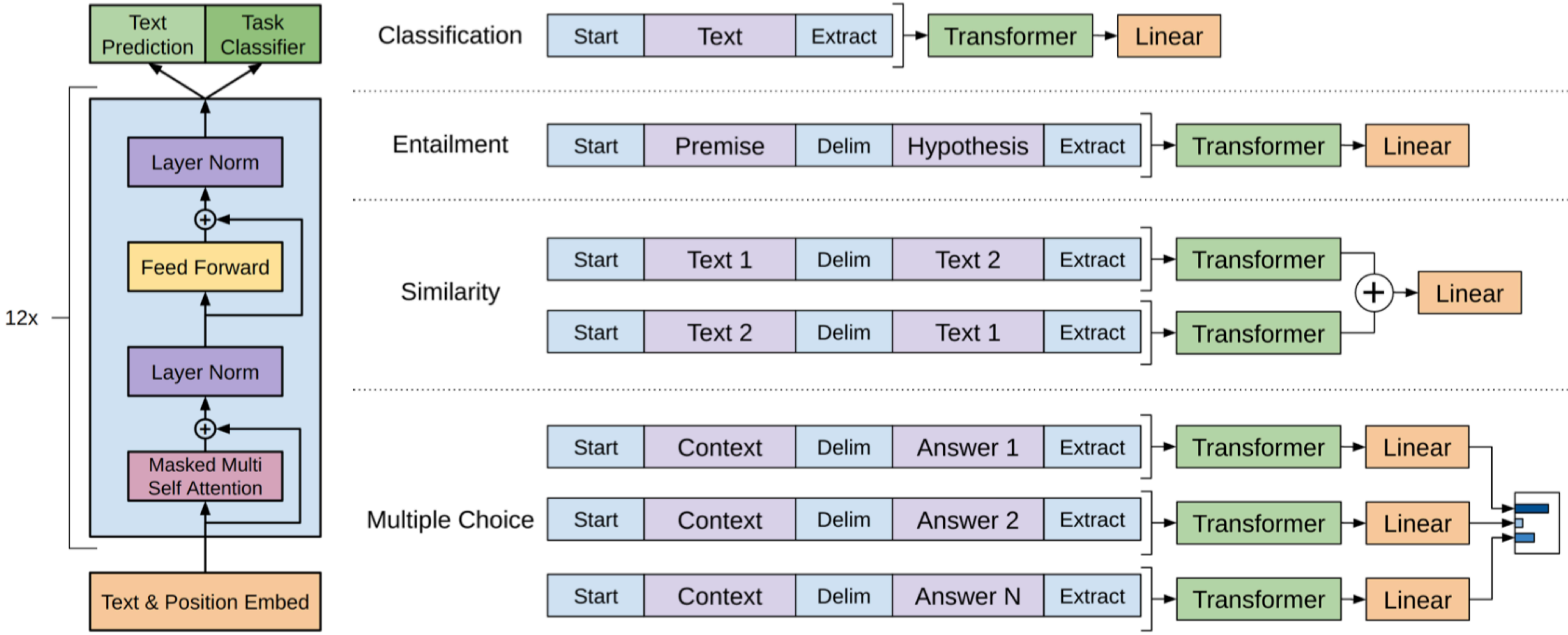
  - Using auxiliary loss could improve performance

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

- **Fine-tuning for Different Tasks**



Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

让世界聆听我们的声音

- **Experimental Results**

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

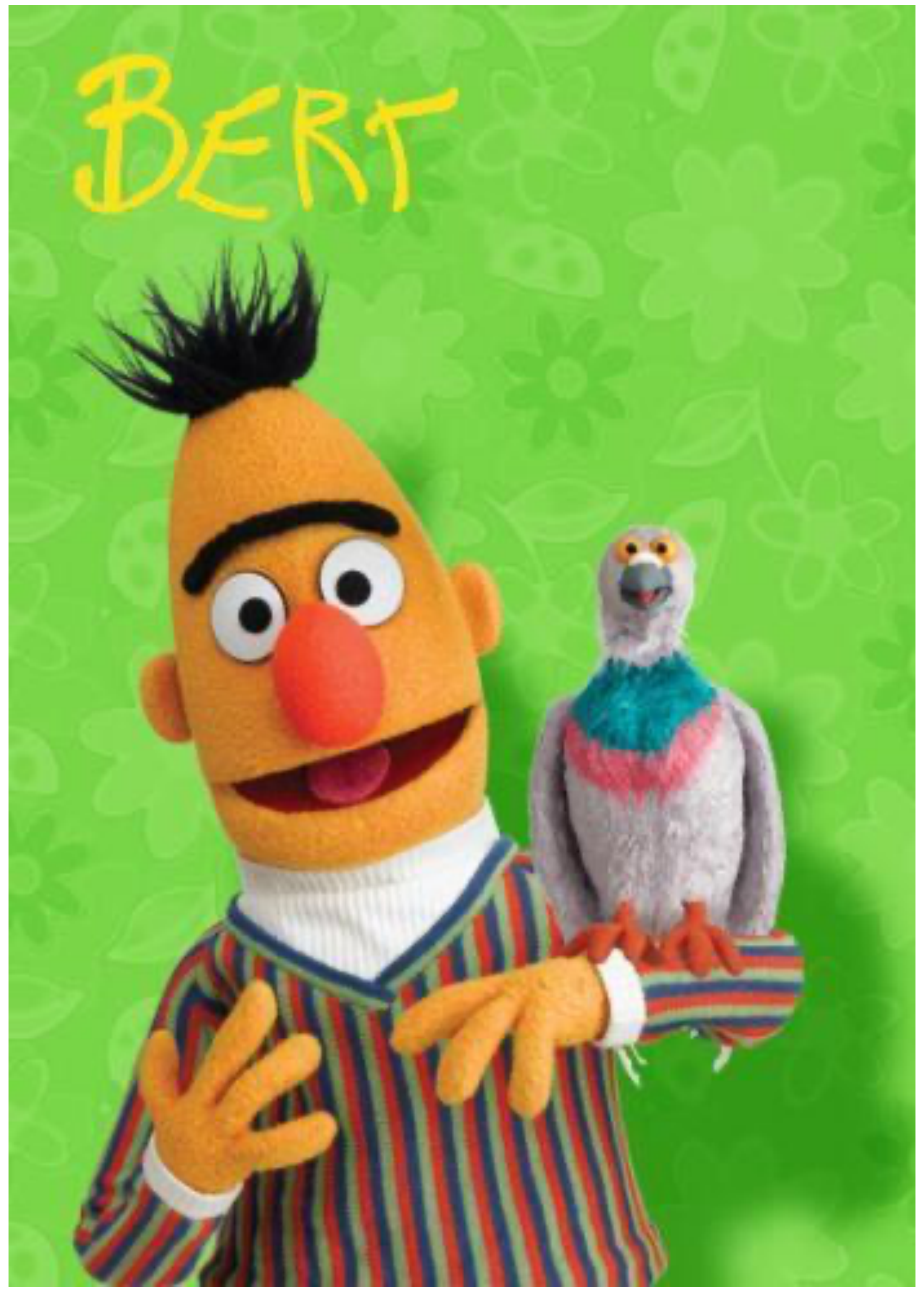
Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Method	**Classification**		**Semantic Similarity**			**GLUE**
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	68.9
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

***Radford et al., 2018.** Improving Language Understanding by Generative Pre-Training*

BERT

- **BERT**: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (NAACL 2019 Best Paper)
  - Demonstrate the importance of bidirectional pre-training for language representations
  - Pre-trained representations eliminate the needs of many heavily-engineered task-specific architectures
  - Pre-training data size: 800M (BooksCorpus) + 2500M (Wikipedia)



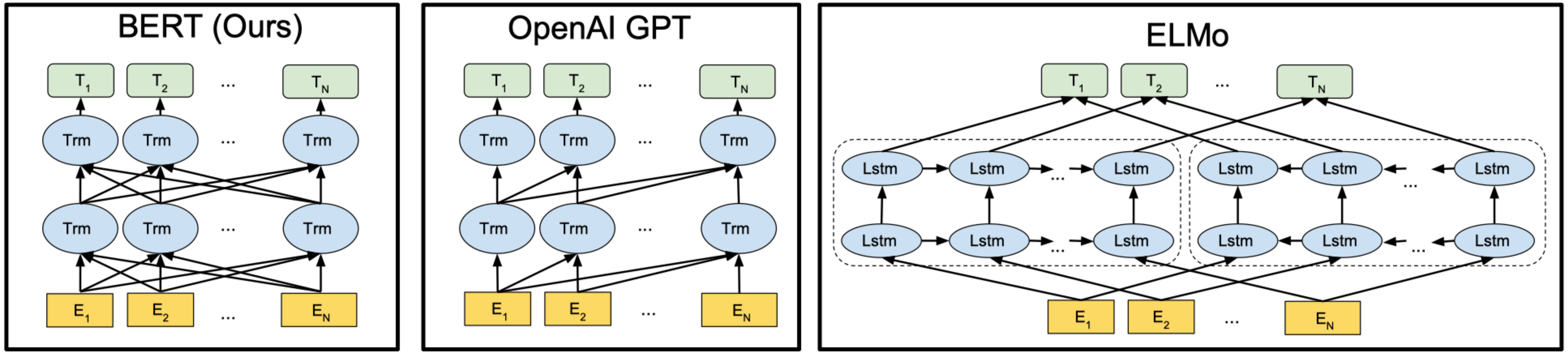
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

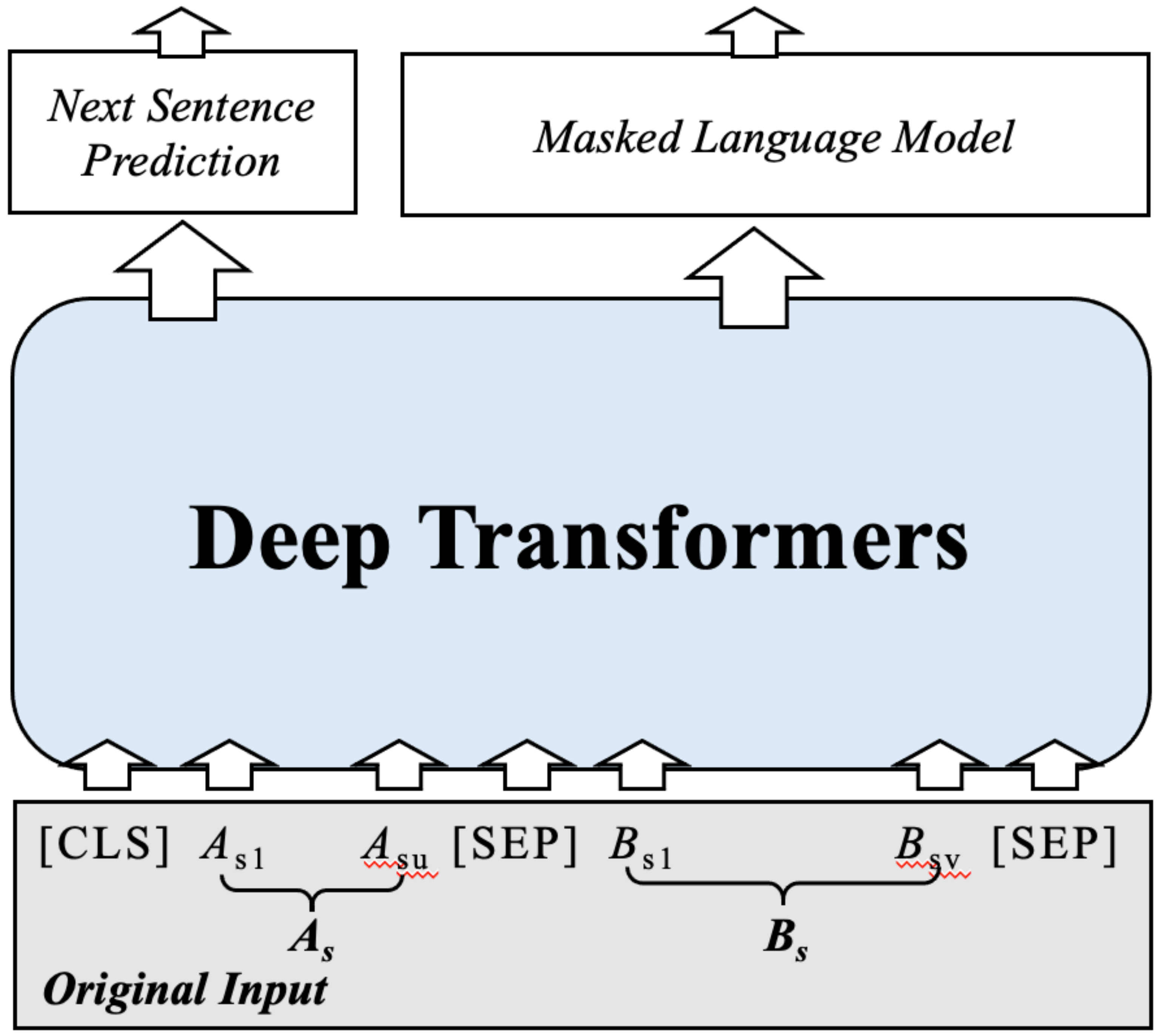
- **Comparisons of GPT/BERT/ELMo**
  - GPT: **unidirectional** left-to-right Transformer LM
  - ELMo: concatenation of **independent** left-to-right and right-to-left LSTM LM
  - BERT: **bi-directional** Transformer



Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Overview**
  - Neural architecture
    - Input representation
    - Deep Transformer-based encoder
  - Two pre-training tasks
    - MLM: Masked Language Model
    - NSP: Next Sentence Prediction



Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Pre-training Task I: Masked Language Model (MLM)**
  - Mask out several input words, and then predict the masked words

```
                     store                 gallon
                       ↑                     ↑
the man went to the [MASK] to buy a [MASK] of milk
```

  - Less masking: Easy to pick them out
  - More masking: Not enough context
  - Take a balance: use a percentage of 15%

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Pre-training Task I: Masked Language Model (MLM)**
  - Problem: Mask token never appear at fine-tuning stage (realistic data)
  - Solution: 15% of the words to predict, but don't replace with `[MASK]` 100% of the time
    - 80% of the time, replace with `[MASK]`
      - `went to the store → went to the [MASK]`
    - 10% of the time, replace random word
      - `went to the store → went to the apple`
    - 10% of the time, keep the same word
      - `went to the store → went to the store`

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Implementation for MLM**
  - File: `create_pretraining_data.py`
  - Function: `create_masked_lm_predictions()`
  - Arguments
    - Tokens (list): tokenized sequence tokens
    - masked_lm_prob (float): how many words (proportion) should be masked
    - max_predictions_per_seq (int): maximum predictions per sequence
    - vocab_words (list): vocabulary
    - rng: random.Random(seed)

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Step 1: Generate candidate indices**
  - Skip `[CLS]` and `[SEP]`
  - Shuffle candidate indexes
  - Determine the prediction number

```
cand_indexes = []
for (i, token) in enumerate(tokens):
  if token == "[CLS]" or token == "[SEP]":
    continue
  cand_indexes.append(i)

rng.shuffle(cand_indexes)

output_tokens = list(tokens)

num_to_predict = min(max_predictions_per_seq,
                     max(1, int(round(len(tokens) * masked_lm_prob))))
```

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Step 2: Mask out proper tokens**
  - Regular checks to avoid overflow
  - Generate random number to determine the masking action

```python
masked_lms = []
covered_indexes = set()
for index in cand_indexes:
  if len(masked_lms) >= num_to_predict:
    break
  if index in covered_indexes:
    continue
  covered_indexes.add(index)
  masked_token = None
  if rng.random() < 0.8:
    masked_token = "[MASK]"          # 80% of the time, replace with [MASK]
  else:
    if rng.random() < 0.5:
      masked_token = tokens[index]  # 10% of the time, keep original
    else:
      masked_token = vocab_words[rng.randint(0, len(vocab_words) - 1)]  # 10% of the time, replace with random word

  output_tokens[index] = masked_token
  masked_lms.append(MaskedLmInstance(index=index, label=tokens[index]))
```

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

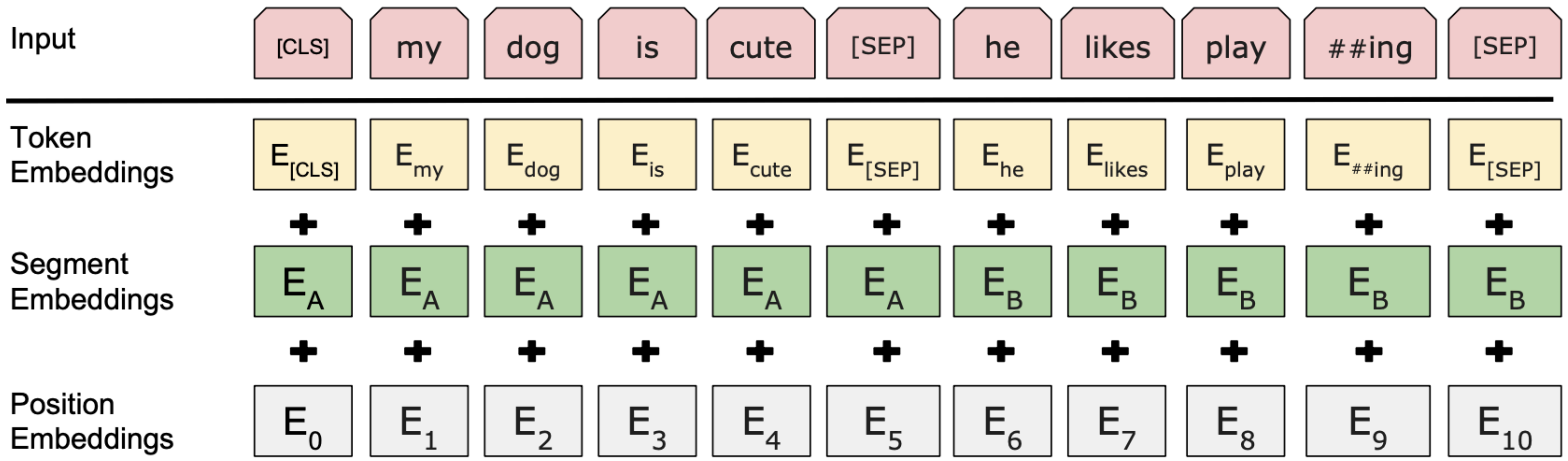
- **Pre-training Task II: Next Sentence Prediction (NSP)**
  - Learn the relationships between sentences (contextual information)
  - Predict whether *Sentence B* is the actual sentence that comes after *Sentence A*, or a random sentence

Sentence A = `The man went to the store.` **Sentence B =** `He bough a gallon of milk.` **Label =** `IsNextSentence`	**Sentence A =** `The man went to the store.` **Sentence B =** `Penguins are flightless.` **Label =** `NotNextSentence`

***Devlin et al., NAACL 2019.** BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

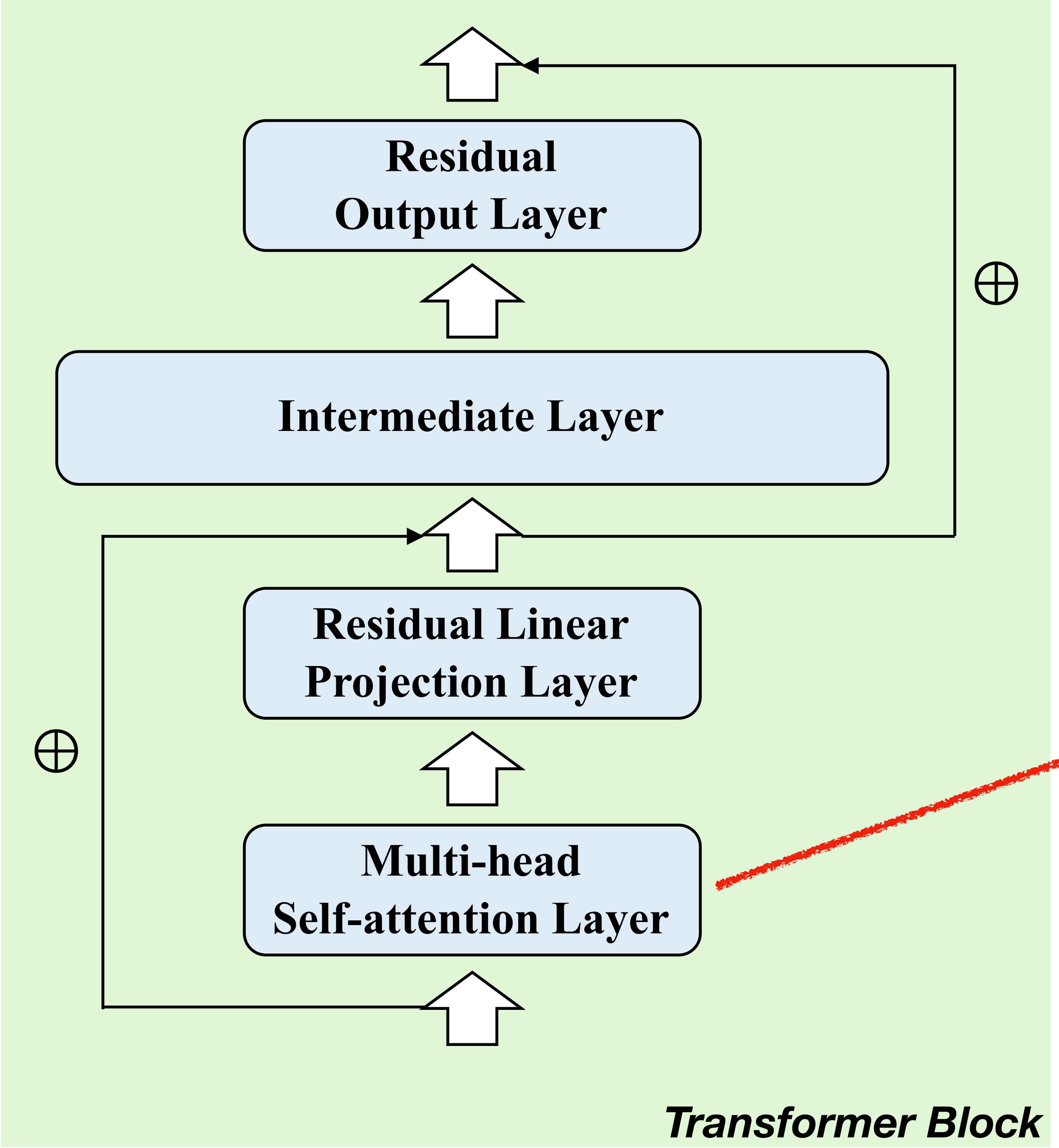
- **Input Representation**
  - Use a 30,000 WordPiece vocabulary
  - The final input is the sum of three embeddings
    - Token Embeddings, Segment Embeddings, Position Embeddings

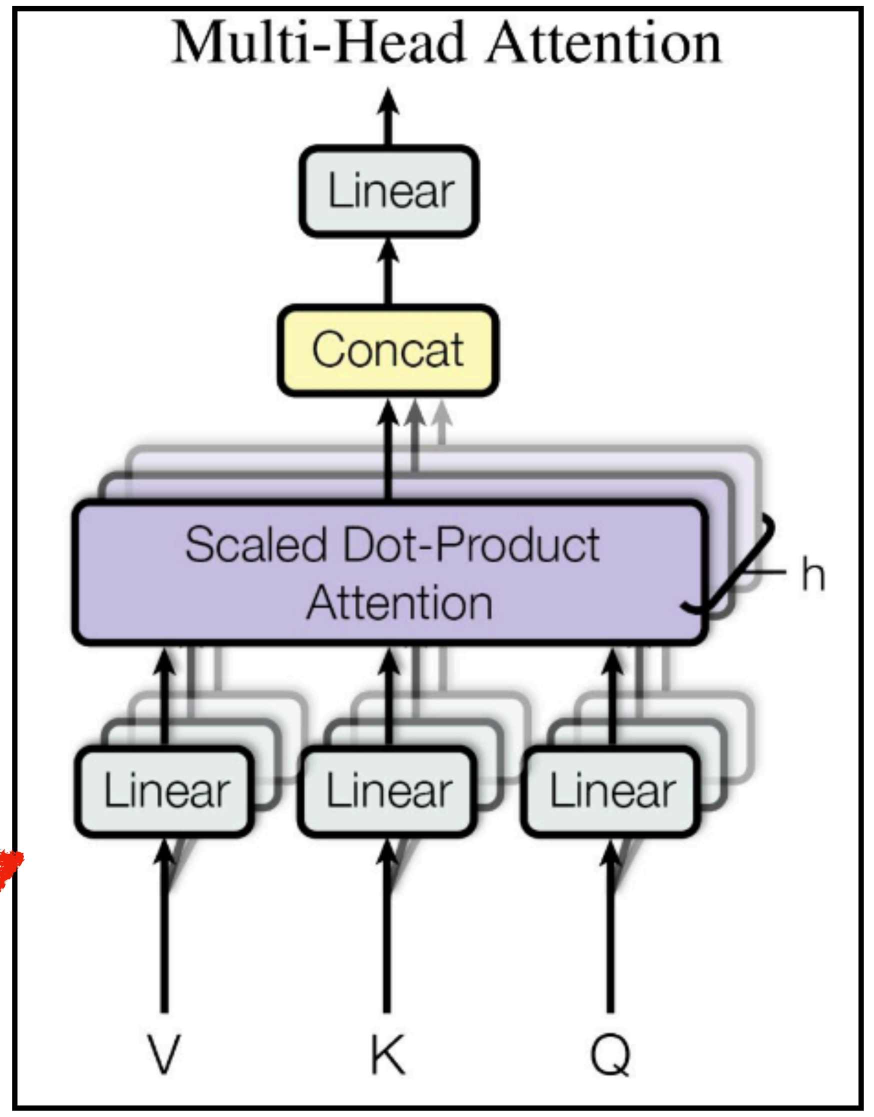


Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Deep Transformer Encoder**
  - Typically a 12 or 24-layer Transformers
  - Main loop
    - Multi-head Self-attention Layer
    - Residual Linear Projection Layer (+LayerNorm)
    - Intermediate Layer
    - Residual Output Layer (+LayerNorm)



