Pre-trained Language Models

Zhiyuan Liu

liuzy@tsinghua.edu.cn

THUNLP

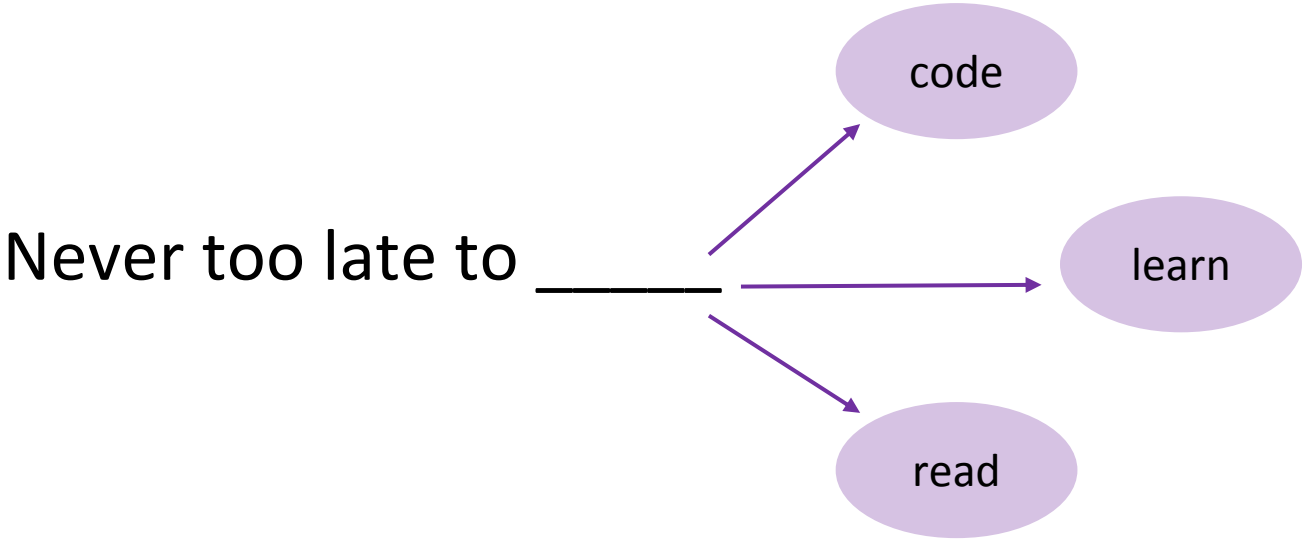
Language Modeling

THUNLP

Review of Language Modeling

- Language Modeling is the task of **predicting the upcoming word**
  - Compute conditional probability of an upcoming word w_n:

$$P(w_n|w_1, w_2, \cdots, w_{n-1})$$



Language Modeling

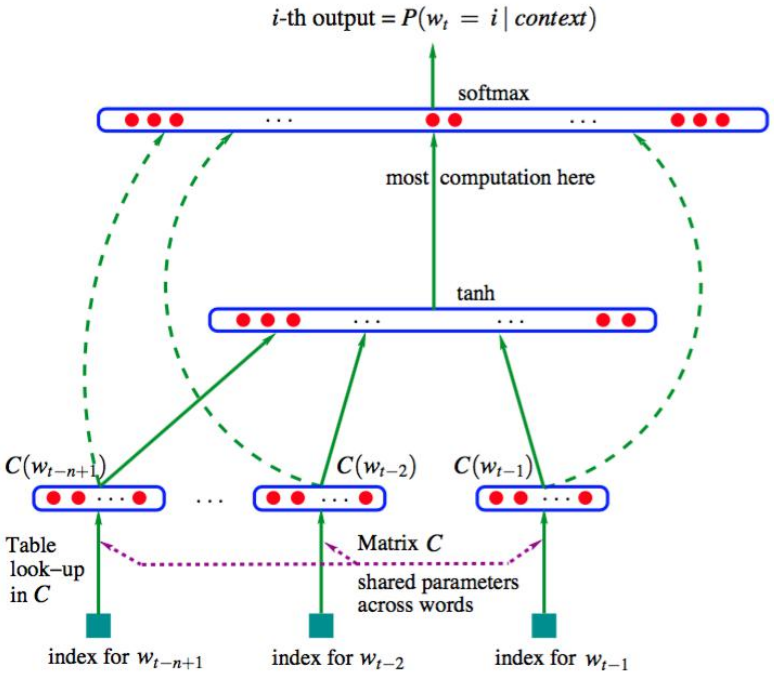
- Language Modeling: the most basic and important NLP task
- Contain a variety of **knowledge** for language understanding, e.g., linguistic knowledge and factual knowledge
- Only require the plain text **without** any human **annotations**

Language Modeling

- The language knowledge learned by language models can be **transferred** to other NLP tasks easily
- There are three representative models for transfer learning of NLP

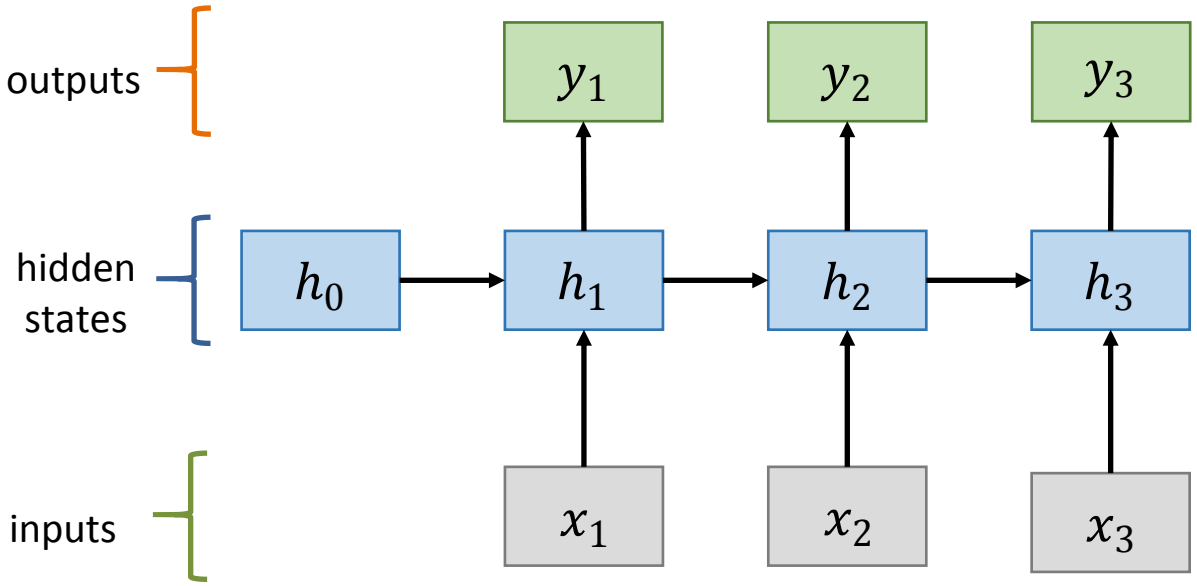
Language Modeling

- Distributed Representations of Words and Phrases and their Compositionality (word2vec)
  - The basic idea is taken from “A Neural Probabilistic Language Model”