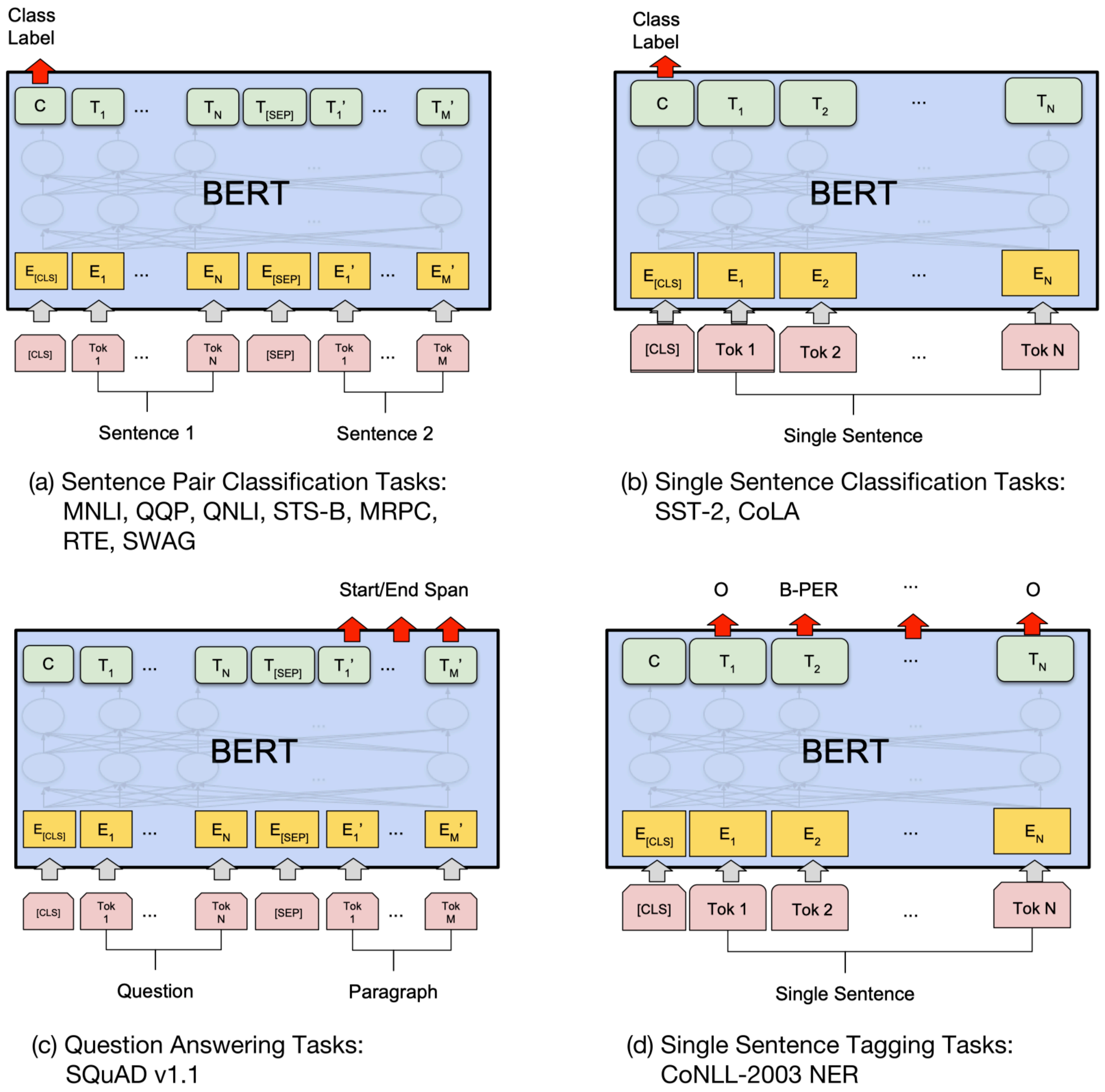


Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

让世界聆听我们的声音

- **Fine-tuning BERT**
  - Input
    - SPC: `[CLS] sent1`
    - SSC: `[CLS] sent1 [SEP] sent2`
    - QA: `[CLS] Q [SEP] P`
  - Leave everything to BERT 😄
  - Output
    - SPC/SSC: `[CLS]` → label
    - QA: start/end pointer network



***Devlin et al., NAACL 2019.** BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Official implementation

Let's read the code!

BERT

- **SQuAD (Stanford Question Answering Dataset)**
  - A span-extraction reading comprehension dataset that contains over 100,000 human-annotated questions
  - **Passage**: Passages from Wikipedia pages, segment into several small paragraphs
  - **Question**: Human-annotated, including various question types (what/when/where/who/how/why, etc.)
  - **Answer**: Continuous segments (text spans) in the passage, which has a larger search space, and much harder to answer than cloze-style RC

SQuAD
The Stanford Question Answering Dataset

Oxygen
The Stanford Question Answering Dataset

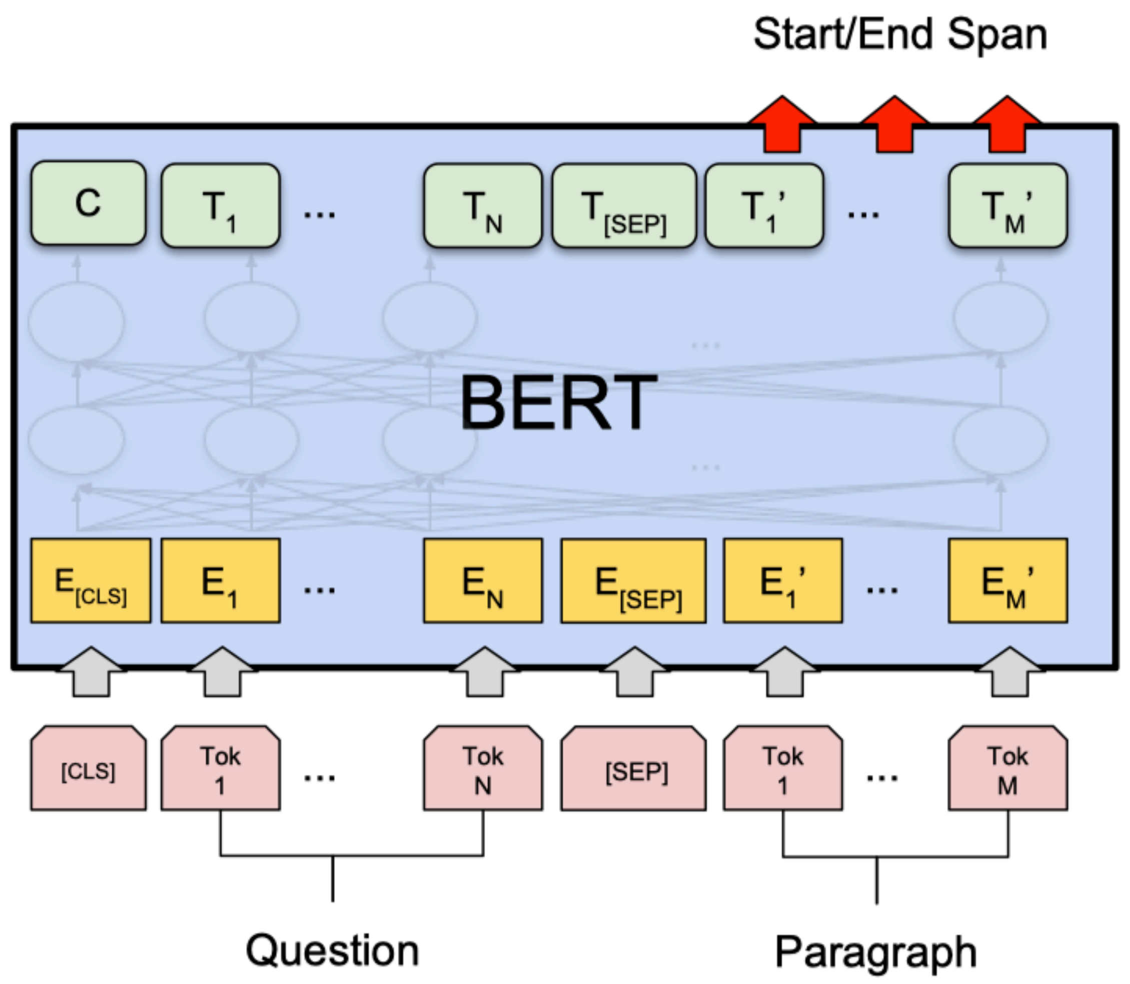
In the meantime, on August 1, 1774, an experiment conducted by the British clergyman Joseph Priestley focused sunlight on mercuric oxide (HgO) inside a glass tube, which liberated a gas he named "dephlogisticated air". He noted that candles burned brighter in the gas and that a mouse was more active and lived longer while breathing it. After breathing the gas himself, he wrote: "The feeling of it to my lungs was not sensibly different from that of common air, but I fancied that my breast felt peculiarly light and easy for some time afterwards." Priestley published his findings in 1775 in a paper titled "An Account of Further Discoveries in Air" which was included in the second volume of his book titled Experiments and Observations on Different Kinds of Air. Because he published his findings first, Priestley is usually given priority in the discovery.

Why is Priestley usually given credit for being first to discover oxygen?
Ground Truth Answers: published his findings first | he published his findings first | he published his findings first | he published his findings first | Because he published his findings first

Rajpurkar et al., EMNLP 2016. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*

BERT

- **Implementation for fine-tuning SQuAD (reading comprehension)**
  - File: `run_squad.py`
  - Function: `create_model()`
  - Arguments
    - bert_config (json): BERT config file
    - is_training (bool): training mode option
    - input_ids (tensor): input ids for token embeddings
    - input_mask (tensor): input mask for indicating non-padding positions
    - segment_ids (tensor): segment_id tensor
    - use_one_hot_embeddings (bool)



Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

让世界聆听我们的声音

- **Step 1: Generate BERT representation**
  - Define a BERT model
  - Generate sequential output (3D-tensor: <batch, seq_len, hid_size>)

```
model = modeling.BertModel(
        config=bert_config,
        is_training=is_training,
        input_ids=input_ids,
        input_mask=input_mask,
        token_type_ids=segment_ids,
        use_one_hot_embeddings=use_one_hot_embeddings)

final_hidden = model.get_sequence_output()

final_hidden_shape = modeling.get_shape_list(final_hidden, expected_rank=3)
batch_size = final_hidden_shape[0]
seq_length = final_hidden_shape[1]
hidden_size = final_hidden_shape[2]
```

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Step 2: Simple output layer for span prediction**
  - Define a fully-connected (dense) layer
  - Squeeze the vector to a scalar to get raw span output

```
output_weights = tf.get_variable(
    "cls/squad/output_weights", [2, hidden_size],
    initializer=tf.truncated_normal_initializer(stddev=0.02))
output_bias = tf.get_variable("cls/squad/output_bias", [2], initializer=tf.zeros_initializer())

final_hidden_matrix = tf.reshape(final_hidden, [batch_size * seq_length, hidden_size])
logits = tf.matmul(final_hidden_matrix, output_weights, transpose_b=True)
logits = tf.nn.bias_add(logits, output_bias)

logits = tf.reshape(logits, [batch_size, seq_length, 2])
logits = tf.transpose(logits, [2, 0, 1])

unstacked_logits = tf.unstack(logits, axis=0)
(start_logits, end_logits) = (unstacked_logits[0], unstacked_logits[1])

return (start_logits, end_logits)
```

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Step 3: Create loss for answer span**
  - Function: `model_fn_builder()` → `compute_loss()`
  - Compute regular cross-entropy loss for the start and end positions

```
def compute_loss(logits, positions):
  one_hot_positions = tf.one_hot(
      positions, depth=seq_length, dtype=tf.float32)
  log_probs = tf.nn.log_softmax(logits, axis=-1)
  loss = -tf.reduce_mean(
      tf.reduce_sum(one_hot_positions * log_probs, axis=-1))
  return loss
```

BERT

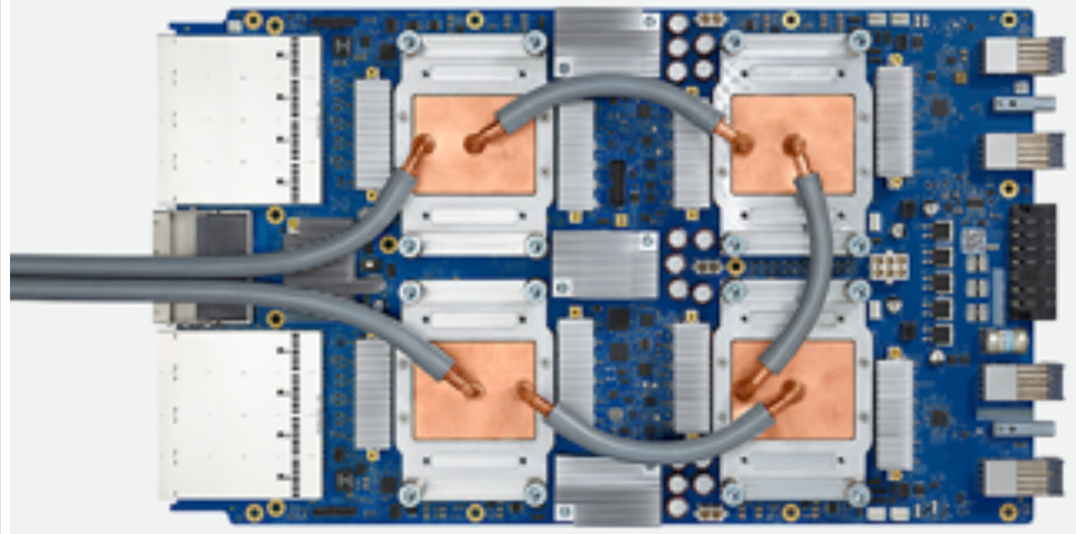
- **Experiments: setups**
  - Data: Wikipedia + BookCorpus (33B words in total)
  - Training: 256 batch * 512 max_token_length, 1M steps
  - Warmup: 10K steps (1% of total training steps)
  - Time: 4 days
  - Computing Device
    - BERT-base: 4 Cloud TPUs in Pod config (16 chips)
    - BERT-large: 16 Cloud TPUs (64 chips)

1 TPU has 2 cores, and 4 chips each

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **TPU (Tensor Processing Units)**

	NVIDIA V100	TPU v2	TPU v3
Hardware			
Architecture	NVIDIA Volta GPU	Google Cloud TPU	Google Cloud TPU
Memory	16GB / 32GB	64GB	128GB
FLOPS	Double: 7 TFLOPS Single: 14 TFLOPS DL: 112 TFLOPS	180 TFLOPS	420 TFLOPS

Google Cloud TPU. *https://cloud.google.com/tpu*

BERT

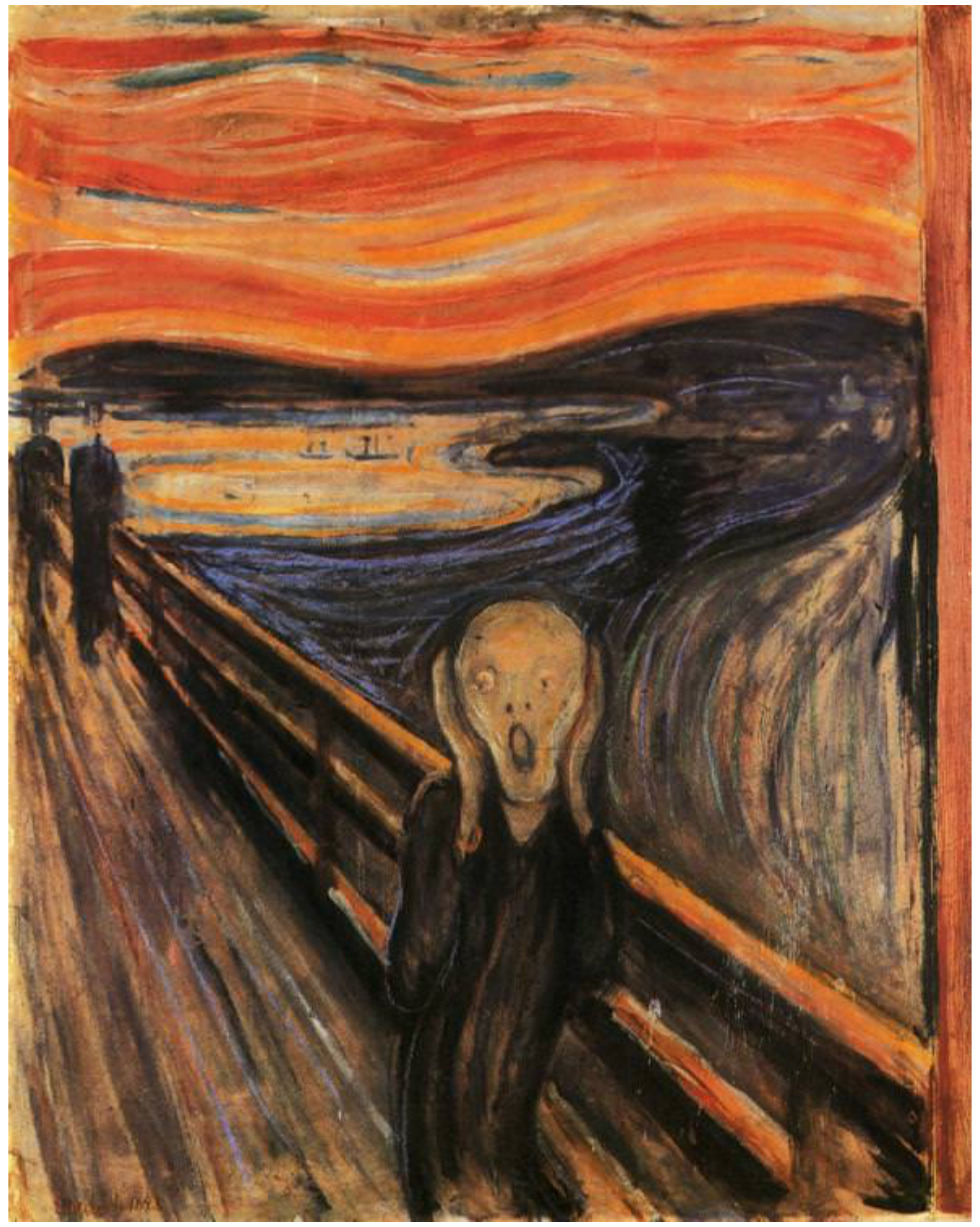
- **Question: How much does it cost to train such a model?**
  - Take BERT-large as an example,

16 Cloud TPUs = 16 * 4.5 = 72 USD / hour

One-day cost = 72 * 24 = 1,728 USD

Four-day cost = 1,728 USD * 4 = 6,912 USD

6,912 USD ≈ 47,715 CNY



Actually, it costs way more, as you won't be able to successfully train a model only once!

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **Experimental Results**
  - significant improvements over GPT/ELMo on GLUE and SQuAD

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **BERT + DAE + AoA** (by HFL)
  - Outperformed human performance (EM/F1) on SQuAD 2.0

Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Nov 16, 2018	AoA + DA + BERT (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	**82.374**	**85.310**
2 Nov 16, 2018	AoA + DA + BERT (single model) *Joint Laboratory of HIT and iFLYTEK Research*	81.178	84.251
3 Nov 16, 2018	Candi-Net+BERT (single model) *42Maru NLP Team*	80.106	82.862
3 Nov 08, 2018	BERT (single model) *Google AI Language*	80.005	83.061
4 Nov 09, 2018	L6Net + BERT (single model) *Layer 6 AI*	79.181	82.259
5 Nov 06, 2018	SLQA+BERT (single model) *Alibaba DAMO NLP* http://www.aclweb.org/anthology/P18-1158	77.003	80.209

Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Mar 20, 2019	BERT + DAE + AoA (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	**87.147**	**89.474**
2 Mar 15, 2019	BERT + ConvLSTM + MTL + Verifier (ensemble) *Layer 6 AI*	86.730	89.286
3 Mar 05, 2019	BERT + N-Gram Masking + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	86.673	89.147
4 Mar 16, 2019	BERT + DAE + AoA (single model) *Joint Laboratory of HIT and iFLYTEK Research*	85.884	88.621
5 Jan 15, 2019	BERT + MMFT + ADA (ensemble) *Microsoft Research Asia*	85.082	87.615

SQuAD2.0
The Stanford Question Answering Dataset

Pranav Rajpurkar @pranavrajpur... · 5天
Human Performance on SQuAD2.0 has now been surpassed! Been in the making for the last year, but finally achieved! Congratulations Joint Laboratory of HIT and iFLYTEK Research on this milestone! @KCrosner @stanfordnlp

BERT-wwm

- **Original Extension to BERT I: Whole Word Masking (wwm)**
    - Original masking: randomly select WordPiece tokens to mask.
    - WWM: always mask all of the tokens corresponding to a word at once.
    - The overall masking rate remains the same.

Original Sentence	the man jumped up , put his basket on phil ##am ##mon ' s head
Original Masked Input	[M] man [M] up , put his [M] on phil [M] ##mon ' s head
BERT-wwm Input	the man [MASK] up , put his basket on [M] [M] [M] ' s head

BERT-wwm

- **Important Note on Whole Word Masking**
  - ‘Masking’ does **NOT** only represent replacing a word into `[MASK]` token.
  - Masking = ‘replace into `[MASK]`’, ’keep original word’ or ‘replaced by random words’.

Original Sentence: there is an apple tree nearby.	
Tokenized Sentence: ["there", "is", "an", "ap", "##p", "##le", "tr", "##ee", "nearby", "."]	
w/o wwm	there [MASK] an ap [MASK] ##le tr [RANDOM] nearby . [MASK] [MASK] an ap ##p [MASK] tr ##ee nearby . there is [MASK] ap ##p ##le [MASK] ##ee [MASK] . there is [MASK] ap [MASK] ##le tr ##ee nearby [MASK] . there is an! ap ##p ##le tr [MASK] nearby [MASK] . there is an [MASK] ##p [MASK] tr ##ee nearby [MASK] .
w/ wwm	there is an [MASK] [MASK] [RANDOM] tr ##ee nearby . there is! [MASK] ap ##p ##le tr ##ee nearby [MASK] . there is [MASK] ap ##p ##le [MASK] [MASK] nearby . there [MASK] [MASK] ap ##p ##le tr ##ee [RANDOM] . there is an ap ##p ##le [MASK] [MASK] nearby [MASK] .

BERT-wwm

- **Experimental Results on BERT-wwm**
    - Significant improvements over vanilla MLM

Model	**SQuAD 1.1 F1/EM**	**Multi NLI Accuracy**
BERT-Large, Uncased (Original)	91.0/84.3	86.05
BERT-Large, Uncased (WWM)	92.8/86.7	87.07
BERT-Large, Cased (Original)	91.5/84.8	86.09
BERT-Large, Cased (WWM)	92.9/86.7	86.46

N-gram Masking

- **Original Extension to BERT II: N-gram Masking**
  - Masking a consecutive N-gram, increasing the difficulty in MLM
  - Yields another gain over MLM/WWM

```
We went to the store to buy some fruits.

→ We went to [M] store to [M] some [M].

→ We went to [M] [M] [M] buy some fruits.
```

Original MLM

N-gram Masking

- **Important Note**: WWM/NM **ONLY** affects the pre-training stage

XLNet

- **XLNet:** Transformer-**XL Net**
  - An autoregressive language modeling that could capture bidirectional contexts
  - Resolve the **pretraining-finetuning discrepancy** in denoising auto-encoder (BERT)

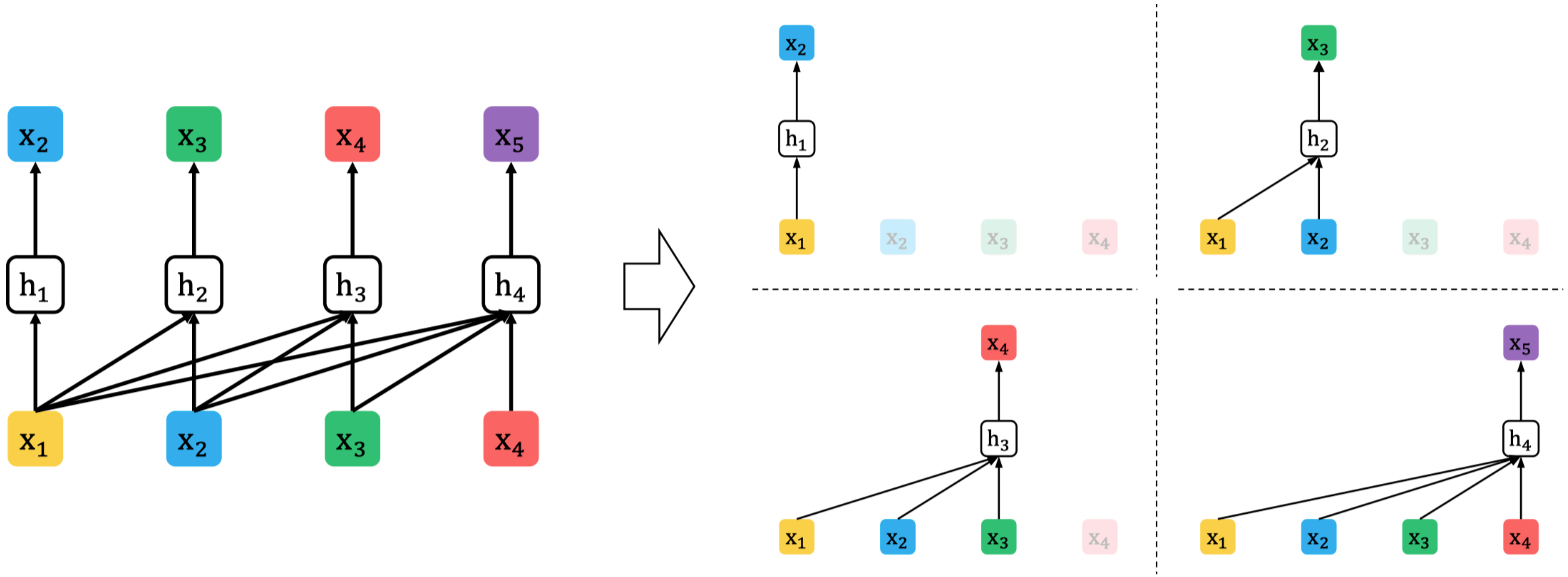
XLNet: Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang[*1], **Zihang Dai**[*12], **Yiming Yang**[1], **Jaime Carbonell**[1], **Ruslan Salakhutdinov**[1], **Quoc V. Le**[2]
[1]Carnegie Mellon University, [2]Google AI Brain Team
{zhiliny,dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

Yang et al., NeurIPS 2020. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

XLNet

- **Standard Language Model**
  - Left-to-right factorization: 1→2→3→4
  - $P(\mathbf{x}) = P(x_1)P(x_2 \mid \mathbf{x_1})P(x_3 \mid \mathbf{x_{1,2}})P(x_4 \mid \mathbf{x_{1,2,3}})\ldots$



Yang et al., NeurIPS 2020. XLNet: Generalized Autoregressive Pretraining for Language Understanding

XLNet

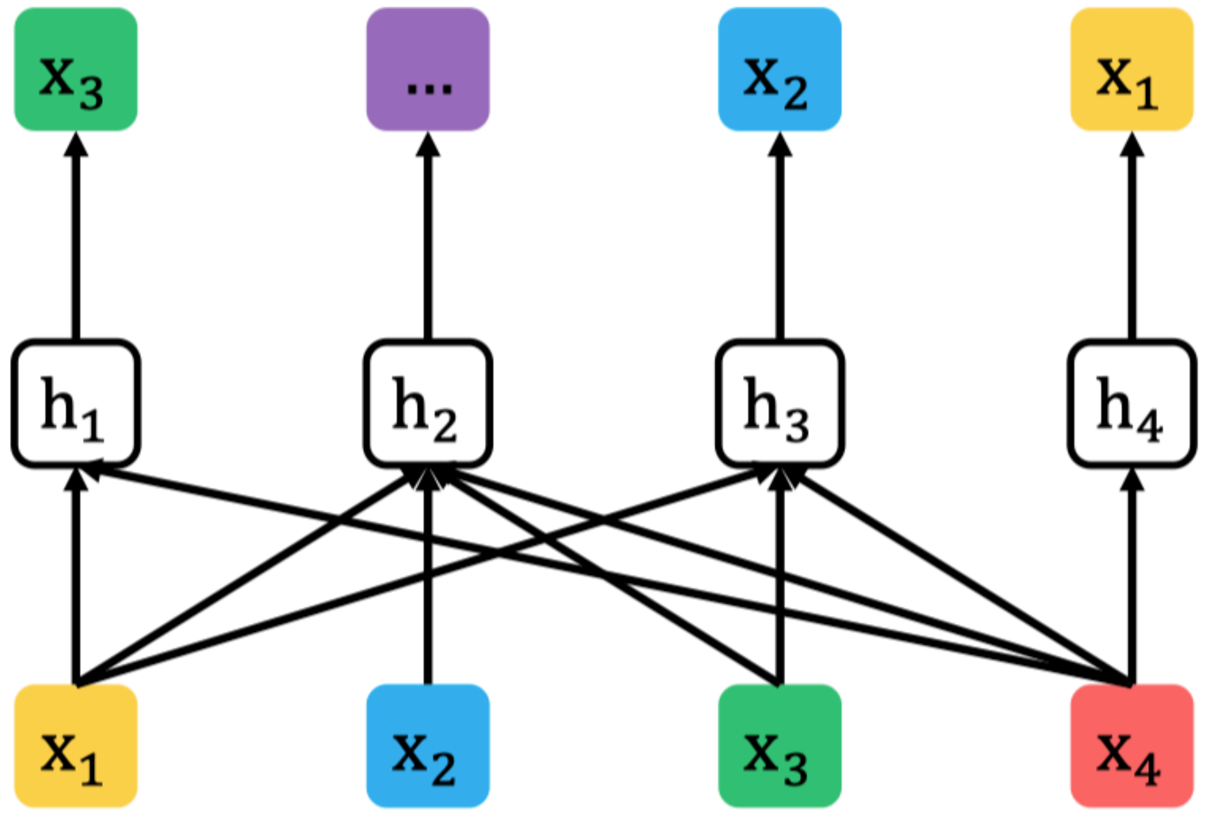
- **Permutation Language Model**
  - Given a sequence $\mathbf{x}$ of length T
  - Uniformly sample a factorization order $\mathbf{z}$ from all possible permutations
  - Maximize the permuted log-likelihood

$$\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\log P(\mathbf{x} \mid \mathbf{z})\right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T} P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right]$$

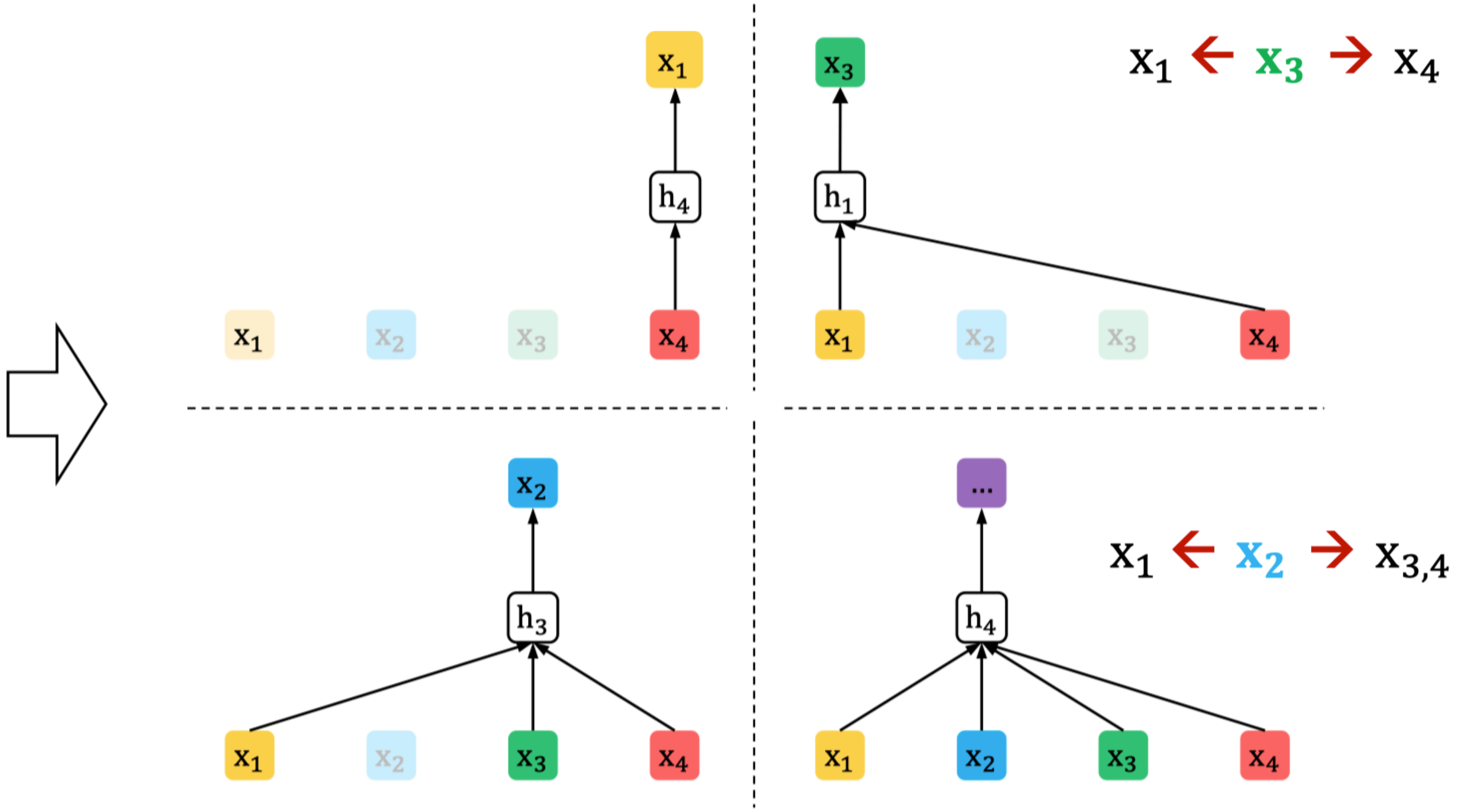
Yang et al., NeurIPS 2020. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

XLNet

- **Permutation Language Model**
  - Change the factorization order to: 4→1→3→2
  - $P(\mathbf{x}) = P(x_4)P(x_1 \mid \mathbf{x}_4)P(x_3 \mid \mathbf{x}_{1,4})P(x_2 \mid \mathbf{x}_{1,3,4})\ldots$



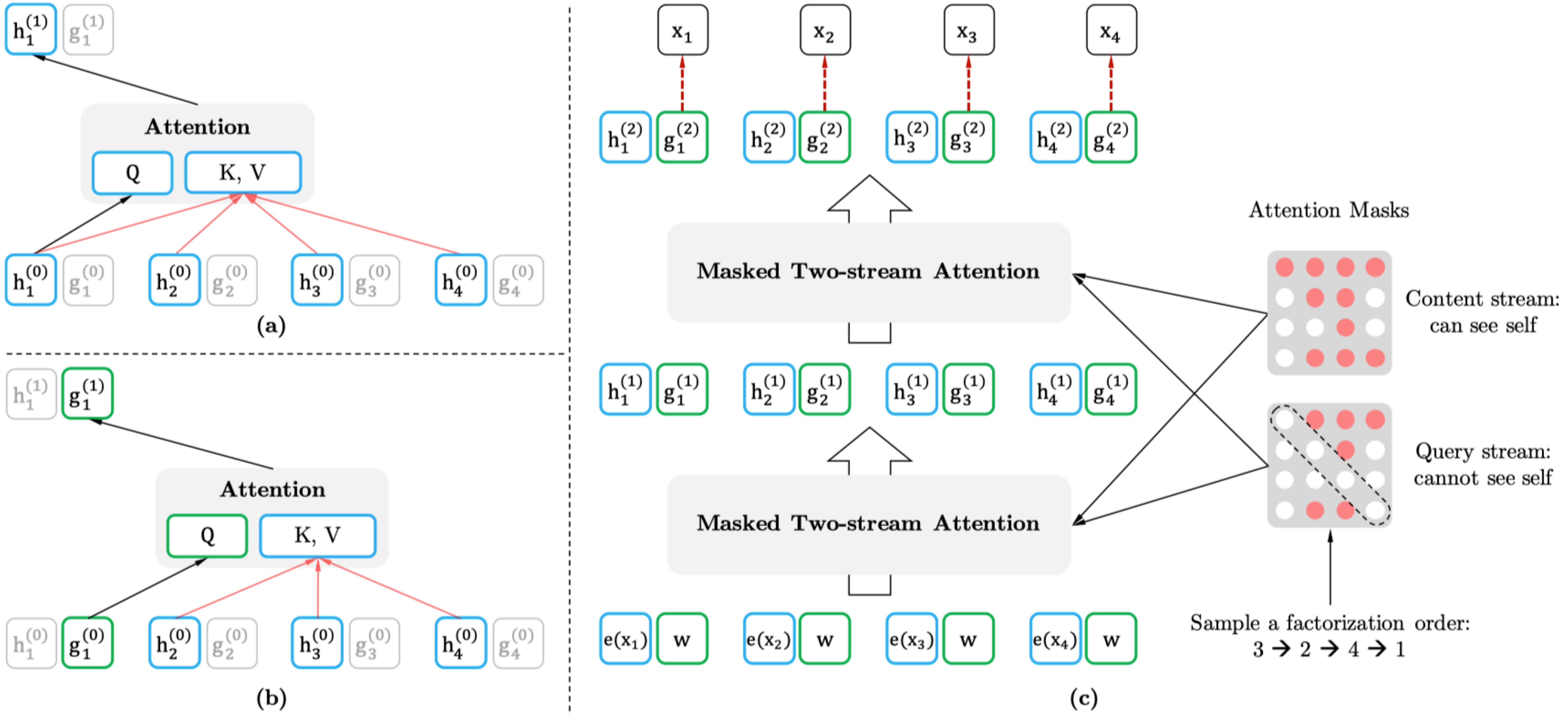
Bidirectional Context



***Yang et al., NeurIPS 2020.** XLNet: Generalized Autoregressive Pretraining for Language Understanding*

XLNet

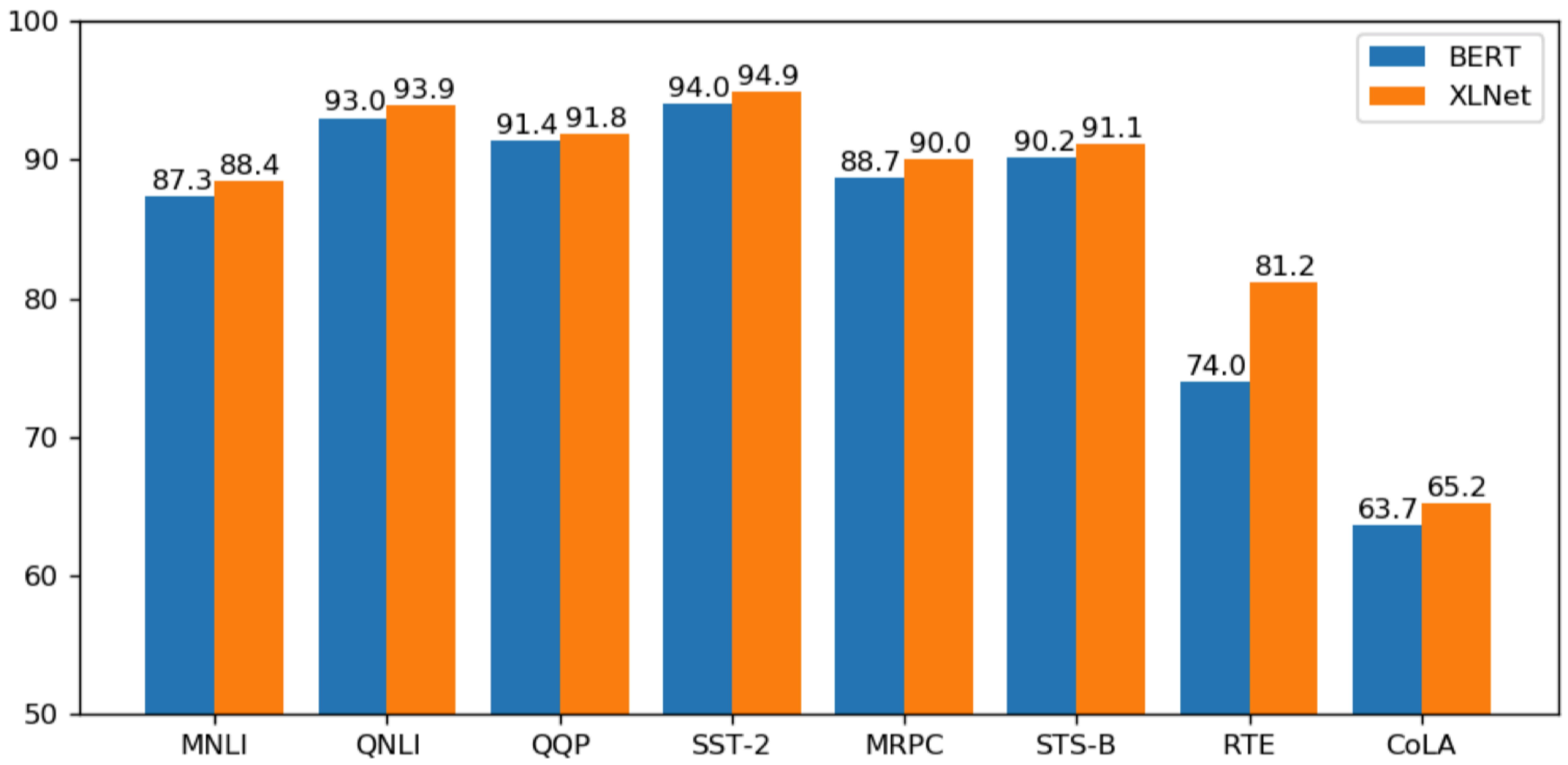
- **Two-Stream Self-Attention**
  - Content stream attention and Query stream attention

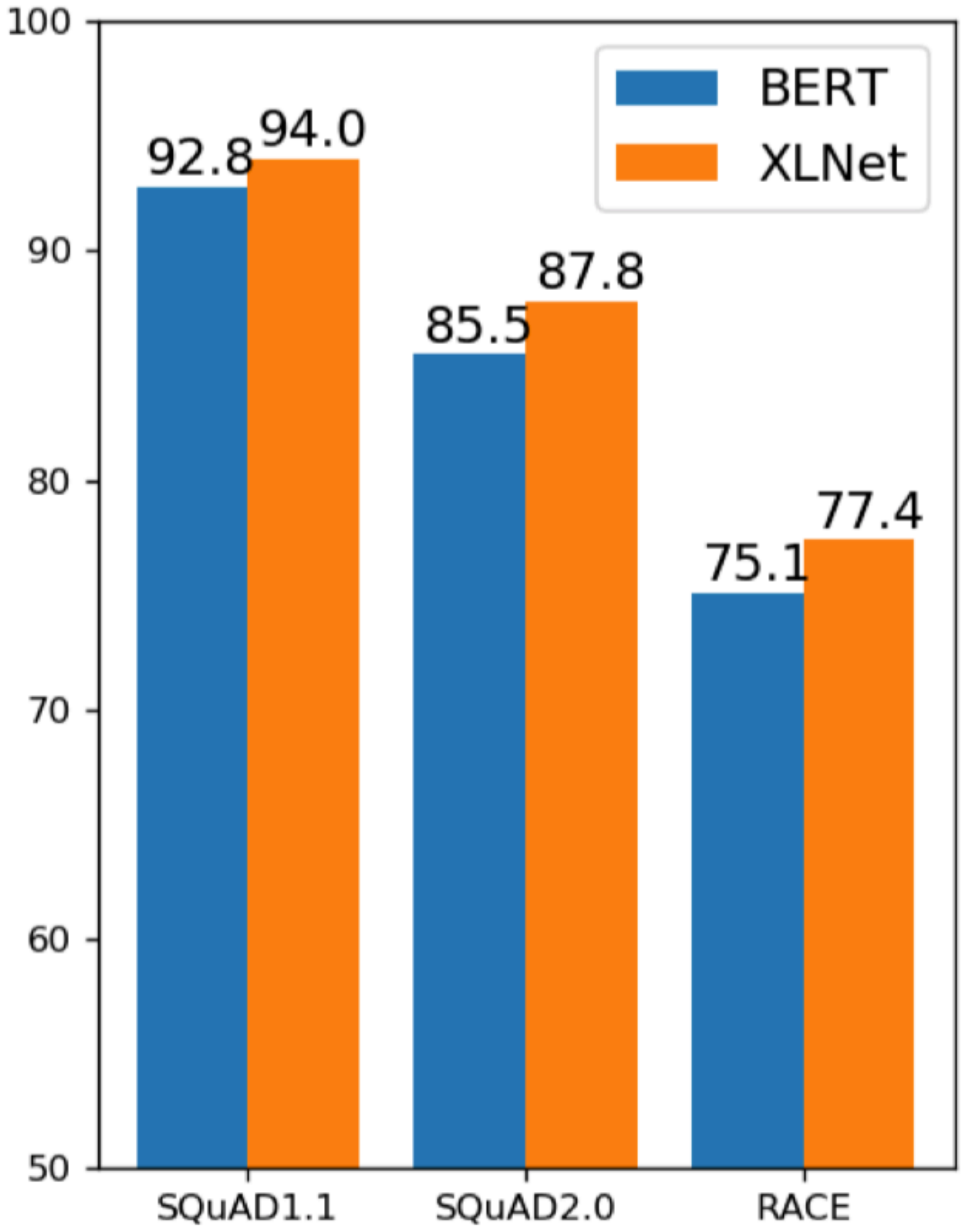


XLNet

- **A Fair Comparison with BERT**
  - XLNet yields better performance over BERT under a **comparable** setting

Read more





Yang et al., NeurIPS 2020. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

RoBERTa

- **RoBERTa: R**obustly **o**ptimized **BERT** pretraining **a**pproach
  - Investigate important choices in BERT design, such as masking strategies, the use of next sentence prediction, etc.
  - Propose CC-News dataset, confirming that more data will lead to better performance

RoBERTa: A Robustly Optimized BERT Pretraining Approach

Yinhan Liu[*§] **Myle Ott**[*§] **Naman Goyal**[*§] **Jingfei Du**[*§] **Mandar Joshi**[†]
Danqi Chen[§] **Omer Levy**[§] **Mike Lewis**[§] **Luke Zettlemoyer**[†§] **Veselin Stoyanov**[§]

[†] Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA
{mandar90,lsz}@cs.washington.edu

[§] Facebook AI
{yinhanliu,myleott,naman,jingfeidu,
danqi,omerlevy,mikelewis,lsz,ves}@fb.com

Liu et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

RoBERTa

- **Static Masking vs. Dynamic Masking**
  - Increasing the randomness of the masking tokens
  - Static: Masking pattern is determined **AFTER** pre-processing
  - Dynamic: Masking pattern is determined **DURING** pre-training

Epoch

Static Masking	Dynamic Masking
went to the store → went to the [MASK]	went to the store → went to the [MASK]
went to the store → went to the [MASK]	went to the store → went to [MASK] store
went to the store → went to the [MASK]	went to the store → [MASK] to the store
went to the store → went to the [MASK]	went to the store → went to the store
went to the store → went to the [MASK]	went to the store → went [MASK] the store

Static Masking **Dynamic Masking**

Liu et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

RoBERTa

- **Necessity of NSP Task**
  - Removing NSP task yields marginal improvements

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

Original BERT implementation → SEGMENT-PAIR
Natural sentences → SENTENCE-PAIR
Could cross the document boundary → FULL-SENTENCES
Could NOT cross the document boundary → DOC-SENTENCES

Liu et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

RoBERTa

- **Larger Batches with More Data**
  - If possible, use a larger batch with more data
  - A proper extension to the pre-training steps also improves the performance
  - It has been widely proven that using a larger batch is ESSENTIAL for pre-training 💡

batch size	learning rate	epochs	steps	perplexity	MNLI-m	SST-2
256	1e-4	32	1M	3.99	84.7	92.5
2K	7e-4	32	125K	3.68	85.2	93.1
		64	250K	3.59	85.3	**94.1**
		128	500K	3.51	85.4	93.5
8K	1e-3	32	31K	3.77	84.4	93.2
		64	63K	3.60	85.3	93.5
		128	125K	**3.50**	**85.8**	**94.1**

Liu et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

RoBERTa

- **Final Choices for RoBERTa: Sum Up All Good Things**
  - **Pre-training Tasks**
    - Dynamic Masking
    - Full-Sentences without NSP loss
  - **Pre-training Setups**
    - Large mini-batches: 256 → 8192
    - Large byte-level BPE: 30k → 50K

Liu et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

RoBERTa

- **Experimental Results**
  - Comparable Setting: XLNet > RoBERTa > BERT
  - Training even longer may further improve the performance of RoBERTa

Model	data	batch size	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 et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

ALBERT

- **ALBERT: A Lite BERT** for Self-supervised Learning of Language Representations
  - Aims to provide a **parameter-compact** BERT
  - Two techniques are proposed: factorized embedding parameterization, cross-layer parameter sharing

ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

Zhenzhong Lan[1] **Mingda Chen**[2*] **Sebastian Goodman**[1] **Kevin Gimpel**[2]
Piyush Sharma[1] **Radu Soricut**[1]
[1]Google Research [2]Toyota Technological Institute at Chicago
{lanzhzh, seabass, piyushsharma, rsoricut}@google.com
{mchen, kgimpel}@ttic.edu

Lan et al., ICLR 2020. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*

ALBERT

- **Factorized Embedding Parameterization**
  - In original BERT, `embedding_size` == `hidden_size`
  - With FEP,

$$O(V \times H) \rightarrow O(V \times E + E \times H)$$

  - For example, $V = 30{,}000,\ H = 1024,\ E = 128$

BERT > **ALBERT**

$V*H = 30000*1024=\mathbf{30{,}720{,}000}$

$V*E+E*H = 30000*128+128*1024=\mathbf{3{,}971{,}072}$

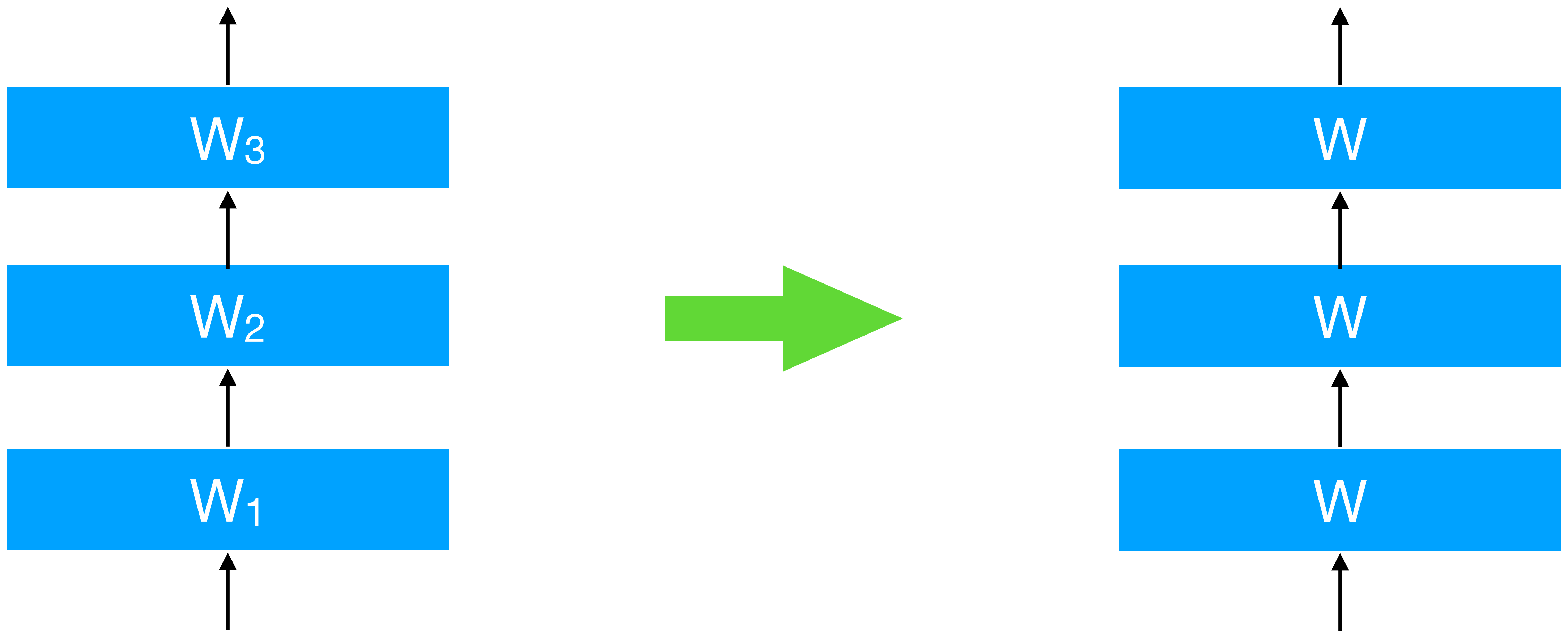
Lan et al., ICLR 2020. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*

ALBERT

- **Cross-Layer Parameter Sharing**
  - The weight for each Transformer layer is **shared**
  - Parameter sharing is parameter efficient but NOT memory efficient!