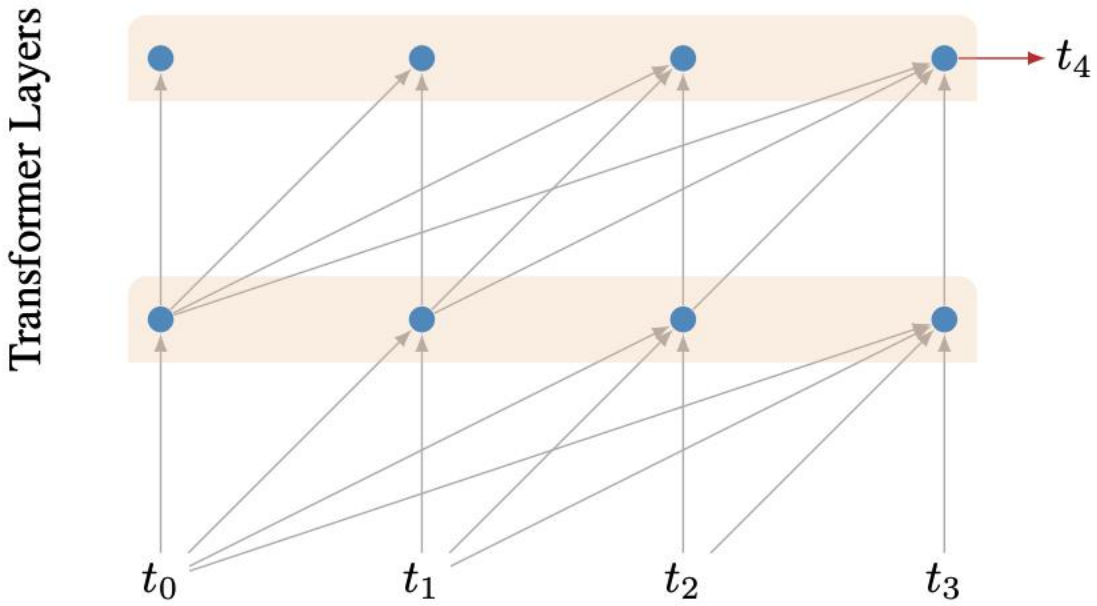
Language Modeling

- Semi-supervised Sequence Learning (Pre-trained RNN)
  - The basic idea is taken from “Recurrent neural network based language model”



Language Modeling

- GPT&BERT
  - The basic idea is taken from “Character-Level Language Modeling with Deeper Self-Attention”



Pre-trained Langue models (PLMs)

THUNLP

What are PLMs

- We have mentioned several PLMs in the last section
  - word2vec, GPT, BERT, …
- PLMs: language models having powerful **transferability** for other NLP tasks
- word2vec is the first PLM
- Nowadays, the PLMs based on Transformers are very popular (e.g. BERT)

Two Mainstreams of PLMs

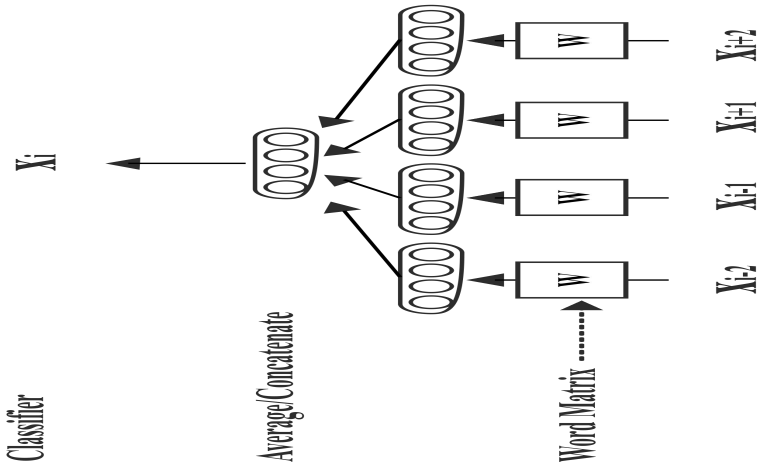
- **Feature-based** approaches
  - The most representative model of feature-based approaches is word2vec
  - Use the outputs of PLMs as the inputs of our downstream models
- **Fine-tuning** approaches
  - The most representative model of fine-tuning approaches is BERT.
  - The language models will also be the downstream models and their parameters will be updated

Development of Feature-based Approaches

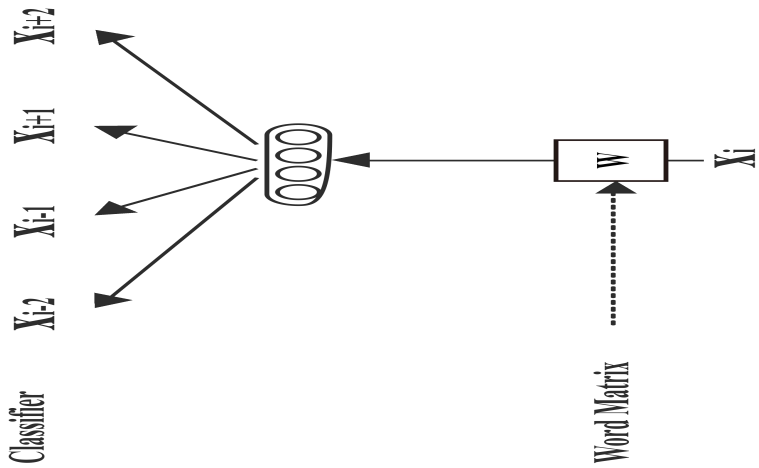
THUNLP

Review of word2vec

- Two model architectures to produce word embedding
  - continuous bag-of-words (CBOW)
  - continuous skip-gram



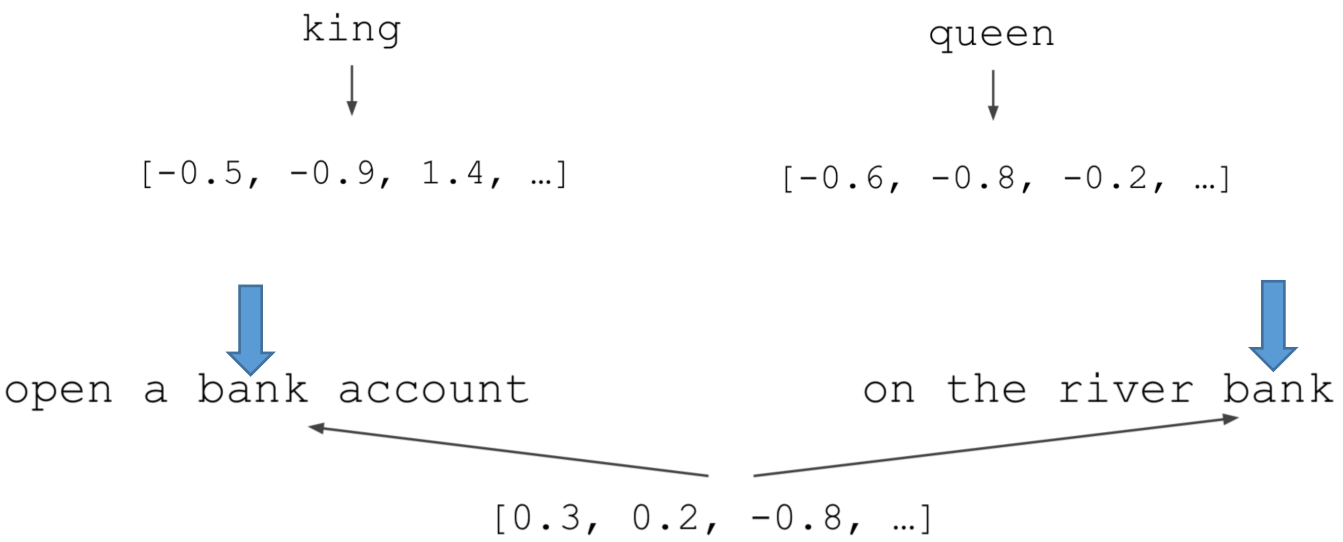
CBOW



Skip-gram

Problem of word2vec

- Word embeddings are applied in a context-free manner



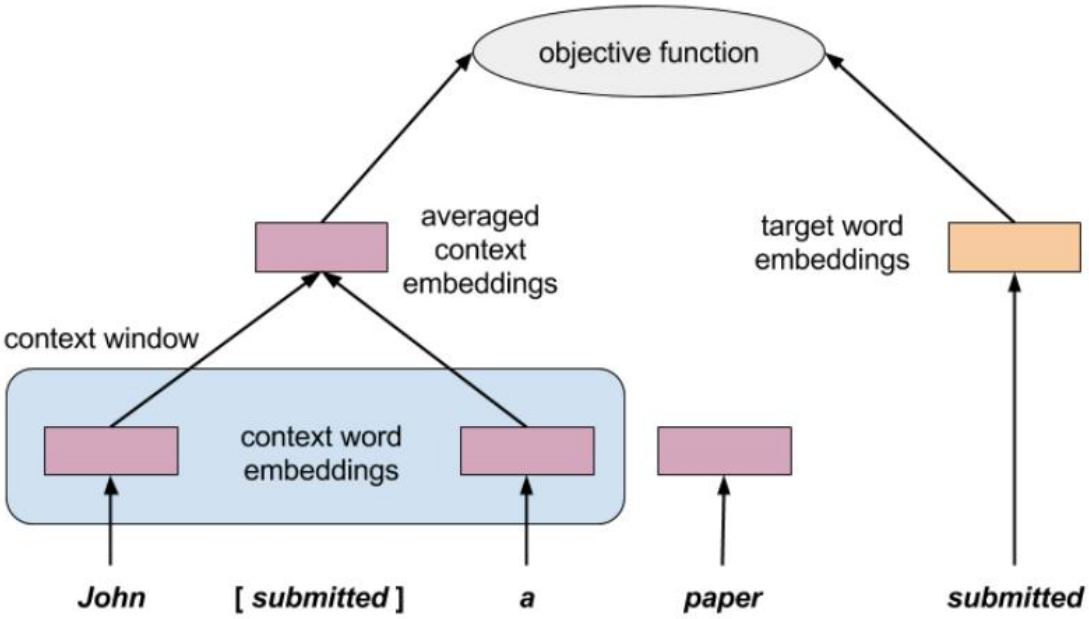
context2vec

- Considering the context-free problem of word embeddings
- Basic idea is taken from the original CBOW word2vec model
- Use one layer of Bi-LSTM to encode the context information instead of averaging the embeddings of the context words

Learning Generic Context Embedding with Bidirectional LSTM. ACL 2016.