This will NOT save your GPU/TPU memory!

Total Params = $W_1 + W_2 + W_3$
Disk Usage = $W_1 + W_2 + W_3$
Memory Usage = $W_1 + W_2 + W_3$



Total Params = W
Disk Usage = W
Memory Usage = 3W

Lan et al., ICLR 2020. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*

ALBERT

- **Sentence Order Prediction (SOP)**
  - NSP = coherence prediction + topic prediction
  - But the topic prediction is quite easy (i.e. NSP)
  - In SOP,
    - Positive examples: same as BERT, two consecutive text segments

+	Sentence 1	Sentence 2

    - Negative examples: **swapped** two consecutive text segments

−	Sentence 2	Sentence 1

Lan et al., ICLR 2020. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*

ALBERT

让世界聆听我们的声音

- **Effectiveness of Each Component**
  - ALBERT is much compact in size (parameters) but **NOT** in speedup
    - ALBERT-large ≈ BERT-base
    - ALBERT-xlarge ≈ BERT-large
  - ALBERT-xxlarge yields the best performance
  - Parameter sharing and embedding decomposition **HURTS** performance
  - SOP task yields better performance than NSP/None

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

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

Table 3: The effect of vocabulary embedding size on the performance of ALBERT-base.

	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

Table 5: The effect of sentence-prediction loss, NSP vs. SOP, on intrinsic and downstream tasks.

***Lan et al., ICLR 2020.** ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*

ELECTRA

- **ELECTRA: E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
  - A new generator-discriminator framework for pre-trained language model
  - Efficient training scheme that achieves much quicker pre-training

ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS

Kevin Clark
Stanford University
kevclark@cs.stanford.edu

Minh-Thang Luong
Google Brain
thangluong@google.com

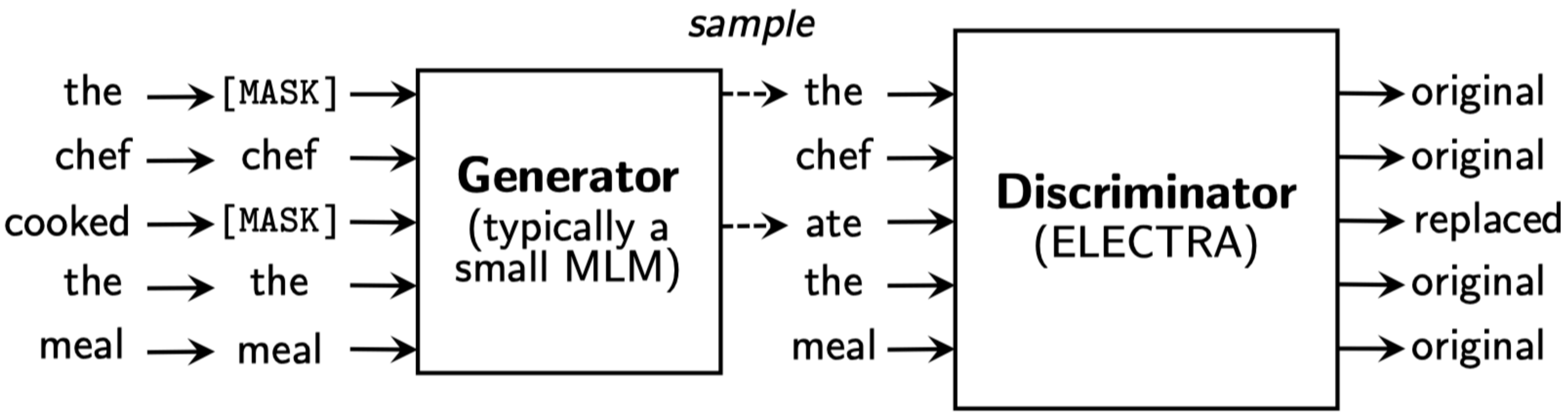
Quoc V. Le
Google Brain
qvl@google.com

Christopher D. Manning
Stanford University & CIFAR Fellow
manning@cs.stanford.edu

Clark et al., ICLR 2020. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

ELECTRA

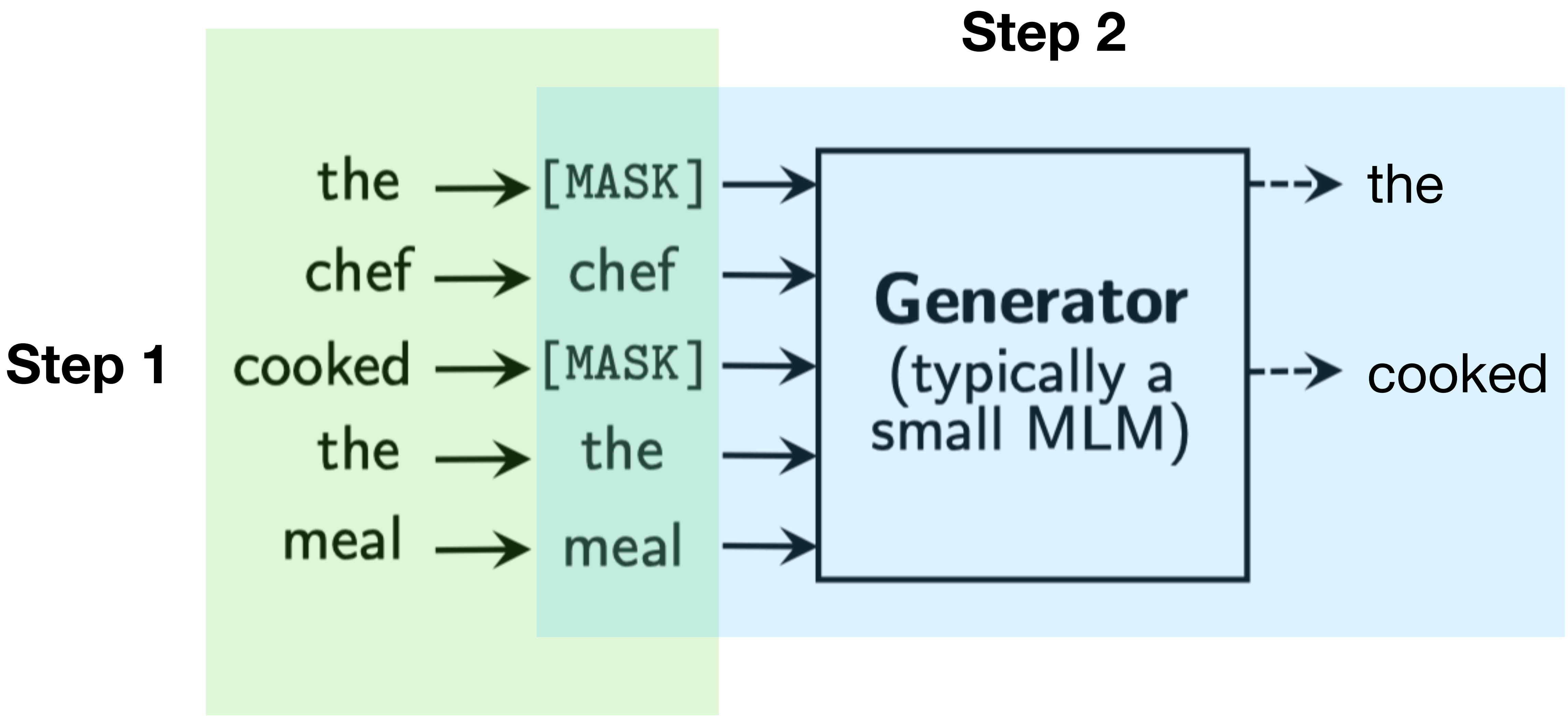
- **Overall Architecture**
  - Generator-Discriminator structure, similar to GAN (Goodfellow et al., 2014)
  - **Generator**: a small MLM that learns to predict the original words of masked tokens
  - **Discriminator**: discriminate whether the input token in replaced by generator



Clark et al., ICLR 2020. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

ELECTRA

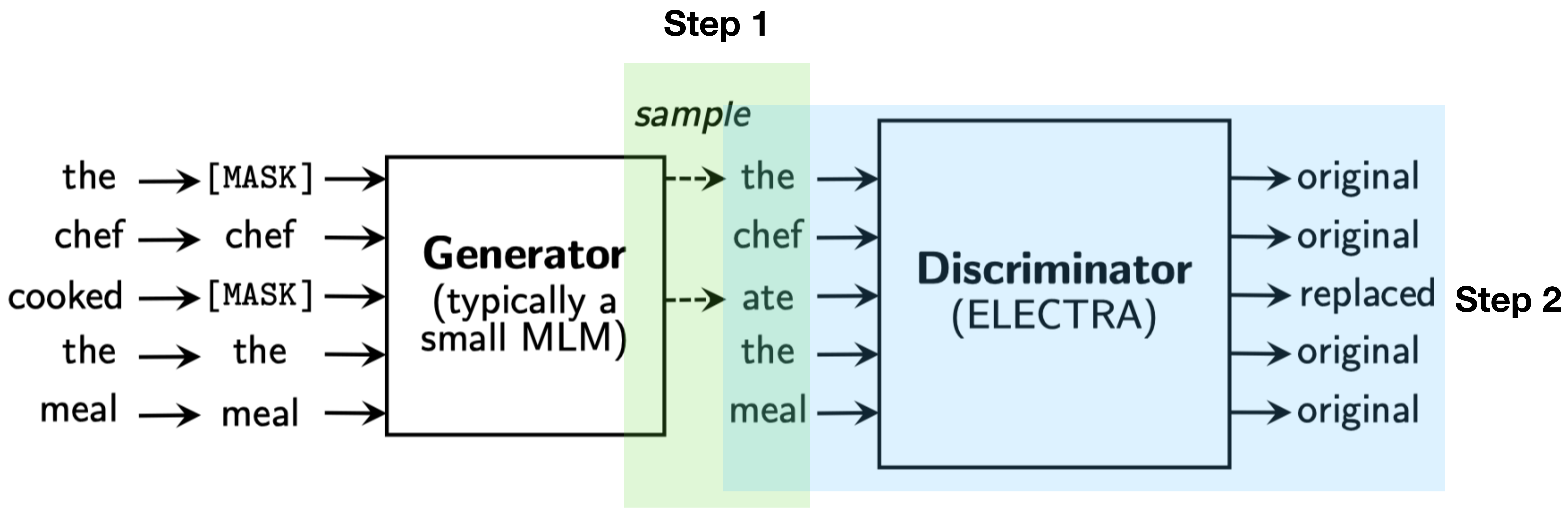
- **Generator: a small MLM**
  - Step 1: mask out a random (15%) set of positions in the input sequence
  - Step 2: learns to recover the original words



Clark et al., ICLR 2020. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

ELECTRA

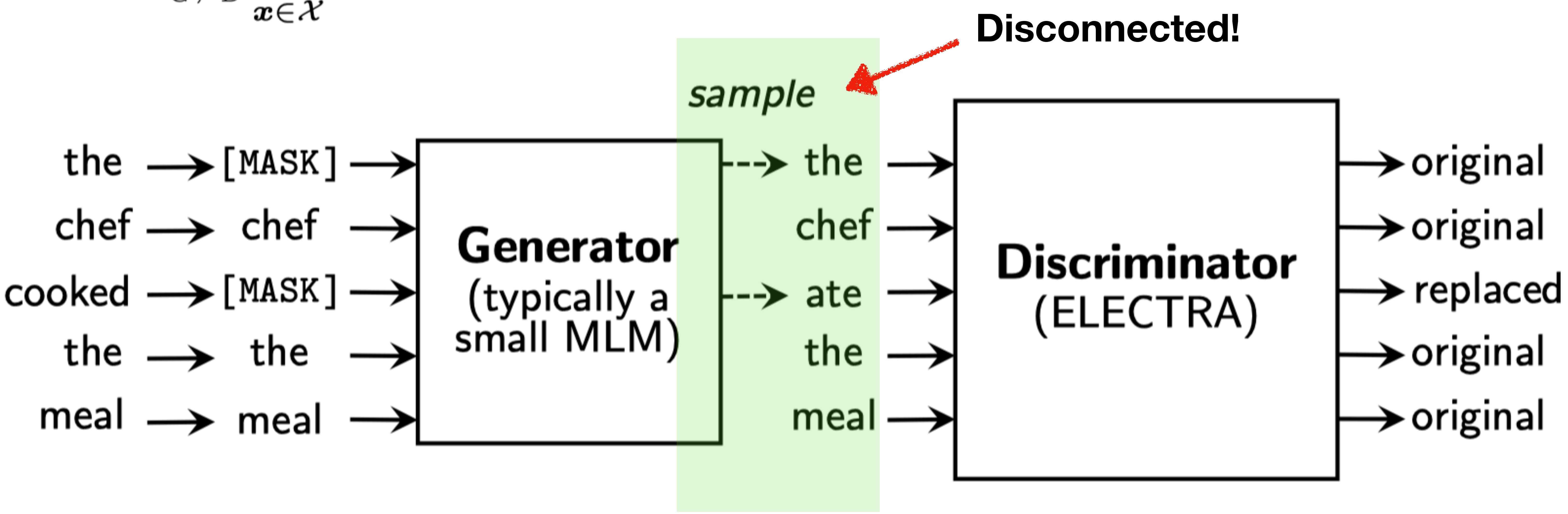
- **Discriminator: regular BERT**
  - Step 1: replace masked tokens with generated tokens
  - Step 2: discriminate whether the token is replaced



Clark et al., ICLR 2020. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

ELECTRA

- **ELECTRA is NOT trained like a GAN**
    - It is impossible to BP through sampling from generator
    - They tried to use reinforcement learning (RL) but it results in a worse performance
    - Final loss is $\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\mathrm{MLM}}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{\mathrm{Disc}}(\boldsymbol{x}, \theta_D)$



Clark et al., ICLR 2020. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

ELECTRA

- **Experimental Results**
  - ELECTRA-small/base yields significant improvements over BERT counterparts

Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

Clark et al., ICLR 2020. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

ELECTRA

- **Experimental Results**
  - ELECTRA-large results on GLUE-dev/test

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	92.4	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

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**

Clark et al., ICLR 2020. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

Summary

	GPT	BERT	XLNet	RoBERTa	ALBERT	ELECTRA
Type	AR	AE	AR	AE	AE	AE
Embedding	T	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P
Masking	/	T	/	T	T	T
LM Task	LM	MLM	PLM	MLM	MLM	Gen-Dis
Paired Task	/	NSP		/	SOP	/
Data Source	BC	BC+Wiki	BC+Wiki+Giga5 +CW+CC	BC+Wiki+CCNews +OWT+Stories	BC+Wiki	BC+Wiki+Giga5 +CW+CC
Data Size	/	/	110G	160G	16G	~110G
Tokenization	BPE	WordPiece	SentencePiece	BPE	SentencePiece	WordPiece
# Tokens	800M	3300M	32.89B	/	/	~33B
# Vocabulary	40,000	30,522	32,000	50,000	30,000	30,522
# MaxSeqLen	512	512	512	512	512	512
# Layers	12	12/24	12/24	12/24	12/24/24/12	12/12/24

中文预训练语言模型

CHINESE PRE-TRAINED LANGUAGE MODELS

1. BERT-wwm
2. ERNIE
3. NEZHA
4. ZEN
5. MacBERT

中文BERT-wwm

- **Pre-Training with Whole Word Masking for Chinese BERT**
  - We adapt whole word masking strategy in Chinese context
  - We also compare the state-of-the-art Chinese pre-trained models in detail, including BERT, ERNIE, BERT-wwm

Pre-Training with Whole Word Masking for Chinese BERT

Yiming Cui[†‡*], Wanxiang Che[†], Ting Liu[†], Bing Qin[†], Ziqing Yang[‡], Shijin Wang[‡], Guoping Hu[‡]
[†]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[‡]Joint Laboratory of HIT and iFLYTEK (HFL), iFLYTEK Research, Beijing, China
[‡]iFLYTEK Hebei AI Research, Hebei, China

Cui et al., arXiv 2019. *Pre-Training with Whole Word Masking for Chinese BERT*

中文BERT-wwm

- **Chinese BERT with Whole Word Masking**
  - Chinese word is comprised of characters
  - We use LTP for Chinese Word Segmentation (CWS) to detect **word boundary**

[Original Sentence]
使用语言模型来预测下一个词的probability。
[Original Sentence with CWS]
使用 语言 **模型** 来 **预测** 下 一个 词 的 **probability** 。

[Original BERT Input]
使 用 语 言 **[MASK] 型** 来 **[MASK] 测** 下 一 个 词 的 **pro [MASK] ##lity** 。
[Whold Word Masking Input]
使 用 语 言 **[MASK] [MASK]** 来 **[MASK] [MASK]** 下 一 个 词 的 **[MASK] [MASK] [MASK]** 。

Remember: [MASK] could also be ‘replace by another word’ or ‘keep original word’

Cui et al., arXiv 2019. *Pre-Training with Whole Word Masking for Chinese BERT*

ERNIE

- **ERNIE: E**nhanced **R**epresentation through k**N**owledge **I**nt**E**gration
  - Masking units instead of only tokens
    - Phrase-level masking
    - Entity-level masking
  - Over 173M sentences for pre-training

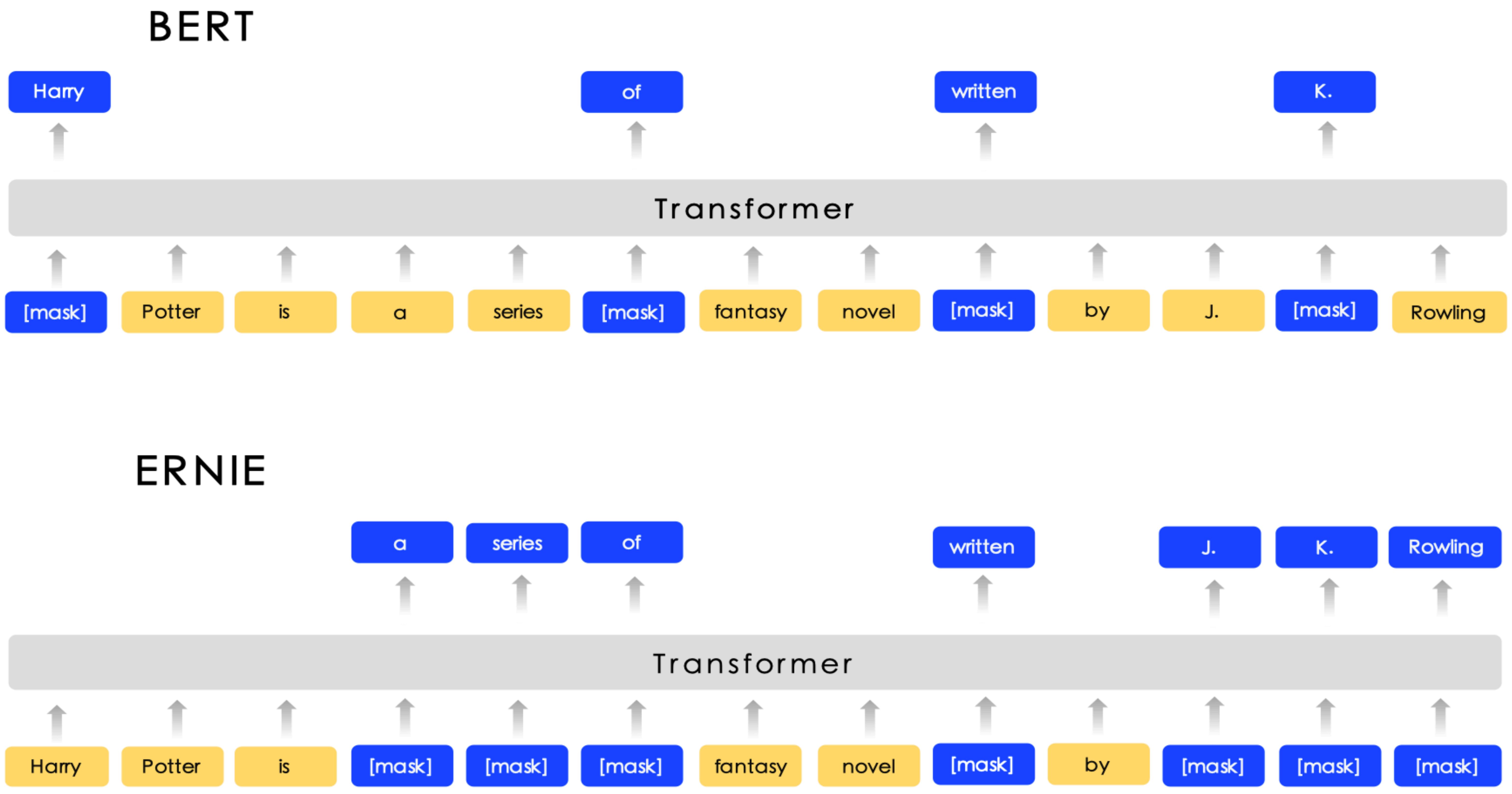
ERNIE: Enhanced Representation through Knowledge Integration

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng
Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, Hua Wu
Baidu Inc.
{sunyu02,wangshuohuan,liyukun01,fengshikun01,tianhao,wu_hua}@baidu.com

Sun et al., arXiv 2019. *ERNIE: Enhanced Representation through Knowledge Integration*

ERNIE

- **Comparisons: BERT vs. ERNIE**



Sun et al., arXiv 2019. *ERNIE: Enhanced Representation through Knowledge Integration*

ERNIE

- **Basic-Level Masking**
  - 15% basic language units are masked.
- **Phrase-Level Masking**
  - Consecutive words are masked. The phrase boundary is identified by lexical analysis and chunking tools.
- **Entity-Level Masking**
  - Mask named entity, such as person names, locations, organizations, etc.

Sentence	Harry	Potter	is	a	series	of	fantasy	novels	written	by	British	author	J.	K.	Rowling
Basic-level Masking	[mask]	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	J.	[mask]	Rowling
Entity-level Masking	Harry	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]
Phrase-level Masking	Harry	Potter	is	[mask]	[mask]	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]

***Sun et al., arXiv 2019.** ERNIE: Enhanced Representation through Knowledge Integration*

ERNIE

- **ERNIE 2.0: A Continual Pre-training Framework for Language Understanding**
  - A continual pre-training framework in an incremental way
  - A bunch of new unsupervised pre-training tasks
  - Pre-training data size: 7378M tokens

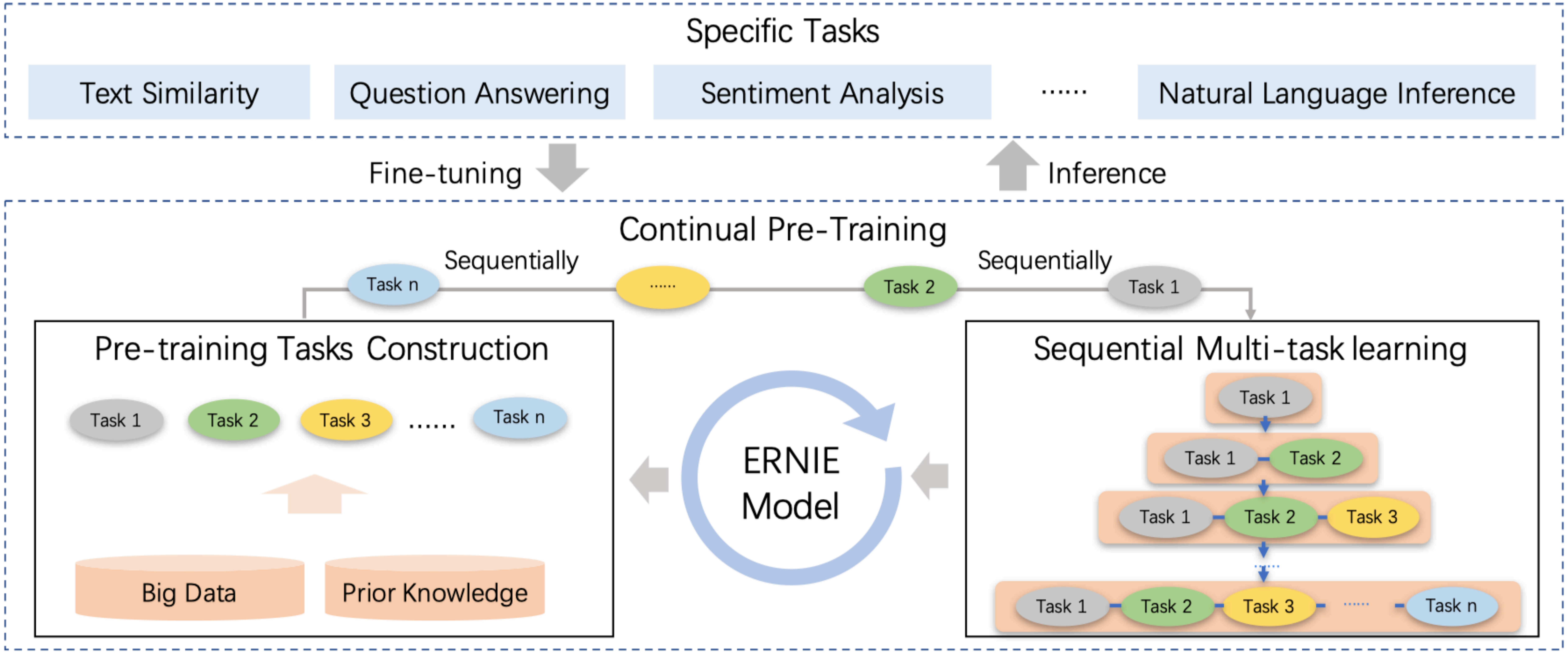
ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, Haifeng Wang
Baidu Inc., Beijing, China
{sunyu02, wangshuohuan, tianhao, wu_hua,wanghaifeng}@baidu.com

Sun et al., AAAI 2020. *ERNIE 2.0: A Continual Pre-training Framework for Language Understanding*

ERNIE

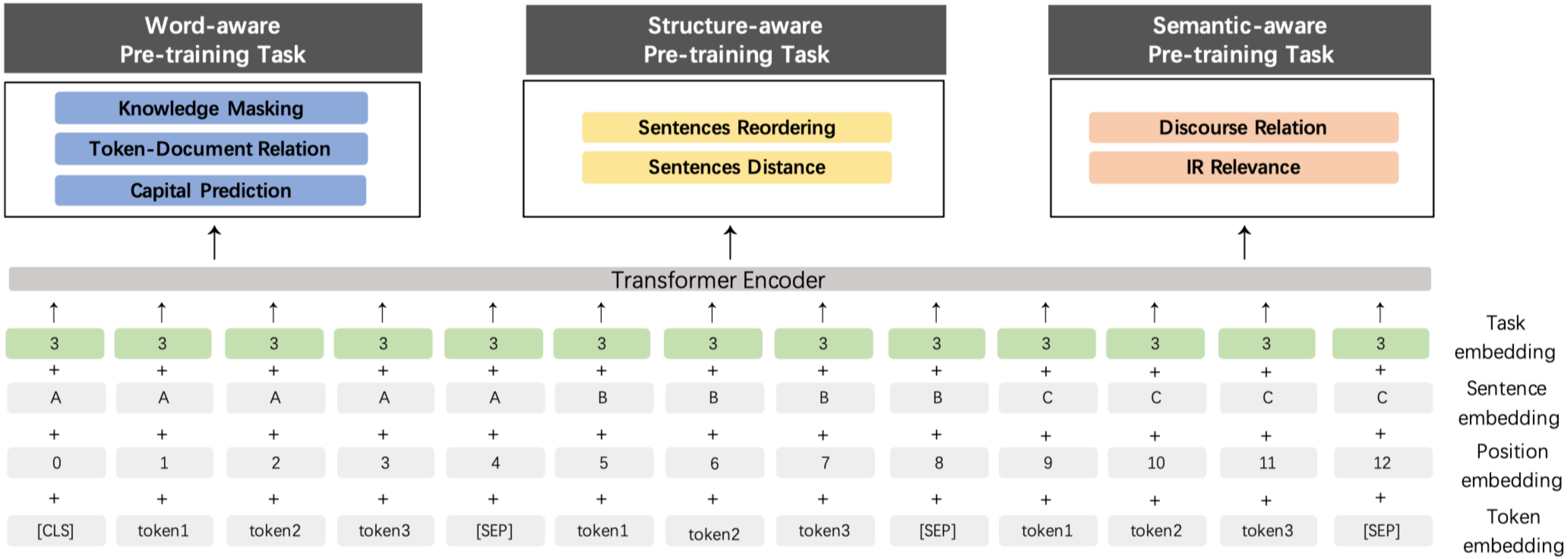
- **Overall Framework**



Sun et al., AAAI 2020. *ERNIE 2.0: A Continual Pre-training Framework for Language Understanding*

ERNIE

- **Overall Framework**



Sun et al., AAAI 2020. *ERNIE 2.0: A Continual Pre-training Framework for Language Understanding*

NEZHA

- **NEZHA**: **NE**ural Contextuali**Z**ed Representation for C**H**inese **LA**nguage Understanding
    - Propose a relative position encoding scheme for attention calculation
    - Pre-training data size: 202M (Wiki) + 4734M (Baike) + 5600M (news) ≈ 10B tokens

NEZHA: NEURAL CONTEXTUALIZED REPRESENTATION FOR CHINESE LANGUAGE UNDERSTANDING

TECHNICAL REPORT

Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin*, Xin Jiang, Xiao Chen, Qun Liu
Noah's Ark Lab, *HiSilicon, Huawei Technologies
{wei.junqiu1, renxiaozhe, lixiaoguang11, wenyong.huang, liao.yi, wangyasheng, linjiashu, jiang.xin, chen.xiao2, qun.liu}@huawei.com

Wei et al., arXiv 2019. *NEZHA: Neural Contextualized Representation for Chinese Language Understanding*

NEZHA

- **Training Phase: Functional Relative Positional Encoding**
  - Relational position information is considered during attention calculation

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V).$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_k \exp e_{ik}},$$

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}}.$$

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + \boxed{a_{ij}^V})$$

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K + \boxed{a_{ij}^K})^T}{\sqrt{d_z}}.$$

$$a_{ij}[2k] = \sin((j-i)/(10000^{\frac{2\cdot k}{d_z}})),$$

$$a_{ij}[2k+1] = \cos((j-i)/(10000^{\frac{2\cdot k}{d_z}})).$$

Wei et al., arXiv 2019. *NEZHA: Neural Contextualized Representation for Chinese Language Understanding*

NEZHA

- **Training Phase: Other Features**
  - Whole word masking
  - Mixed precision training
    - Also known as using FP16
  - LAMB optimizer
    - Better scalability for large batch training

Wei et al., arXiv 2019. *NEZHA: Neural Contextualized Representation for Chinese Language Understanding*

NEZHA

- **Experimental Results**

Model	CMRC		XNLI		LCQMC		PD-NER		ChnSenti	
	EM	F1	Dev	Test	Dev	Test	Dev	Test	Dev	Test
BASE MODELS										
$\text{BERT}_{\text{BASE}}$	64.06	85.01	78.75	77.27	89.04	87.61	96.53	98.58	94.91	95.42
$\text{BERT}_{\text{BASE}}$-WWM	64.96	85.79	78.79	78.44	89.19	87.16	96.86	98.58	94.67	94.58
$\text{BERT}_{\text{BASE}}$-WWM (in [8])	66.30	85.60	79.00	78.20	89.40	87.00	95.30	65.10	95.10	95.40
$\text{ERNIE-Baidu}_{\text{BASE}}$ 1.0 (in [3])	65.10	85.10	79.9	78.4	89.70	87.40	-	-	95.20	95.40
$\text{ERNIE-Baidu}_{\text{BASE}}$ 2.0 (in [4])	**69.10**	**88.60**	81.20	**79.70**	**90.90**	**87.90**	-	-	**95.70**	95.50
$\text{NEZHA}_{\text{BASE}}$ (ours)	67.07	86.35	**81.37**	79.32	89.98	87.41	97.22	**98.58**	94.74	95.17
$\text{NEZHA}_{\text{BASE}}$-WWM (ours)	67.82	86.25	81.25	79.11	89.85	87.10	**97.41**	98.35	94.75	**95.84**
LARGE MODELS										
$\text{ERNIE-Baidu}_{\text{LARGE}}$ 2.0 (in [4])	**71.50**	**89.90**	**82.60**	81.00	**90.90**	87.90	-	-	**96.10**	95.80
$\text{NEZHA}_{\text{LARGE}}$ (ours)	68.10	87.20	81.53	80.44	90.18	87.20	**97.51**	**97.87**	95.92	95.83
$\text{NEZHA}_{\text{LARGE}}$-WWM (ours)	67.32	86.62	82.21	**81.17**	90.87	**87.94**	97.26	97.63	95.75	**96.00**

Wei et al., arXiv 2019. *NEZHA: Neural Contextualized Representation for Chinese Language Understanding*

ZEN

- **ZEN**: Pre-training Chinese (**Z**) Text Encoder **E**nhanced by **N**-gram Representations
    - Using N-gram information to enhance the text encoder
    - Pre-training data size: 474M tokens (Chinese Wikipedia)

ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations

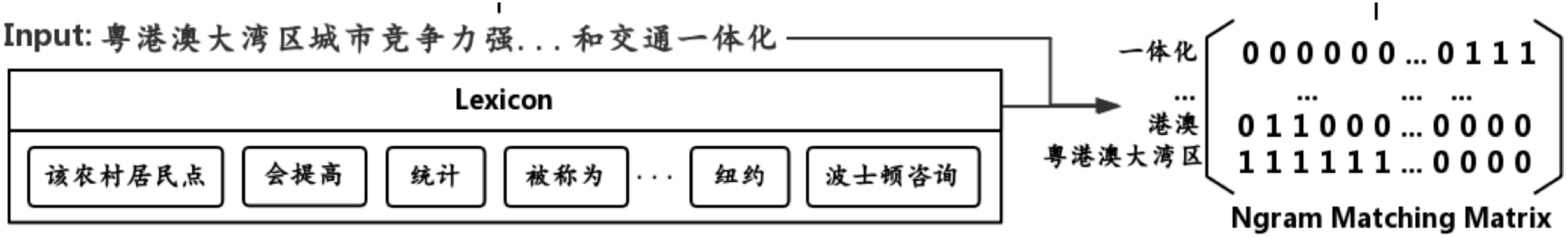
Shizhe Diao[♡*], **Jiaxin Bai**[♡*], **Yan Song**[♠], **Tong Zhang**[♡], **Yonggang Wang**[♠]
[♡]The Hong Kong University of Science and Technology
{sdiaoaa, jbai, tongzhang}@ust.hk
[♠]Sinovation Ventures
{songyan, wangyonggang}@chuangxin.com

Diao et al., arXiv 2019. *ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations*

ZEN

- **Training Phase: N-gram Extraction**
  - Prepare an N-gram lexicon
  - N-gram extraction during pre-training

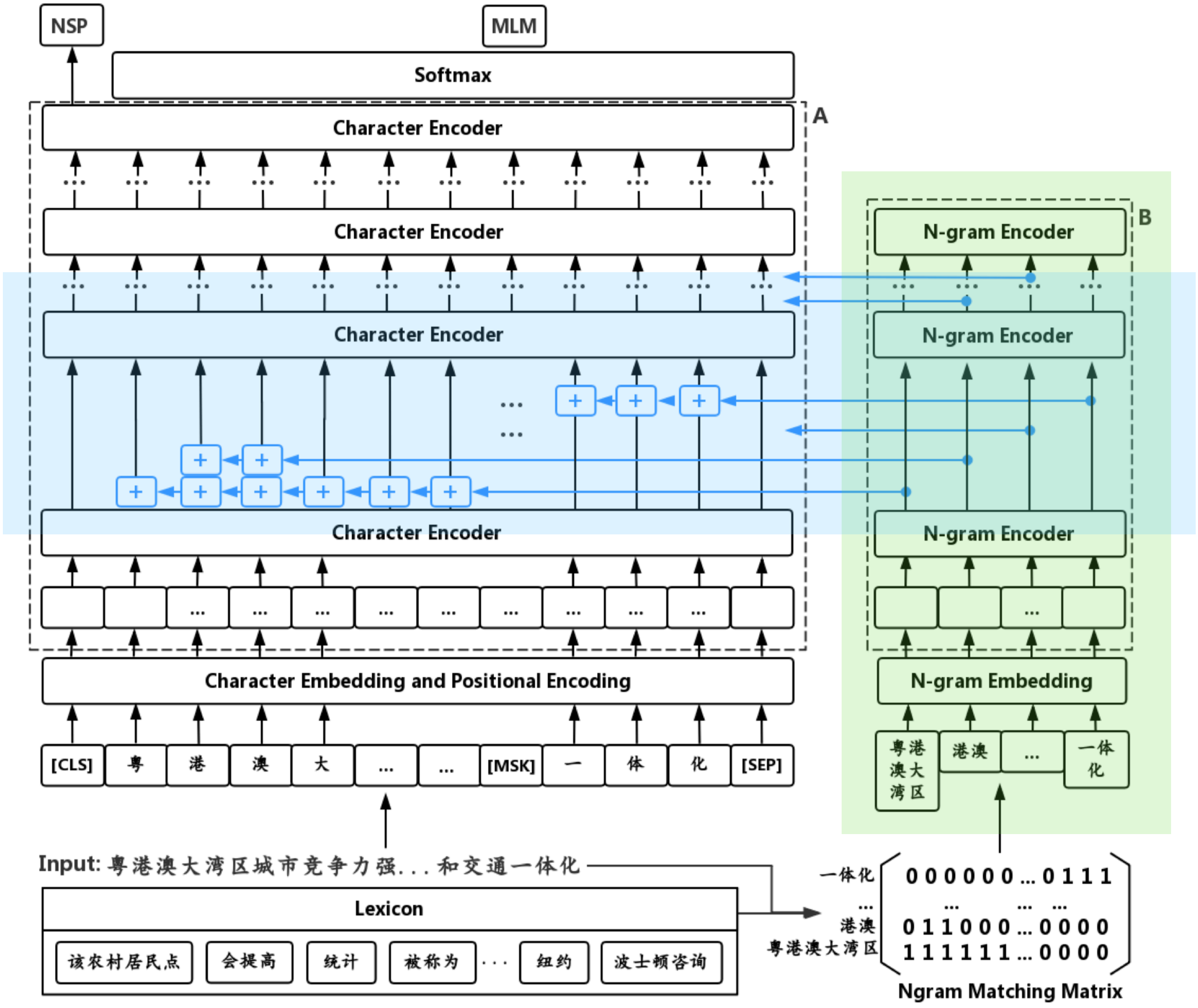
$$m_{ij} = \begin{cases} 1 & c_i \in n_j \\ 0 & c_i \notin n_j \end{cases},$$



***Diao et al., arXiv 2019.** ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations*

ZEN

- **Training Phase: Encoding N-grams (B)**
  - Using an N-gram embedding matrix to project N-grams into embedding representation
  - Transformer-based N-gram encoder
- **Training Phase: Representing N-grams (A)**
  - Add N-gram representation back to original BERT w.r.t. each token in N-gram
  - Layer-by-layer addition



***Diao et al., arXiv 2019.** ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations*

ZEN

- **Experimental Results**
  - Sequence labeling tasks yield better performance than classification tasks

	CWS	POS		NER	DC		SA		SPM		NLI	
	TEST	DEV	TEST	TEST	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
BERT (R)	97.20	95.72	95.43	93.12	96.90	96.71	94.00	94.10	87.22	85.13	75.67	75.01
BERT (P)	97.95	96.30	96.10	94.78	97.60	97.50	94.53	94.67	88.50	86.59	77.40	77.52
BERT-WWM	-	-	-	95.10	97.60	97.60	94.50	95.00	89.20	86.80	78.40	78.00
ERNIE 1.0	-	-	-	95.10	97.30	97.30	95.20	95.40	89.70	87.40	79.90	78.40
ERNIE 2.0 (B)	-	-	-	-	-	-	**95.70**	95.50	**90.90**	87.90	81.20	**79.70**
NEZHA (B)	-	-	-	-	-	-	94.74	95.17	89.98	87.41	**81.37**	79.32
NEZHA-WWM (B)	-	-	-	-	-	-	94.75	95.84	89.85	87.10	81.25	79.11
ERNIE 2.0 (L)	-	-	-	-	-	-	*96.10*	*95.80*	*90.90*	*87.90*	*82.60*	*81.00*
NEZHA (L)	-	-	-	-	-	-	*95.92*	*95.83*	*90.18*	*87.20*	*81.53*	*80.44*
NEZHA-wwm (L)	-	-	-	-	-	-	*95.75*	*96.00*	*90.87*	*87.94*	*82.21*	*81.17*
ZEN (R)	97.89	96.12	95.82	93.24	97.20	96.87	94.87	94.42	88.10	85.27	77.11	77.03
ZEN (P)	**98.35**	**97.43**	**96.64**	**95.25**	**97.66**	**97.64**	95.66	**96.08**	90.20	**87.95**	80.48	79.20

R: random
P: from pre-train

Diao et al., arXiv 2019. *ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations*

MacBERT

- **MacBERT: M**LM **a**s **c**orrection **BERT**
  - Evaluate state-of-the-art PLMs in Chinese with relatively comparable settings
  - Propose a new PLM called MacBERT
  - Create a series of Chinese PLMs and open-source to the community

Revisiting Pre-trained Models for Chinese Natural Language Processing

Yiming Cui[1,2], Wanxiang Che[1], Ting Liu[1], Bing Qin[1], Shijin Wang[2,3], Guoping Hu[2]
[1]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[3]iFLYTEK AI Research (Hebei), Langfang, China
[1]{ymcui,car,tliu,qinb}@ir.hit.edu.cn
[2,3]{ymcui,sjwang3,gphu}@iflytek.com

Cui et al., Findings of EMNLP 2020. *Revisiting Pre-trained Models for Chinese Natural Language Processing*

MacBERT

- **MLM as correction**
  - Solve the discrepancy of pre-training and fine-tuning: using the similar word
  - Other techniques: whole word masking, N-gram masking

用 语 言 模 型 **预 测** 下 一 个 词

BERT

- 80% of the time, replace with [M]
  - 用 语 言 模 型 **[M] [M]** 下 一 个 词
- 10% of the time, replace random word
  - 用 语 言 模 型 **预 见** 下 一 个 词
- 10% of the time, keep the same word
  - 用 语 言 模 型 **预 测** 下 一 个 词

MACBERT

- 80% of the time, replace with [M]
  - 用 语 言 模 型 **预 见** 下 一 个 词
- 10% of the time, replace random word
  - 用 语 言 模 型 **好 是** 下 一 个 词
- 10% of the time, keep the same word
  - 用 语 言 模 型 **预 测** 下 一 个 词

Cui et al., Findings of EMNLP 2020. *Revisiting Pre-trained Models for Chinese Natural Language Processing*

MacBERT

- **Comparisons of PLMs**

	BERT	ERNIE	XLNet	RoBERTa	ALBERT	ELECTRA	MacBERT
Type	AE	AE	AR	AE	AE	AE	AE
Embeddings	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P
LM Task	MLM	MLM	PLM	MLM	MLM	Gen-Dis	Mac
Masking	T	T/E/Ph	-	T	T	T	WWM/NM
Paired Task	NSP	NSP	-	-	SOP	-	SOP

Cui et al., Findings of EMNLP 2020. *Revisiting Pre-trained Models for Chinese Natural Language Processing*

MacBERT

- **Experimental Results**
  - Significant improvements on machine reading comprehension tasks
  - Moderate improvements on classification tasks

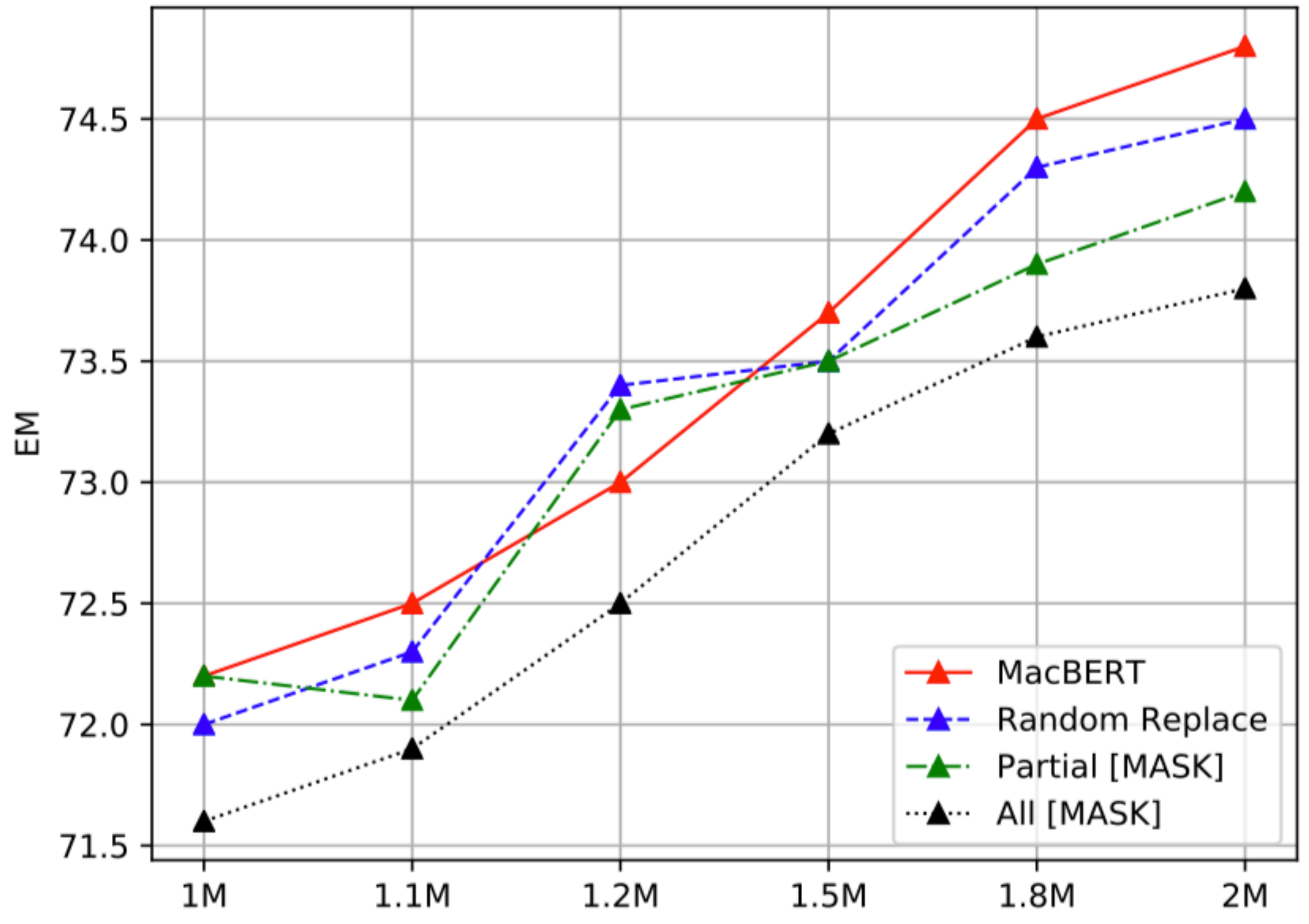
CMRC 2018	Dev EM	Dev F1	Test EM	Test F1	Challenge EM	Challenge F1
BERT	65.5 (64.4)	84.5 (84.0)	70.0 (68.7)	87.0 (86.3)	18.6 (17.0)	43.3 (41.3)
BERT-wwm	66.3 (65.0)	85.6 (84.7)	70.5 (69.1)	87.4 (86.7)	21.0 (19.3)	47.0 (43.9)
BERT-wwm-ext	67.1 (65.6)	85.7 (85.0)	71.4 (70.0)	87.7 (87.0)	24.0 (20.0)	47.3 (44.6)
RoBERTa-wwm-ext	67.4 (66.5)	87.2 **(86.5)**	72.6 (71.4)	89.4 (88.8)	26.2 (24.6)	51.0 (49.1)
ELECTRA-base	68.4 **(68.0)**	84.8 (84.6)	73.1 **(72.7)**	87.1 (86.9)	22.6 (21.7)	45.0 (43.8)
MacBERT-base	**69.5** (67.3)	**87.7 (86.5)**	**73.3** (72.0)	**89.6 (89.1)**	**27.5 (25.6)**	**53.7 (50.2)**
ELECTRA-large	69.1 (68.2)	85.2 (84.5)	73.9 (72.8)	87.1 (86.6)	23.0 (21.6)	44.2 (43.2)
RoBERTa-wwm-ext-large	68.5 (67.6)	88.4 (87.9)	74.2 (72.4)	90.6 (90.0)	31.5 **(30.1)**	60.1 (57.5)
MacBERT-large	**70.7 (68.6)**	**88.9 (88.2)**	**74.8 (73.2)**	**90.7 (90.1)**	**31.9** (29.6)	**60.2 (57.6)**

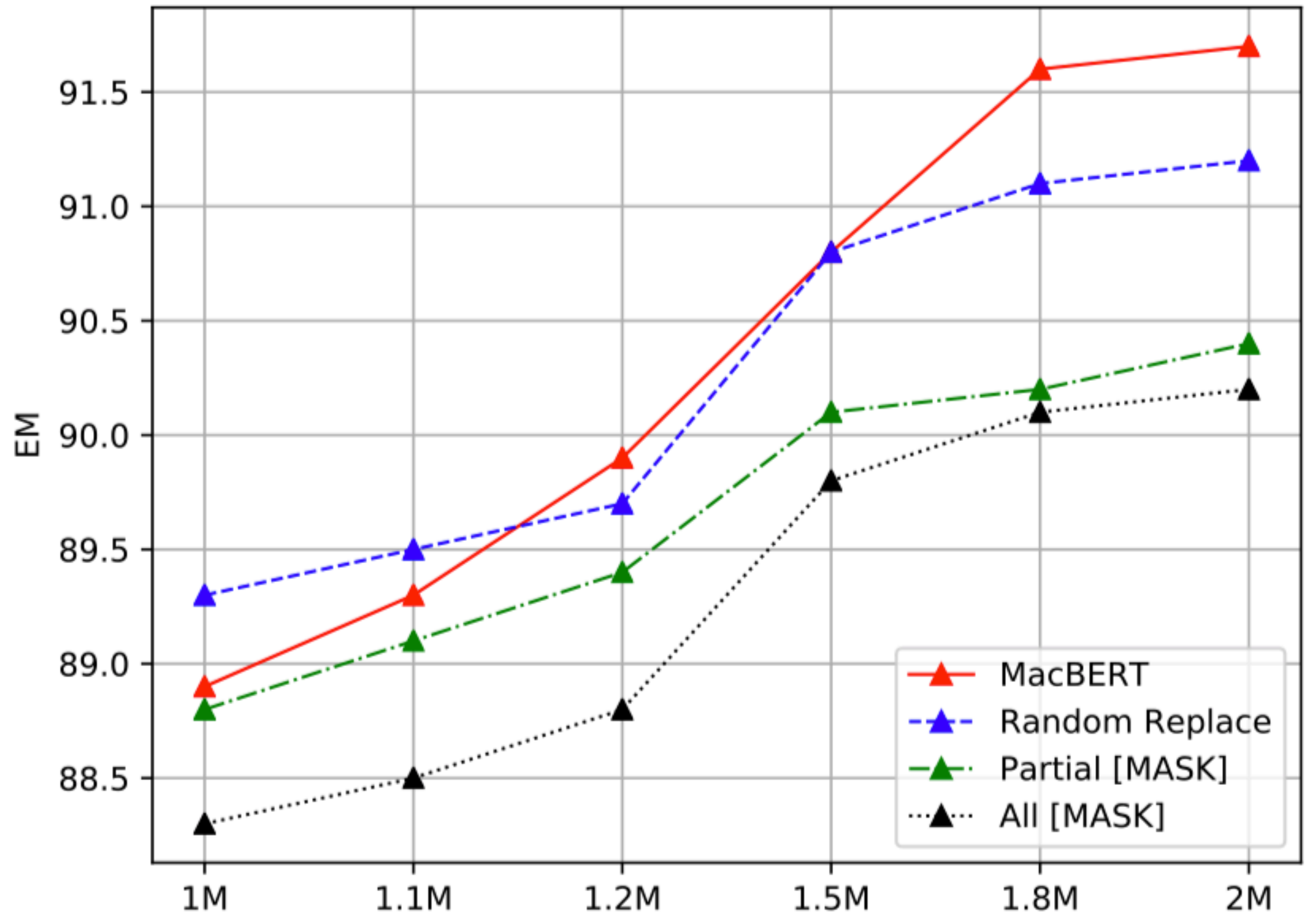
Sentence Pair Classification	XNLI Dev	XNLI Test	LCQMC Dev	LCQMC Test	BQ Corpus Dev	BQ Corpus Test
BERT	77.8 (77.4)	77.8 (77.5)	89.4 (88.4)	86.9 (86.4)	86.0 (85.5)	84.8 (84.6)
BERT-wwm	79.0 (78.4)	78.2 (78.0)	89.4 (89.2)	87.0 (86.8)	86.1 **(85.6)**	85.2 **(84.9)**
BERT-wwm-ext	79.4 (78.6)	78.7 (78.3)	89.6 (89.2)	87.1 (86.6)	**86.4** (85.5)	**85.3** (84.8)
RoBERTa-wwm-ext	80.0 (79.2)	78.8 (78.3)	89.0 (88.7)	86.4 (86.1)	86.0 (85.4)	85.0 (84.6)
ELECTRA-base	77.9 (77.0)	78.4 (77.8)	**90.2 (89.8)**	**87.6 (87.3)**	84.8 (84.7)	84.5 (84.0)
MacBERT-base	**80.4 (79.5)**	**79.3 (78.9)**	89.6 (89.3)	86.5 (86.3)	86.0 (85.4)	85.1 (84.7)
ELECTRA-large	81.5 (80.8)	81.0 **(80.9)**	**90.7 (90.4)**	87.3 **(87.2)**	**86.7 (86.2)**	85.1 (84.8)
RoBERTa-wwm-ext-large	82.1 (81.3)	81.2 (80.6)	90.4 (90.0)	87.0 (86.8)	86.3 (85.7)	**85.8 (84.9)**
MacBERT-large	**82.4 (81.8)**	**81.3** (80.6)	90.6 (90.3)	**87.6** (87.1)	86.2 **(85.7)**	85.6 **(85.0)**

Cui et al., Findings of EMNLP 2020. *Revisiting Pre-trained Models for Chinese Natural Language Processing*

MacBERT

- **Investigation on MLM Tasks**
  - MacBERT: 80% of tokens are replaced into their similar words, and 10% replaced into random words.
  - Random Replace: 90% of tokens are replaced into random words.
  - Partial Mask: original BERT implementation, with 80% tokens replaced into [MASK] tokens, and 10% replaced into random words.
  - All Mask: 90% tokens replaced with [MASK] tokens.