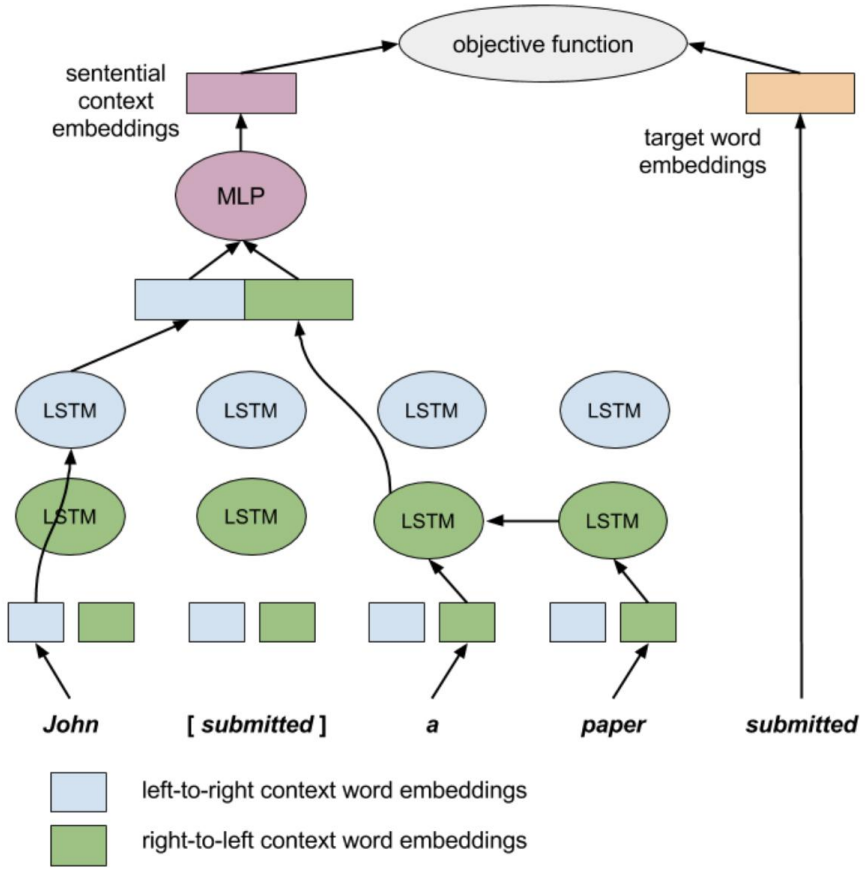
context2vec

- CBOW applies an average function to the context words to infer the contextual target word embedding
  - Avg(Embedding(John), Embedding(a))



context2vec

- context2vec applies a complex parametric network instead of a simple average function



context2vec

- Samples of the closest words to a given context

Sentential Context	Closest target words
This [ ] is due	item, fact-sheet, offer, pack, card
This [ ] is due not just to mere luck	offer, suggestion, announcement, item, prize
This [ ] is due not just to mere luck, but to outstanding work and dedication	award, prize, turnabout, offer, gift
[ ] is due not just to mere luck, but to outstanding work and dedication	it, success, this, victory, prize-money

context2vec

- The difference between word2vec and context2vec
  - For word2vec, there is only one word embedding for each word, no matter what the context is
  - For context2vec, the word embedding is **only depended on the context** without the target word
- context2vec is a preliminary work on contextualized word embedding

From word2vec to Languag Modeling

- Training paradigm of word2vec is **very efficient:** a simplified version of language modeling
- With the development of GPUs and TPUs: possible to train a language model on a large-scale corpus
- The complete training paradigm of language modeling can come up with better language understanding

ELMo

- NAACL 2018 Best paper
- Basic architecture: Stacked Bi-LSTMs
- Conduct language modeling on 1B Word Benchmark
- Combine the hidden state of each layer together as the contextualized word embedding

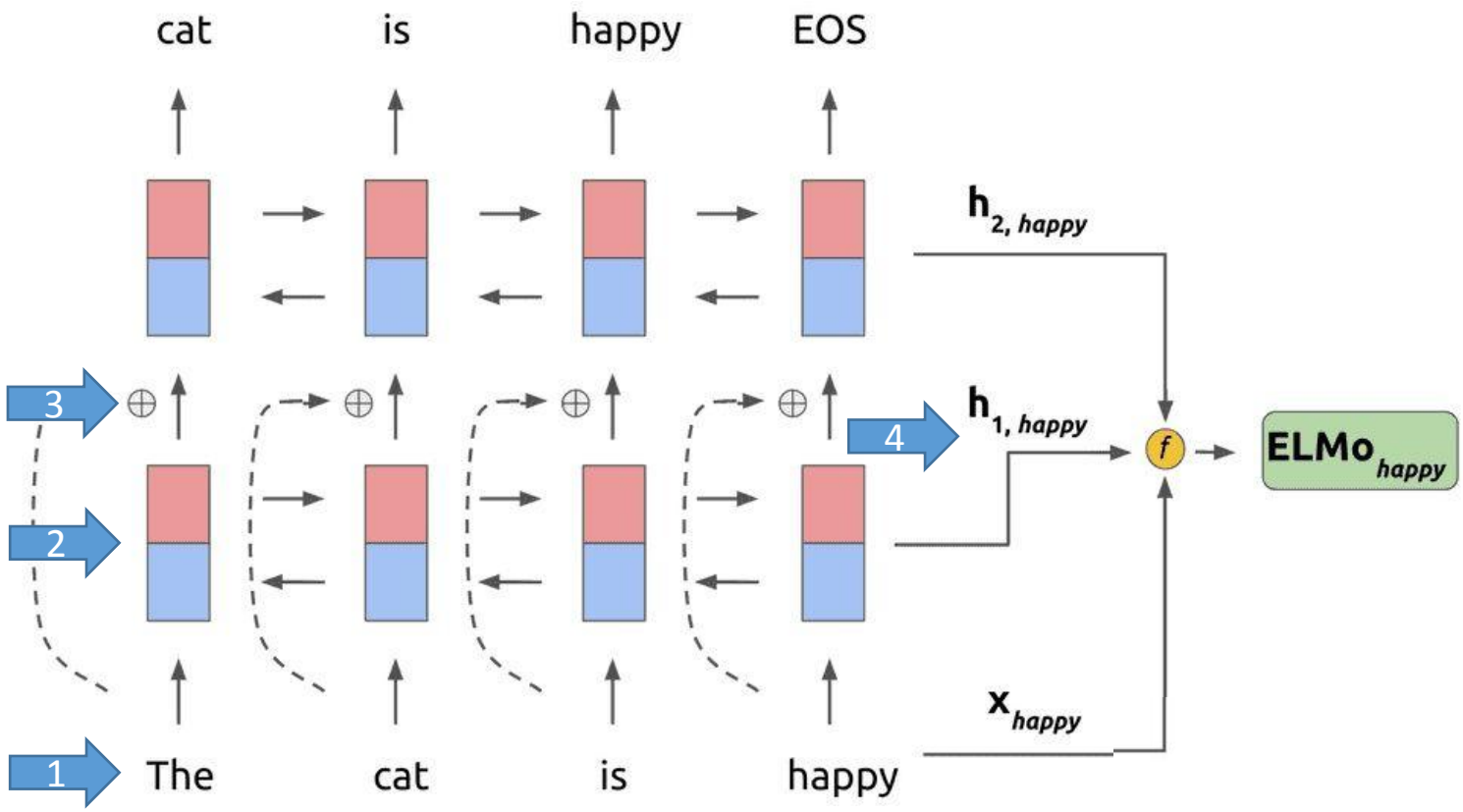
ELMo: Deep Contextual Word Embeddings. NAACL 2018.