Cui et al., Findings of EMNLP 2020. *Revisiting Pre-trained Models for Chinese Natural Language Processing*

MacBERT

- **Open-Source Chinese PLM Series**
  - BERT: BERT-wwm, BERT-wwm-ext
  - XLNet: XLNet-base, XLNet-mid
  - RoBERTa: RoBERTa-wwm-ext, RoBERTa-wwm-ext-large
  - RBT: RBT3, RBTL3
  - ELECTRA: ELECTRA-small, ELECTRA-small-ex, ELECTRA-base, ELECTRA-large
  - MacBERT: MacBERT-base, MacBERT-large

Our open-source PLM series achieves 5,500+ ⭐ !!!

Cui et al., Findings of EMNLP 2020. *Revisiting Pre-trained Models for Chinese Natural Language Processing*

MacBERT

- **HFL Ranks No.1 in GLUE Benchmark**
  - Further pre-training on ALBERT-xxlarge with Mac objective
  - Dynamic keyword matching (DKM) approach is also applied for better fine-tuning

GLUE

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3	91.1	97.8	92.0	94.5	52.6
2	Alibaba DAMO NLP	StructBERT + TAPT		90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0	90.9	90.7	97.4	91.2	94.5	49.1
3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	91.3	97.5	91.7	94.5	51.2
4	ERNIE Team - Baidu	ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	91.0	96.6	90.9	94.5	51.7
5	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9	92.8	94.5	53.1

CLUE

- **CLUE: C**hinese **L**anguage **U**nderstanding **E**valuation
  - A similar benchmark to GLUE
  - Provide 3 categories, 9 tasks of Chinese NLU
  - Comprehensive comparisons on the existing PLMs

Corpus	|Train|	|Dev|	|Test|	Task	Metric	Source
Single-Sentence Tasks						
TNEWS	53.3k	10k	10k	short text classification	acc.	news title and keywords
IFLYTEK	12.1k	2.6k	2.6k	long text classification	acc.	app descriptions
CLUEWSC2020	1,244	304	290	coreference resolution	acc.	Chinese fiction books
Sentence Pair Tasks						
AFQMC	34.3k	4.3k	3.9k	semantic similarity	acc.	online customer service
CSL	20k	3k	3k	keyword recognition	acc.	academic (CNKI)
OCNLI	50k	3k	3k	natural language inference	acc.	5 genres
Machine Reading Comprehension Tasks						
CMRC 2018	10k	3.4k	4.9k	answer span extraction	EM.	Wikipedia
ChID	577k	23k	23k	multiple-choice, idiom	acc.	novel, essay, and news
C^3	11.9k	3.8k	3.9k	multiple-choice, free-form	acc.	mixed-genre

		Single Sentence			Sentence Pair			MRC		
Model	**Avg**	**TNEWS**	**IFLYTEK**	**CLUEWSC2020**	**AFQMC**	**CSL**	**OCNLI**	**CMRC**	**ChID**	**C^3**
BERT-base	69.26	56.58	60.29	63.45	73.70	80.36	72.70	69.72	82.04	64.50
BERT-wwm-ext-base	70.27	56.84	59.43	62.41	74.07	80.63	74.42	73.23	82.90	68.50
ALBERT-tiny	56.01	53.35	48.71	63.38	69.92	74.56	65.12	53.68	43.53	31.86
ALBERT-xxlarge	72.49	59.46	62.89	61.54	75.60	83.63	77.70	75.15	83.15	73.28
ERNIE-base	69.72	58.33	58.96	63.44	73.83	79.10	74.11	73.32	82.28	64.10
XLNet-mid	68.58	56.24	57.85	61.04	70.50	81.26	72.63	66.51	83.47	67.68
RoBERTa-large	71.01	57.86	62.55	62.44	74.02	81.36	76.82	76.11	84.50	63.44
RoBERTa-wwm-ext-base	71.17	56.94	60.31	72.07	74.04	81.00	74.72	73.89	83.62	63.90
RoBERTa-wwm-ext-large	74.80	58.61	62.98	81.38	76.55	82.13	77.30	76.58	85.37	72.32
Human	**85.09**	**71.00**	**66.00**	**98.00**	**81.0**	**84.0**	**90.30**	**92.40**	**87.10**	**96.00**

Xu et al., COLING 2020. *CLUE: A Chinese Language Understanding Evaluation Benchmark*

预训练语言模型近期研究进展

RECENT ADVANCES IN PRE-TRAINED LANGUAGE MODELS

Recent PLMs

- **Trending**
  - GPT-2, GPT-3, T5
- **Distillation**
  - DistilBERT, TinyBERT, MobileBERT
  - TextBrewer
- **Multi-lingual**
  - mBERT, XLM, XLM-R

GPT-2

- **GPT-2: Language Models are Unsupervised Multitask Learners**
  - Language model can perform down-stream tasks in a zero-shot setting
  - Capacity of the language model is essential to the success of zero-shot task transfer

Language Models are Unsupervised Multitask Learners

Alec Radford [* 1] **Jeffrey Wu** [* 1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [** 1] **Ilya Sutskever** [** 1]

Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

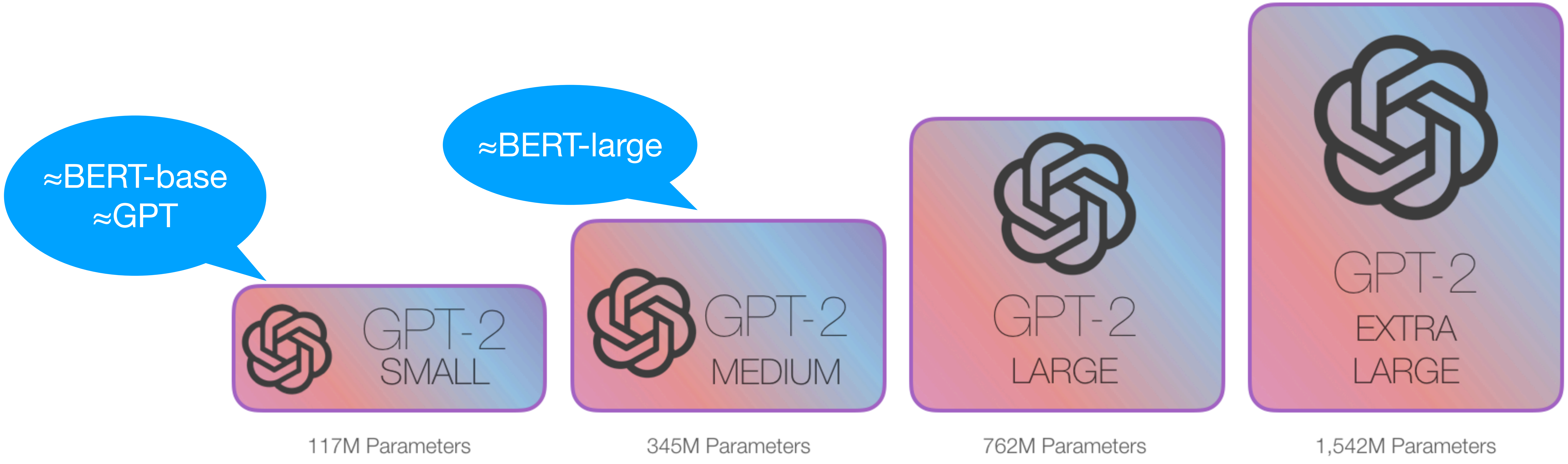
GPT-2

- **Training Phase**
  - Almost IDENTICAL model structure GPT,
  - Pre-training data: 6GB → 40GB uncompressed free text
  - A few modifications
    - Layer normalization is moved to the input of each sub-block
    - An additional layer normalization is added after the final self-attention block
    - Vocabulary is expanded to 50,257 (GPT: 40,000)
    - Context size is increased to 1024 (GPT: 512)
- **Inference Phase**
  - Adapt to each task using a free form input

Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

GPT-2

- **Model Sizes**



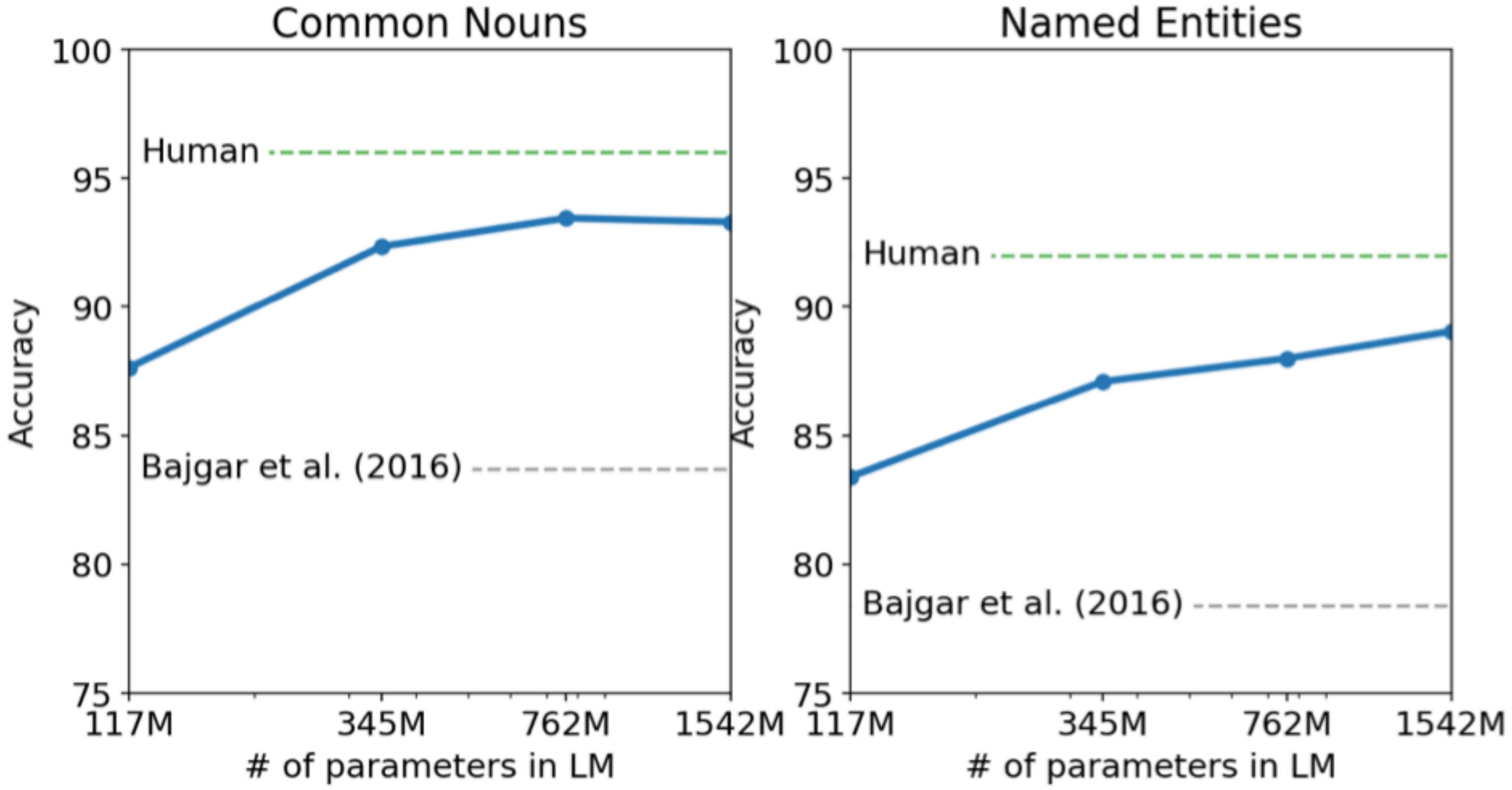
Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

GPT-2

- **Experimental Results**

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Cloze-style Reading Comprehension



Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

GPT-3

- **GPT-3: Language Models are Few-Shot Learners**
  - Almost nothing new on top of GPT-2 architecture
  - Substantially BIGGER model than all previous PLMs - we need more power!

Language Models are Few-Shot Learners

Tom B. Brown* **Benjamin Mann*** **Nick Ryder*** **Melanie Subbiah***

Jared Kaplan† **Prafulla Dhariwal** **Arvind Neelakantan** **Pranav Shyam** **Girish Sastry**

Amanda Askell **Sandhini Agarwal** **Ariel Herbert-Voss** **Gretchen Krueger** **Tom Henighan**

Rewon Child **Aditya Ramesh** **Daniel M. Ziegler** **Jeffrey Wu** **Clemens Winter**

Christopher Hesse **Mark Chen** **Eric Sigler** **Mateusz Litwin** **Scott Gray**

Benjamin Chess **Jack Clark** **Christopher Berner**

Sam McCandlish **Alec Radford** **Ilya Sutskever** **Dario Amodei**

Brown et al., 2020. *Language Models are Few-Shot Learners*

GPT-3

- **Model**
  - Similar to GPT-2 with alternating dense and locally banded sparse attention patterns in the transformer layer

GPT-2 has **1.5B** params, takes about **6G** disk space

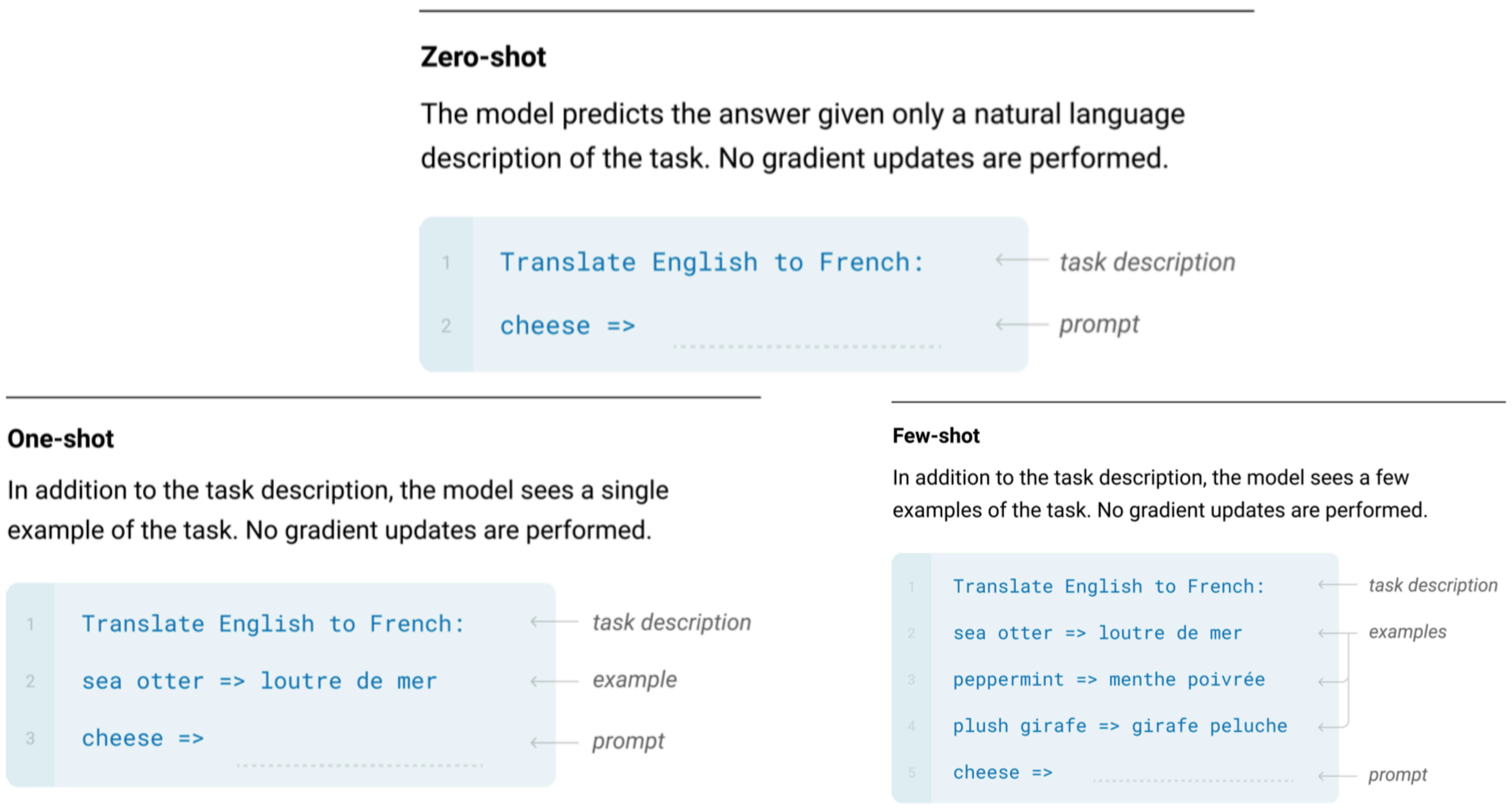
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

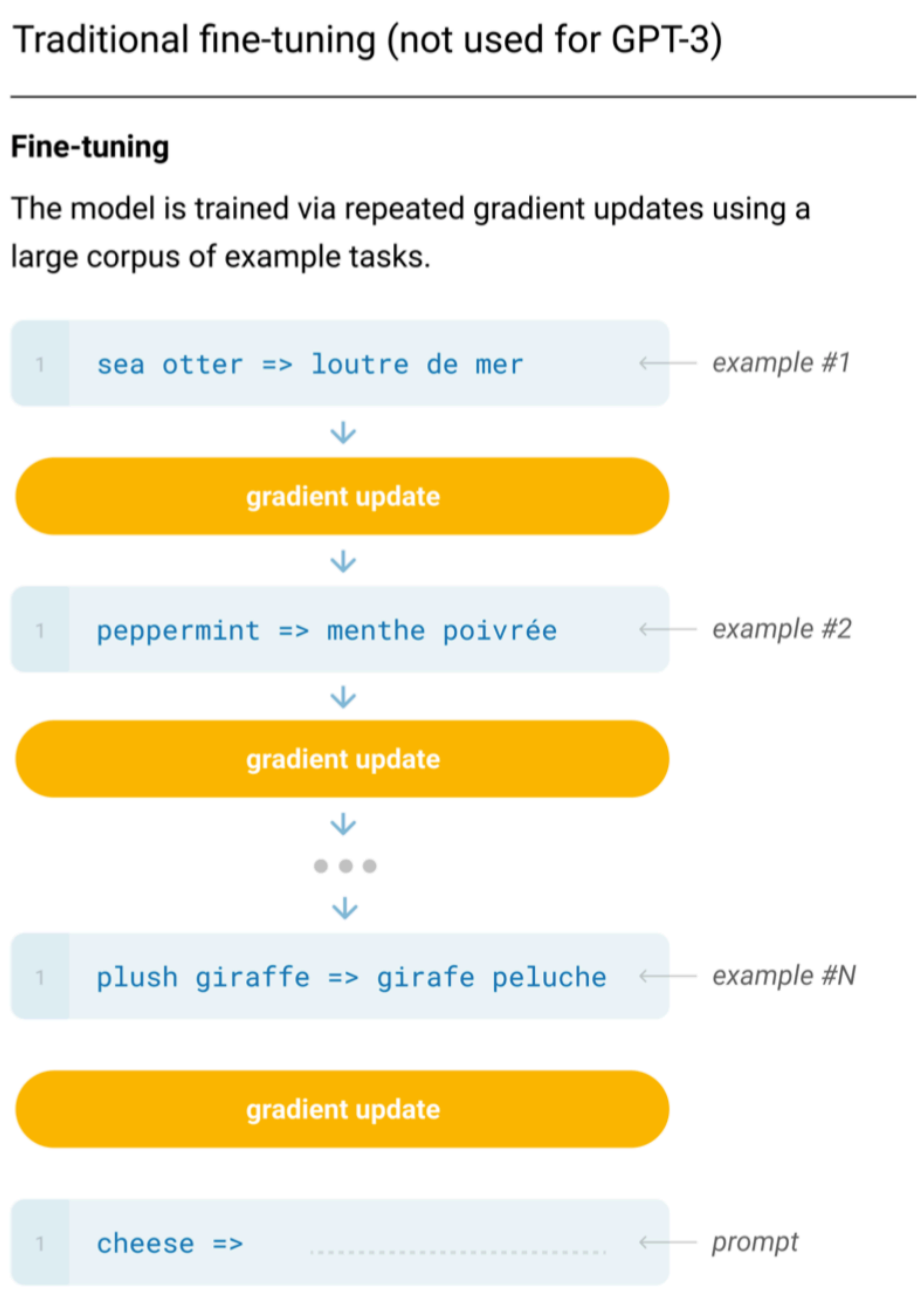
>700G disk space

Brown et al., 2020. *Language Models are Few-Shot Learners*

GPT-3

- **Settings**
  - Traditional scheme: fine-tuning
  - In GPT-3: zero-shot, one-shot, few-shot





Brown et al., 2020. *Language Models are Few-Shot Learners*

GPT-3

- **Results**
  - Remarkable performance on zero-shot, one-shot, few-shot settings

LM

Setting	PTB
SOTA (Zero-Shot)	35.8[a]
GPT-3 Zero-Shot	**20.5**

CLOZE

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

QA

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

MT

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	<u>37.5</u>	34.9	28.3	35.2	<u>35.2</u>	33.1
mBART [LGG+20]	-	-	<u>29.8</u>	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	<u>39.2</u>	29.7	<u>40.6</u>	21.0	<u>39.5</u>

CQA

Setting	PIQA	ARC (Easy)	ARC (Challenge)	OpenBookQA
Fine-tuned SOTA	79.4	**92.0**[KKS+20]	**78.5**[KKS+20]	**87.2**[KKS+20]
GPT-3 Zero-Shot	**80.5***	68.8	51.4	57.6
GPT-3 One-Shot	**80.5***	71.2	53.2	58.8
GPT-3 Few-Shot	**82.8***	70.1	51.5	65.4

MRC

Setting	CoQA	DROP	QuAC	SQuADv2	RACE-h	RACE-m
Fine-tuned SOTA	**90.7**[a]	**89.1**[b]	**74.4**[c]	**93.0**[d]	**90.0**[e]	**93.1**[e]
GPT-3 Zero-Shot	81.5	23.6	41.5	59.5	45.5	58.4
GPT-3 One-Shot	84.0	34.3	43.3	65.4	45.9	57.4
GPT-3 Few-Shot	85.0	36.5	44.3	69.8	46.8	58.1

T5

- **T5: T**ext-**to**-**T**ext **T**ransfer **T**ransformer
  - Propose an encoder-decoder scheme for all NLP tasks
  - Comprehensive model design comparisons
  - Pre-training data size: C4 (~750G)

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

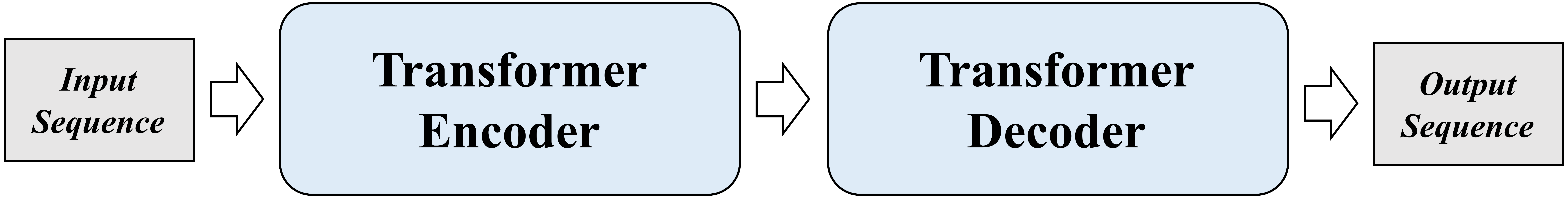
Colin Raffel* Noam Shazeer* Adam Roberts* Katherine Lee*
Sharan Narang Michael Matena Yanqi Zhou Wei Li Peter J. Liu

Google

Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

T5

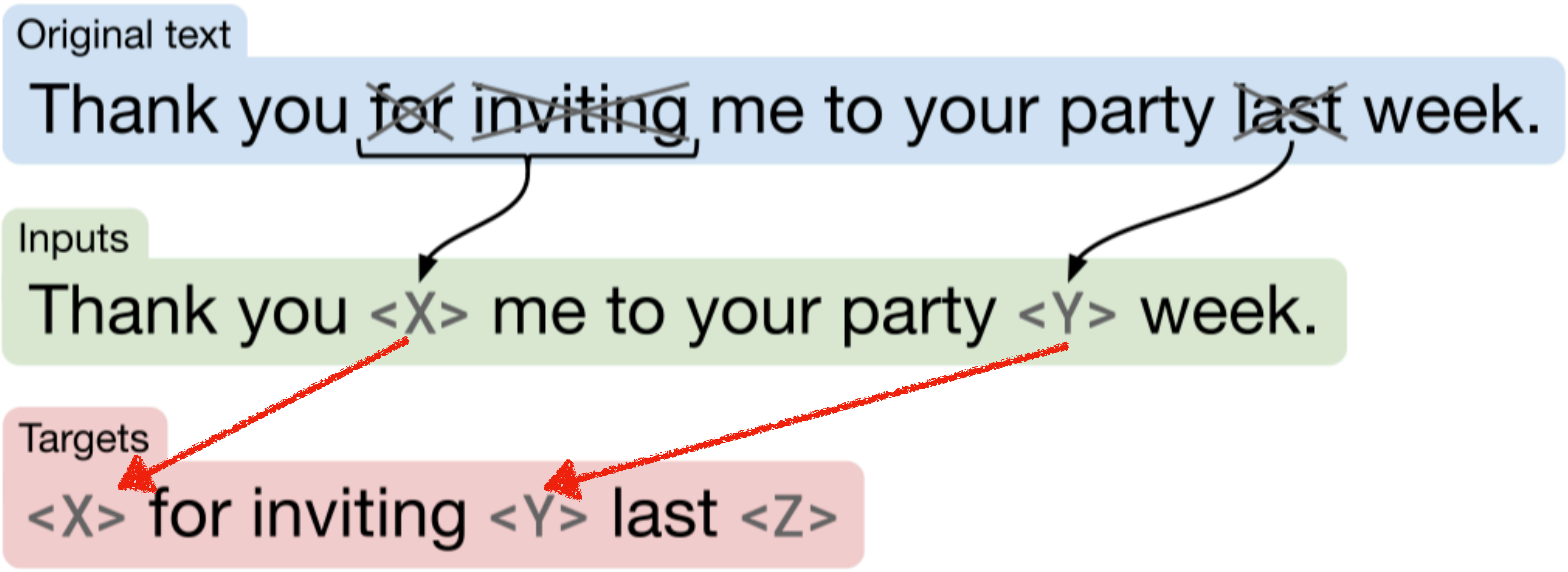
- **Overall Architecture**
  - Transformer-based Encoder-Decoder architecture
  - Taking every NLP task as a “text-to-text” problem



Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

T5

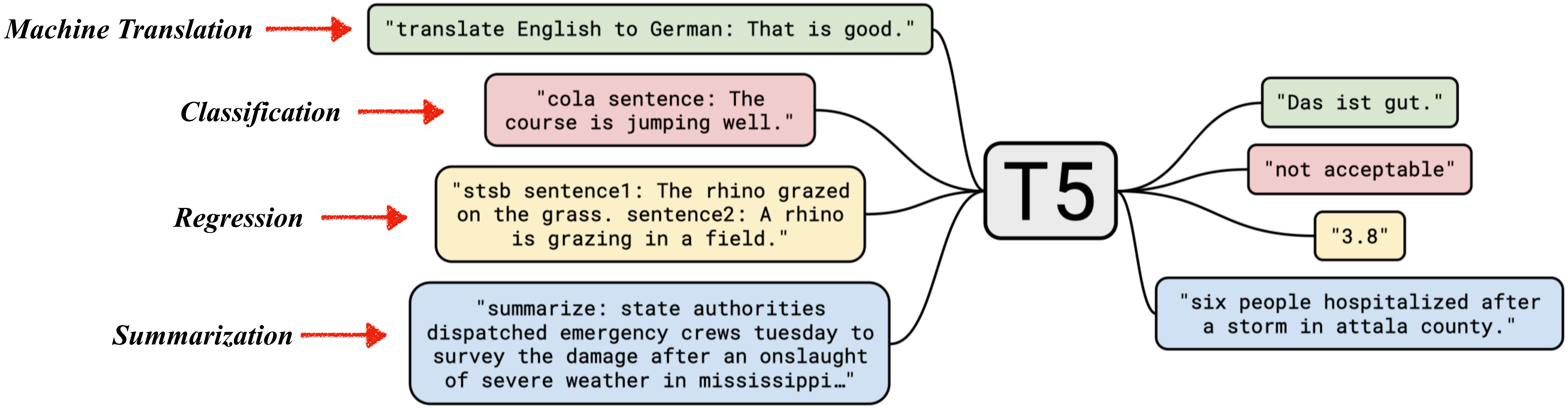
- **Training Stage**
  - A span-corruption unsupervised training objective
  - Randomly mask 15% tokens in the input sequence



Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

T5

- **Fine-tuning Phase**
  - Universal input-output scheme for all downstream tasks



Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

T5

- **Experimental Results**
  - Small: 60M, base: 220M, large: 770M

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	**97.1**[a]	**93.6**[b]	**91.5**[b]	**92.7**[b]	**92.3**[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**89.7**	**70.8**	**97.1**	91.9	89.2	92.5	92.1

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	**74.8**[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	74.6	90.4	**92.0**	**91.7**	96.7	**92.5**	**93.2**

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	88.95[d]	94.52[d]	84.6[e]	87.1[e]	90.5[e]	95.2[e]	90.6[e]
T5-Small	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5-Base	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5-Large	86.66	93.79	82.3	85.4	91.6	94.8	83.4
T5-3B	88.53	94.95	86.4	89.9	90.3	94.4	92.0
T5-11B	**90.06**	**95.64**	**88.9**	**91.0**	**93.0**	**96.4**	**94.8**

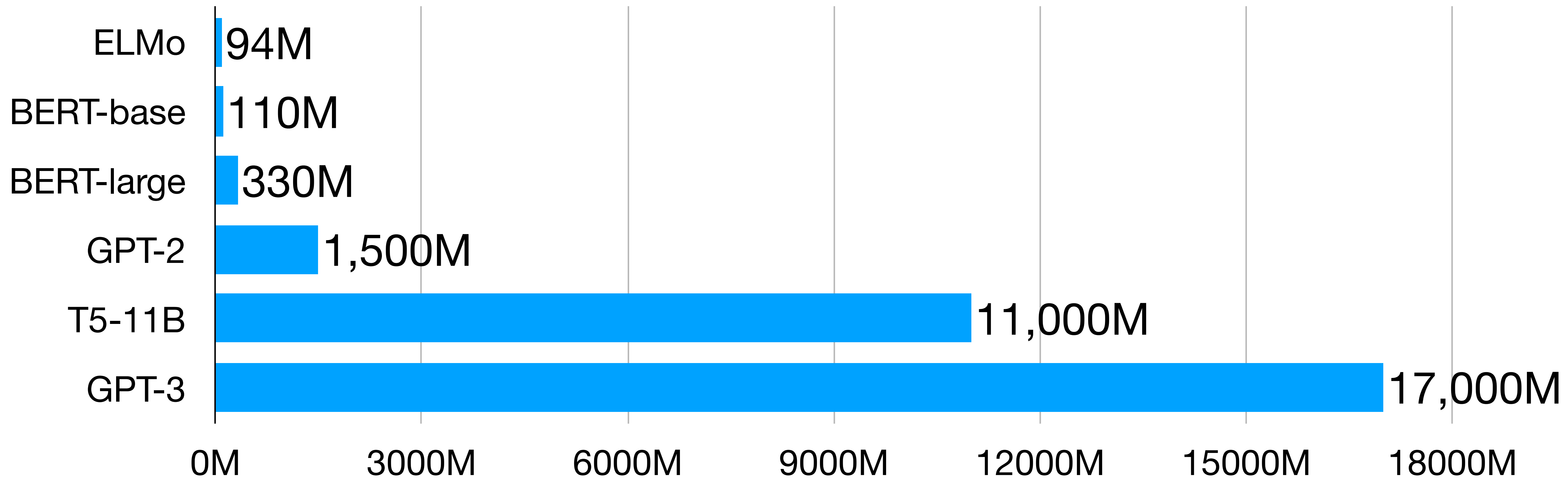
Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[e]	52.5[e]	90.6[e]	90.0[e]	88.2[e]	69.9[e]	89.0[e]
T5-Small	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5-Base	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5-Large	83.3	50.7	86.8	85.9	87.8	69.3	86.3
T5-3B	86.8	58.3	91.2	90.4	90.7	72.1	90.4
T5-11B	**88.2**	**62.3**	**93.3**	**92.5**	**92.5**	**76.1**	**93.8**

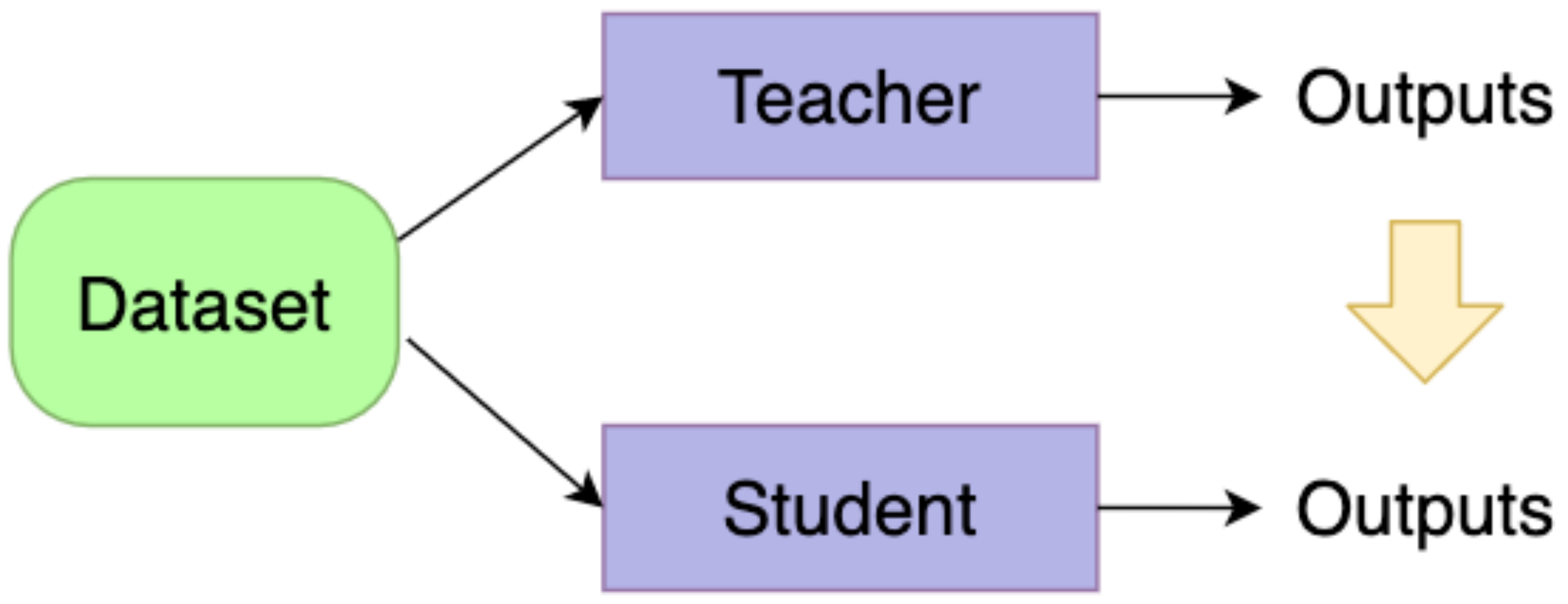
Model	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM ROUGE-2	CNN/DM ROUGE-L
Previous best	**33.8**[f]	**43.8**[f]	**38.5**[g]	43.47[h]	20.30[h]	40.63[h]
T5-Small	26.7	36.0	26.8	41.12	19.56	38.35
T5-Base	30.9	41.2	28.0	42.05	20.34	39.40
T5-Large	32.0	41.5	28.1	42.50	20.68	39.75
T5-3B	31.8	42.6	28.2	42.72	21.02	39.94
T5-11B	32.1	43.4	28.1	**43.52**	**21.55**	**40.69**

Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

Distillation

- **Towards More Compact and Efficient PLMs**
  - PLMs are way bigger than traditional neural network models
  - Real-time application requires much quicker inference time and compact size
  - **Knowledge distillation** is a technique of transferring knowledge from a large (teacher) model to a small (student) model, without significant loss in performance.





Knowledge Distillation

Distillation

- **DistilBERT**
  - A general-purpose / task-agnostic pre-trained distilled version of BERT
  - 40% smaller, 60% faster, retains 97% of the language understanding capabilities

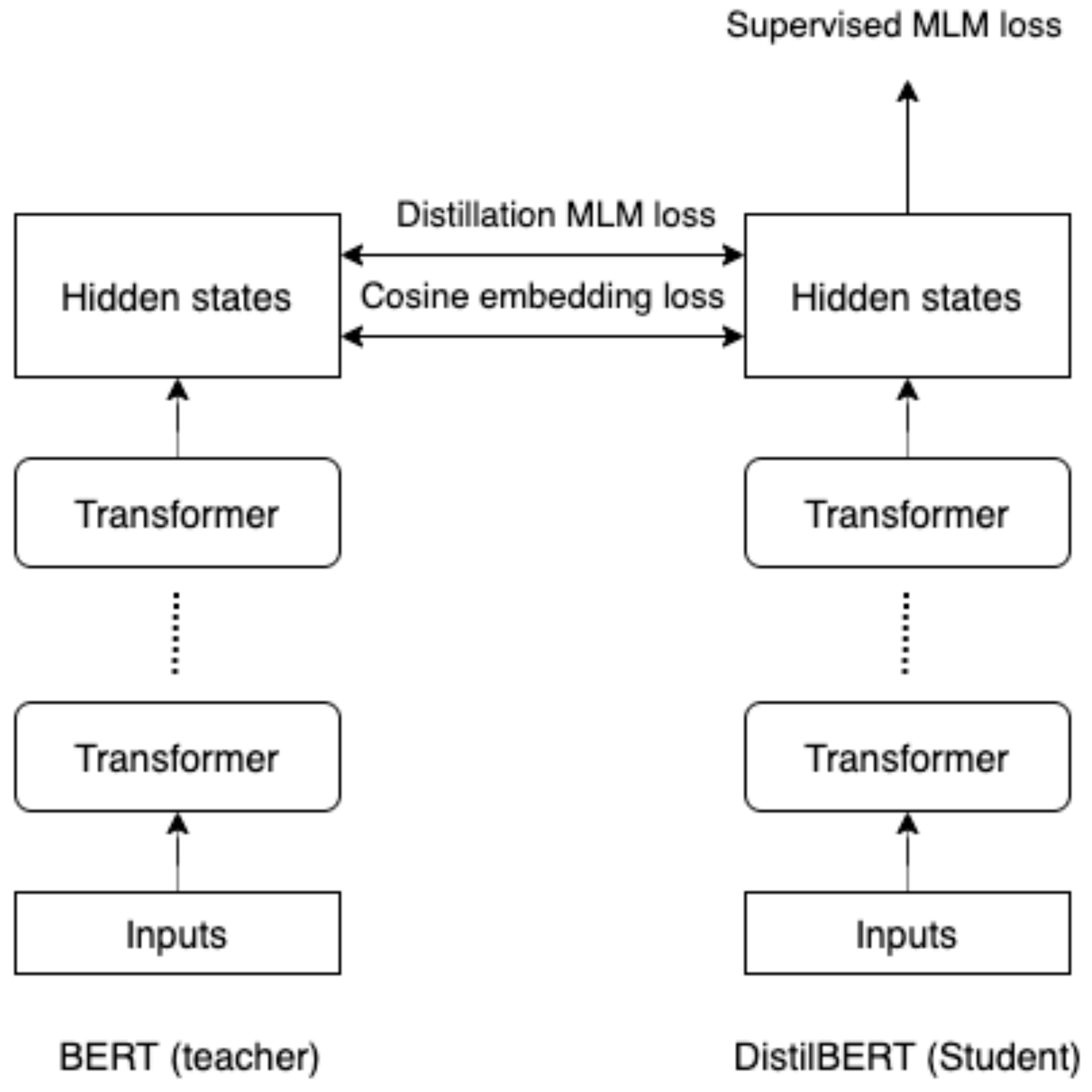
DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF
Hugging Face
{victor,lysandre,julien,thomas}@huggingface.co

Sanh et al., arXiv 2019. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*

Distillation

- **DistilBERT**
  - The student (DistilBERT, 6-layer) has the same general architecture
  - The training objective is the linear combination of the following losses
    - a supervised MLM loss with hard-labels given by the dataset
    - a distillation MLM loss with soft-labels given by the teacher
    - a cosine embedding loss which aligns the directions of the student and teacher hidden states vectors.



Sanh et al., arXiv 2019. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*

Distillation

- **TinyBERT**
  - A new distillation method (Transformer Distillation) that matches different representations from BERT layers
  - A two-stage learning framework with performing the proposed Transformer distillation at both the pre-training and fine-tuning stages
  - TinyBERT achieves over 96% the performance of teacher (BERT-base) on GLUE while having much fewer parameters (~13.3%)

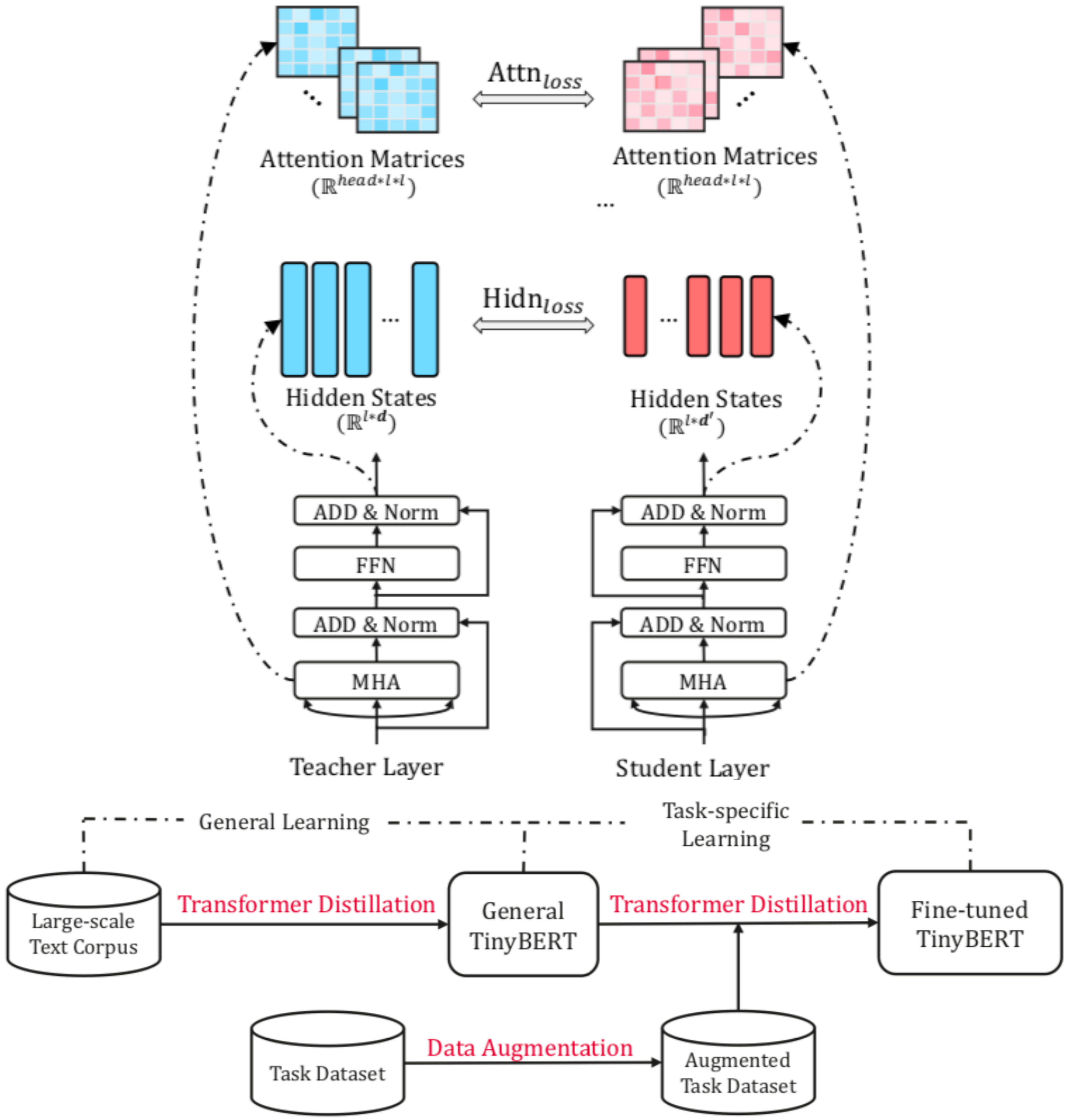
TINYBERT: DISTILLING BERT FOR NATURAL LANGUAGE UNDERSTANDING

Xiaoqi Jiao[1*†], **Yichun Yin**[2*], **Lifeng Shang**[2], **Xin Jiang**[2]
Xiao Chen[2], **Linlin Li**[3], **Fang Wang**[1] **and Qun Liu**[2]
[1]Huazhong University of Science and Technology
[2]Huawei Noah's Ark Lab
[3]Huawei Technologies Co., Ltd.

Jiao et al., arXiv 2019. *TinyBERT: Distilling BERT for Natural Language Understanding*

Distillation

- **TinyBERT**
  - Transformer Distillation performs the distillation on different representations
    - Normal prediction-layer distillation
    - **Attention distillation** and hidden states distillation
  - Two-stage learning
    - General (MLM) distillation: use the original BERT without fine-tuning as the teacher, and a large-scale text corpus as the training data
    - Task-specific Distillation: use the fine-tuned BERT as the teacher, re-perform the proposed distillation method on an augmented task-specific dataset



Jiao et al., arXiv 2019. *TinyBERT: Distilling BERT for Natural Language Understanding*

Distillation

- **MobileBERT**
  - MobileBERT is as deep as BERT-large while each layer is thinner, with re-designed building blocks.
  - A variety of knowledge transfer strategies have been carefully investigated.
  - MobileBERT is 4.3x smaller and 5.5x faster than BERT-base, while it can still achieve competitive results on well-known NLP benchmarks.

MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices

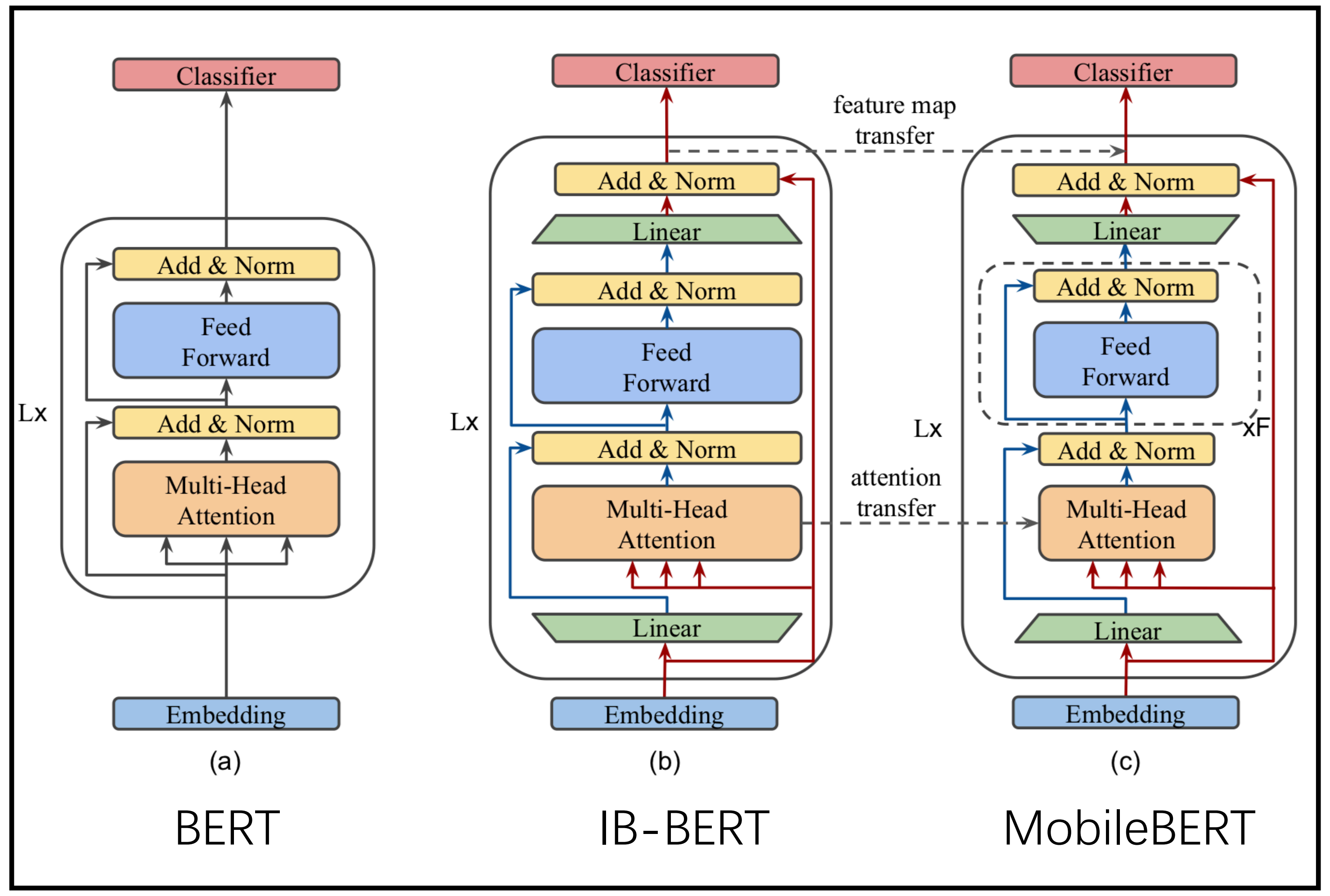
Zhiqing Sun[1*], Hongkun Yu[2], Xiaodan Song[2], Renjie Liu[2], Yiming Yang[1], Denny Zhou[2]

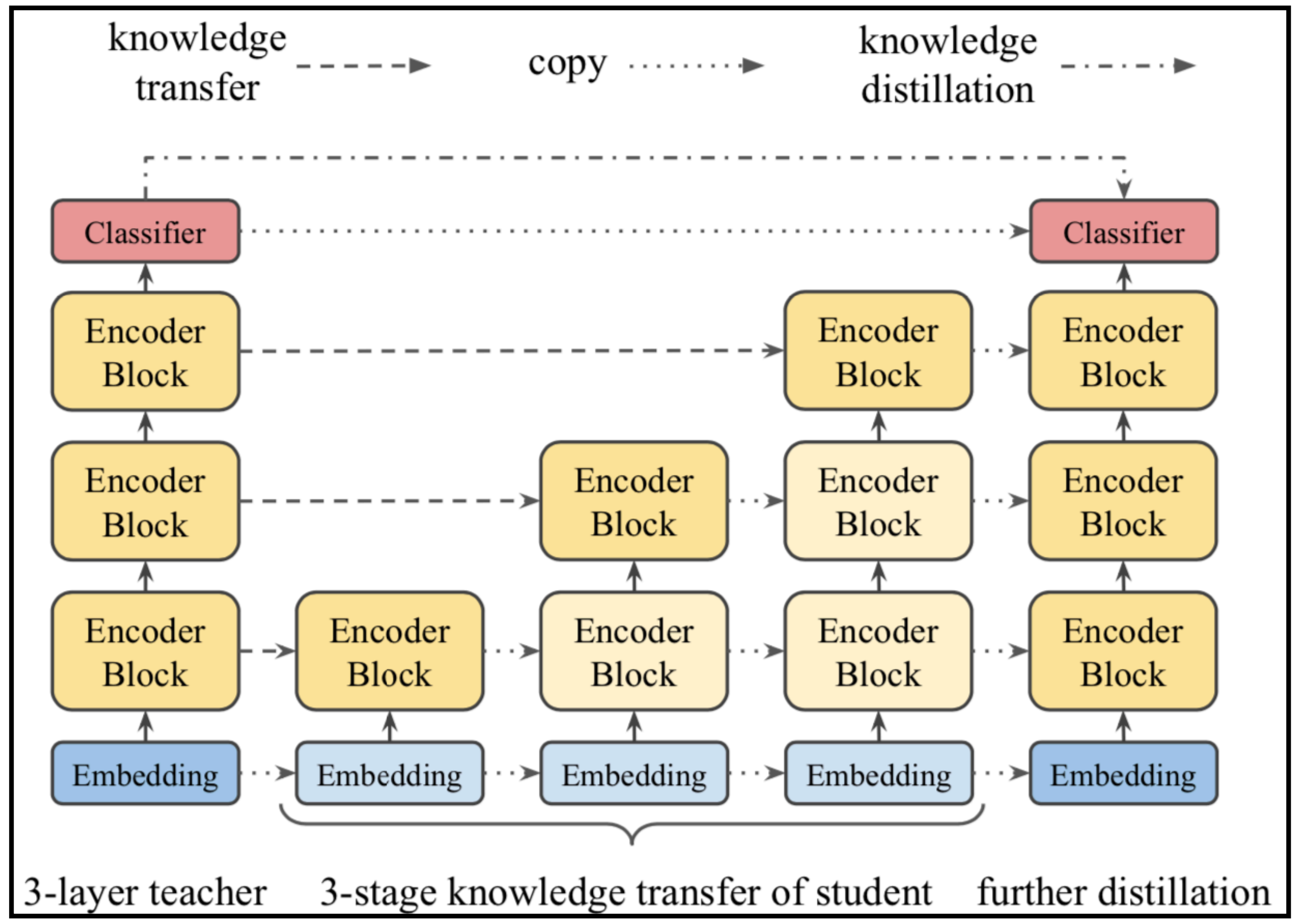
[1]Carnegie Mellon University {zhiqings, yiming}@cs.cmu.edu
[2]Google Brain {hongkuny, xiaodansong, renjieliu, dennyzhou}@google.com

Sun et al., ACL 2020. *MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices*

Distillation

- **MobileBERT**
  - Trains the teacher (IB-BERT, 24 layers) from scratch; Distills MobileBERT from the IB-BERT
  - Objectives: Normal Distillation MLM loss + the following knowledge transfer objectives
    - Feature map transfer: matches feature maps between IB-BERT and MobileBERT
    - Attention Transfer: matches self-attention maps between IB-BERT and MobileBERT
  - Training Strategies: Progressive Knowledge Transfer
    - Progressively train each layer with L stages: when training the l-th layer, freeze the layers below.





Sun et al., ACL 2020. *MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices*

- **TextBrewer: An Open-Source Knowledge Distillation Toolkit**
  - A PyTorch-based distillation toolkit for NLP
  - Aims to save the effort of setting up distillation experiments

- **Wide-model-support**: especially transformer-based models
- **Flexible**: includes various distillation methods and strategies which can be freely combined
- **Easy to use**: no need to modify your model code, most parts of your existing training scripts could be re-used
- **Built for NLP**: works on typical tasks like text classification, reading comprehension, and sequence labeling

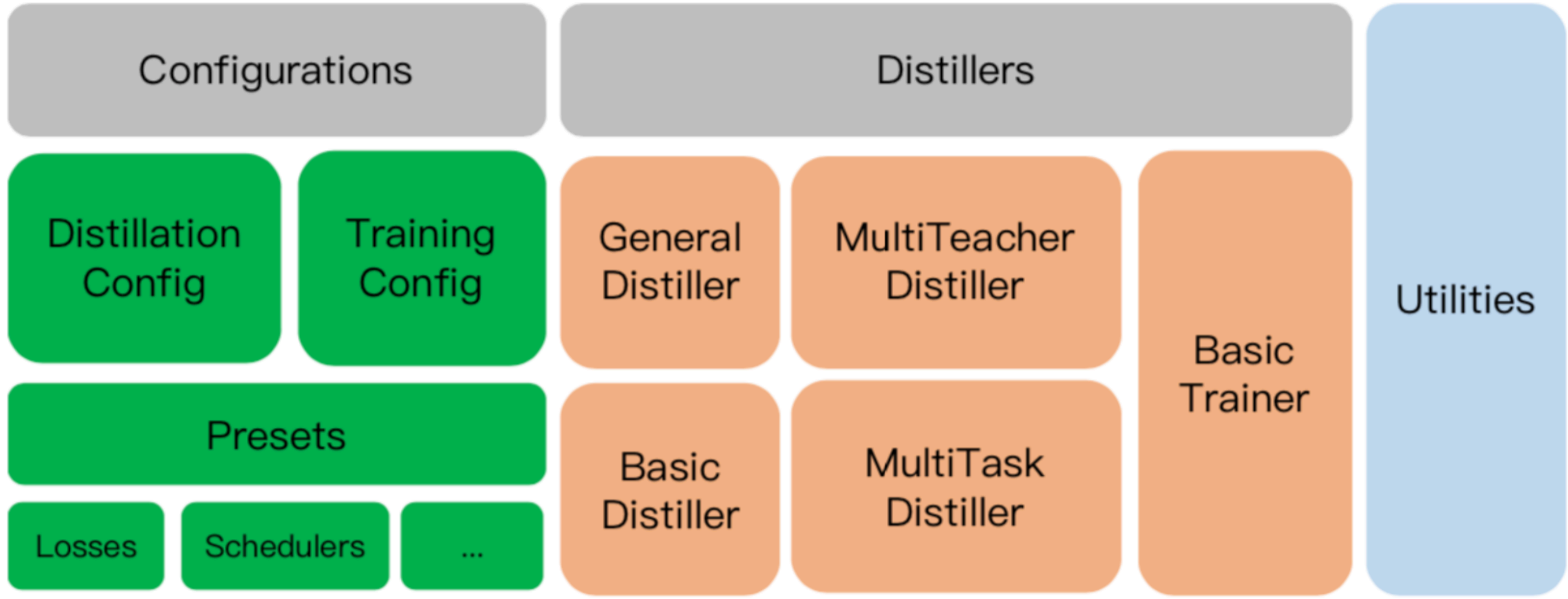
TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing

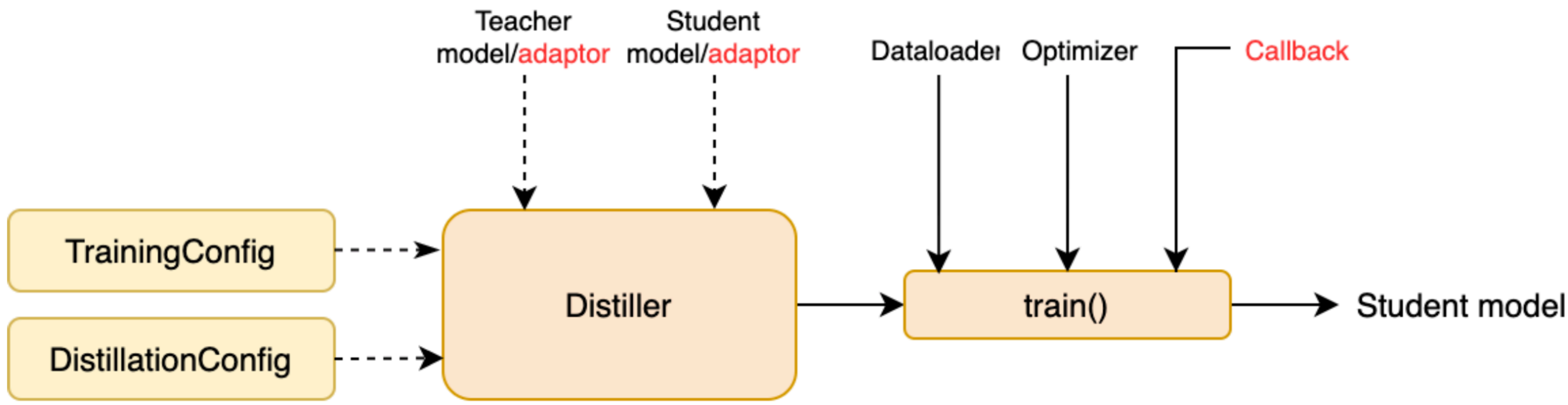
Ziqing Yang[†], Yiming Cui[‡†], Zhipeng Chen[†],
Wanxiang Che[‡], Ting Liu[‡], Shijin Wang[†§], Guoping Hu[†]
[†]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[‡]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[§]iFLYTEK AI Research (Hebei), Langfang, China

Yang et al., ACL 2020. *TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing*

TextBrewer

- **Overall Architecture**
  - Configurations: define the distillation method and training process
  - Distillers: conduct the actual distillation process (5 kinds of distillers)
  - Utilities: useful tools such as model size analysis and simple data augmentation strategies
- **Workflow**
  - preparatory work
    - Train the teacher model
    - Initialize the student model, dataloader, and optimizer
  - Distillation with TextBrewer
    - Initialize two configurations and a distiller
    - Define auxiliary functions (adaptors and a callback)
    - Call the train method of the distiller to start distillation





Yang et al., ACL 2020. *TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing*

TextBrewer

- **Setups**
  - Teachers: BERT-base
  - Students: T6 (60%), T3 (41%), T3-small (16%), T4-tiny (same as TinyBERT, 13%)
- **Results: Single-teacher distillation**
  - T6 achieves over 99% of the teachers on all tasks
  - T4-tiny outperforms TinyBERT when training with the same amount of data
- **Results: Multi-teacher distillation**
  - All the models (teachers and the student) are BERT-base structure
  - Student model achieves the best performance, higher than the ensemble result

Single Teacher

Model	MNLI		SQuAD		CoNLL-2003
	m	mm	EM	F1	F1
$BERT_{BASE}$	83.7	84.0	81.5	88.6	91.1
Public					
DistilBERT	81.6	81.1	79.1	86.9	-
TinyBERT	80.5	81.0	-	-	-
+DA	82.8	82.9	72.7	82.1	-
TextBrewer					
BiGRU	-	-	-	-	85.3
T6	83.6	84.0	80.8	88.1	90.7
T3	81.6	82.5	76.3	84.8	87.5
T3-small	81.3	81.7	72.3	81.4	78.6
T4-tiny	82.0	82.6	73.7	82.5	77.5

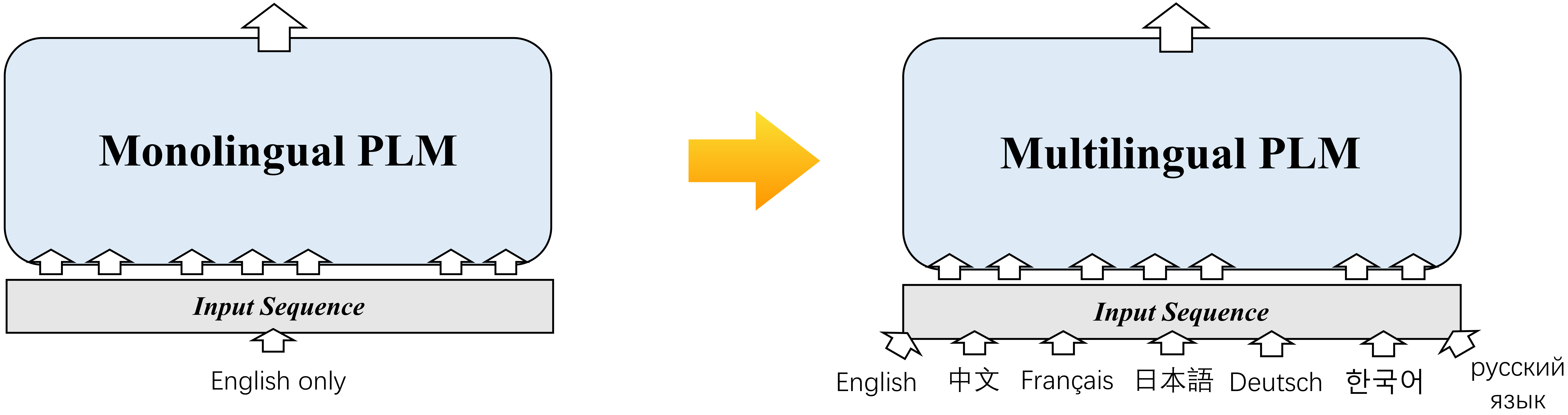
Multi Teacher

Model	MNLI		SQuAD		CoNLL-2003
	m	mm	EM	F1	F1
Teacher 1	83.6	84.0	81.1	88.6	91.2
Teacher 2	83.6	84.2	81.2	88.5	90.8
Teacher 3	83.7	83.8	81.2	88.7	91.3
Ensemble	84.3	84.7	82.3	89.4	91.5
Student	**84.8**	**85.3**	**83.5**	**90.0**	**91.6**

Yang et al., ACL 2020. *TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing*

Multi-lingual

- **All-Language-in-One PLM**
    - Most of the research focuses on English only, leading to a severe unbalance in the language diversity in natural language processing
    - One-language-at-a-time training is computationally expensive, especially for PLM



Multi-lingual

- **Multi-lingual BERT (mBERT)**
  - mBERT is the first multilingual pre-trained model, a single model trained on 104 languages.
  - mBERT has learned high-quality cross-lingual representation and shown surprisingly good zero-shot cross-lingual performance on several NLP tasks.

System	English	Chinese	Spanish	German	Arabic	Urdu
XNLI Baseline - Translate Train	73.7	67.0	68.8	66.5	65.8	56.6
XNLI Baseline - Translate Test	73.7	68.3	70.7	68.7	66.8	59.3
BERT - Translate Train Cased	**81.9**	**76.6**	**77.8**	**75.9**	**70.7**	61.6
BERT - Translate Train Uncased	81.4	74.2	77.3	75.2	70.5	61.7
BERT - Translate Test Uncased	81.4	70.1	74.9	74.4	70.4	**62.1**
BERT - Zero Shot Uncased	81.4	63.8	74.3	70.5	62.1	58.3

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Multi-lingual

- **XLM: Cross-lingual Language Model Pretraining**
  - A new unsupervised method for learning cross-lingual representations (CLM).
  - A new supervised learning objective that improves cross-lingual pretraining when parallel data is available (TLM).
  - The model (XLM) significantly improves the performance on cross-lingual classification, unsupervised machine translation and supervised machine translation.

Cross-lingual Language Model Pretraining

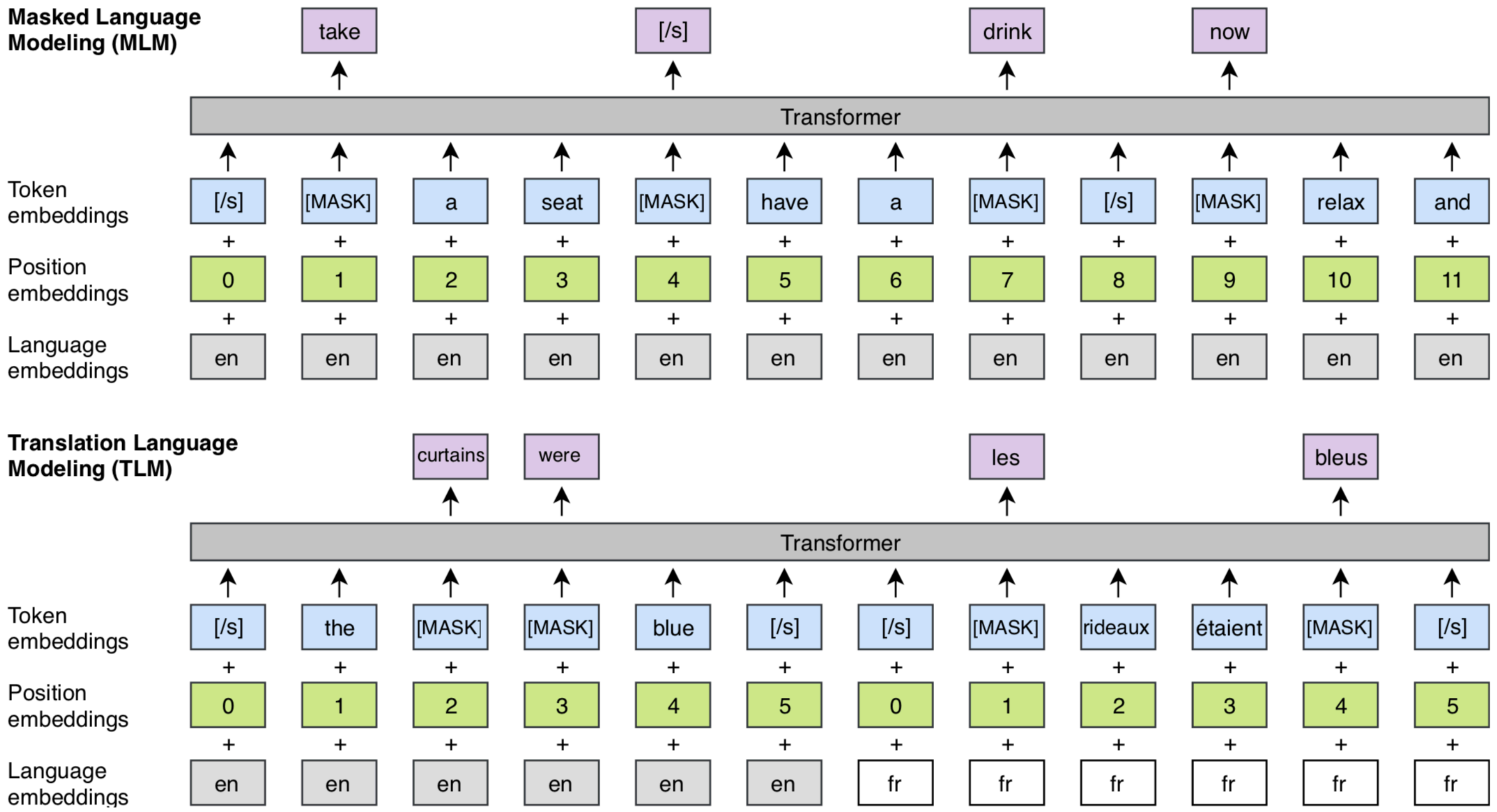
Alexis Conneau*
Facebook AI Research
Université Le Mans
aconneau@fb.com

Guillaume Lample*
Facebook AI Research
Sorbonne Universités
glample@fb.com

Conneau and Lample, NeurIPS 2019. *Cross-lingual Language Model Pretraining*

Multi-lingual

- **XLM**
  - Input representation: adds language embeddings to the input
  - Three language modeling objectives: MLM, CLM, TLM
  - Masked Language Modeling (MLM) Monolingual Data
    - Same as BERT's MLM
  - Casual Language Modeling (CLM) Monolingual Data
    - Model the probability of a word given the history in a sentence
  - Translation Language Modeling (TLM) Parallel Data
    - Similar to MLM, but concatenate parallel sentences as the input
    - Randomly mask words in both source and target sentences
    - Encourages model to leverage context from the other language



Conneau and Lample, NeurIPS 2019. *Cross-lingual Language Model Pretraining*

Multi-lingual

- **XLM-R**
    - Improves cross-lingual language understanding by carefully studying the effects of training unsupervised cross-lingual representations on a very large scale.
    - XLM-R pre-trained on a text in 100 languages obtains state-of-the-art performances on cross-lingual classification, sequence labeling, and question answering.

Unsupervised Cross-lingual Representation Learning at Scale

Alexis Conneau* Kartikay Khandelwal*

Naman Goyal Vishrav Chaudhary Guillaume Wenzek Francisco Guzmán

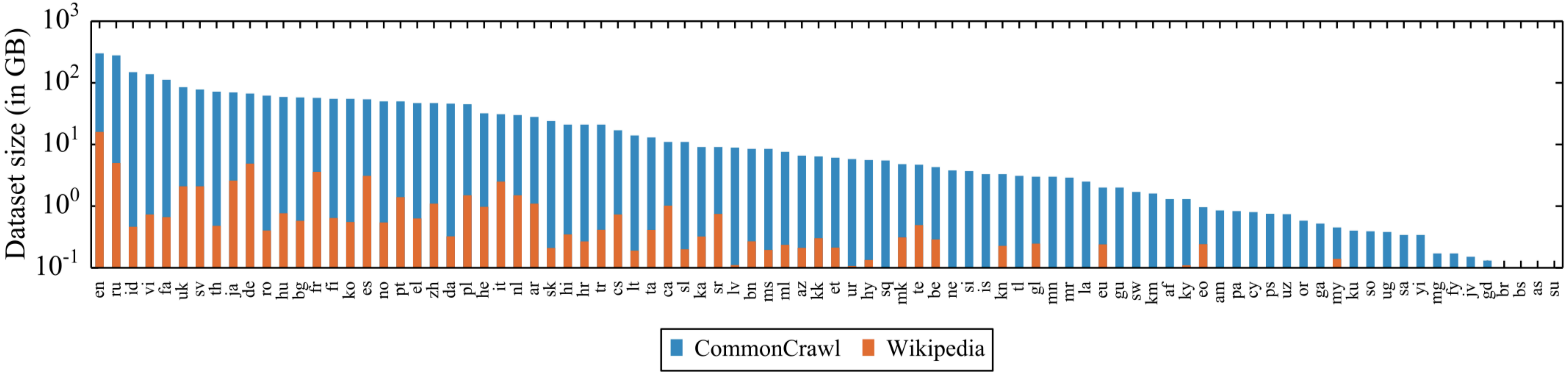
Edouard Grave Myle Ott Luke Zettlemoyer Veselin Stoyanov

Facebook AI

Conneau et al., ACL 2020. *Unsupervised Cross-lingual Representation Learning at Scale*

Multi-lingual

- **Training with a simple objective**
  - model structure is the same as RoBERTa, trained with the MLM objective.
  - unlike XLM, there is **no CLM, TLM**, and language embeddings.
- **Training with more data**
  - Includes 100 languages, with a vocabulary size of 250K.
  - Data used in XLM-R is several orders of magnitude larger than mBERT, in particular for low-resource languages.
  - Unlike XLM model, only monolingual data is used. Parallel data is no longer required.



Conneau et al., ACL 2020. *Unsupervised Cross-lingual Representation Learning at Scale*

Multi-lingual

- **Experimental Results on LM-R**
  - Improves cross-lingual language understanding by carefully studying the effects of training unsupervised cross-lingual representations on a very large scale.
  - XLM-R pre-trained on a text in 100 languages obtains state-of-the-art performances on cross-lingual classification, sequence labeling, and question answering.

Unsupervised Cross-lingual Representation Learning at Scale

Alexis Conneau* Kartikay Khandelwal*

Naman Goyal Vishrav Chaudhary Guillaume Wenzek Francisco Guzmán

Edouard Grave Myle Ott Luke Zettlemoyer Veselin Stoyanov

Facebook AI

Conneau et al., ACL 2020. *Unsupervised Cross-lingual Representation Learning at Scale*

总结

SUMMARY

Summary

- From static to dynamic
  - word2vec, GloVe → CoVe, ELMo
- From dynamic to deep dynamic
  - GPT, BERT, XLNet, RoBERTa, ALBERT, ELECTRA
- Efforts in Chinese PLMs
  - Chinese BERT-wwm, ERNIE, NEZHA, ZEN, MacBERT
- Trending PLMs
  - GPT-2, GPT-3, T5
  - Distillation / Multi-lingual

Thank You!

HFL
Official WeChat

Personal Website

E-mail: me@ymcui.com