Features of ELMo

- Contextual: The representation for each word depends on the entire context in which it is used
- Deep: The word representations combine all layers of a deep pre-trained neural network
- Character based: Robust representations for out-of-vocabulary tokens unseen in training

Details of ELMo

- Construct the contextualized embedding



Details of ELMo

- The steps for using ELMo
  - Train a Bi-directional language model in a **large corpus**
  - **Freeze the encoder** and use it as your lowest word encoder in the **downstream tasks**
  - Apply the encoder to the words and compute the contextualized word embeddings
  - Use these contextualized vectors in your new model

Results of ELMo

- The baseline methods use word2vec as the input embeddings
- The experimental results show that ELMo is better than word2vec

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%

ELMo

- Basic idea of ELMo is simple:
  - Language modeling on a large corpus
  - Deep neural model
- ELMo shows the power of language modeling, also the power of **GPUs**!
- Inspired by the success of ELMo: a variety of PLMs emerge

Development of Fine-tuning Approaches

THUNLP

Problems of RNNs

- **Gradient** Problems
  - Vanishing gradients
  - Exploding gradients
- RNNs: data-driven models requiring much data for training
- RNNs: difficult to train on small training data
- Pre-training RNNs with language modeling can solve this problem
- Fine-tuning RNNs are both **efficient and effective**

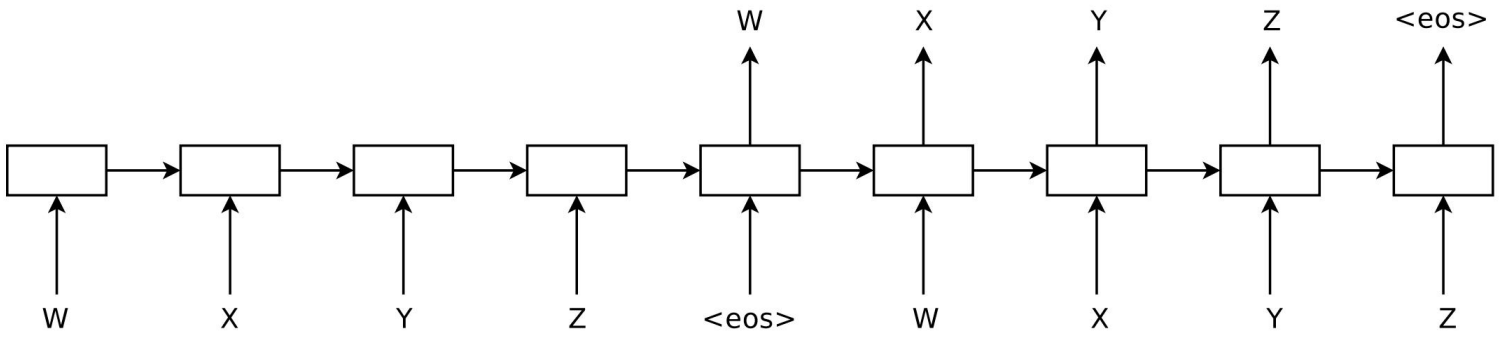
Semi-supervised Sequence Learning

- The first fine-tuning approach in NLP
- Aim to solve the training problem of RNNs
- Two approaches that use **unlabeled data** to improve sequence learning with **recurrent networks**
  - A conventional language model
  - A sequence autoencoder

Semi-supervised Sequence Learning. NIPS 2015.

Semi-supervised Sequence Learning

- Basic idea of **sequence autoencoder** is taken from the success of seq2seq models in neural machine translation
- RNN: read the input sequence into the **hidden state** which is used to r**econstruct** the original sequence



Semi-supervised Sequence Learning

- Difference between language modeling & sequence reconstruction
  - Language models predict the next word based on the **previous context**
  - A sequence autoencoder sees the **whole sequence** before decoding, trying to remember all information in the hidden state
- The hidden state after encoding can be used as a powerful sequence representation since it contains most of the information for decoding

Semi-supervised Sequence Learning

- Experimental results on text classification
- Use the pre-trained parameters to initialize downstream models
- SA-LSTM is the model pre-trained as a sequence autoencoder

A summary of the error rates of SA-LSTMs and previous best reported results.

Dataset	SA-LSTM	Previous best result
IMDB	7.24%	7.42%
Rotten Tomatoes	16.7%	18.5%
20 Newsgroups	15.6%	17.1%
DBpedia	1.19%	1.74%

Semi-supervised Sequence Learning

- Both language model and sequence autoencoder can help stabilize the learning in RNNs

- Although previous works didn't achieve good results with LSTMs, this work shows that LSTM is a powerful model for sequence modeling.

ULMFiT

- Consider the paradigm consisting of pre-training and fine-tuning
- The first step is to pre-train a language model
- The second step is to fine-tune a task-specific model
- ULMFiT introduces a new step for **fine-tuning language model**!
- This new step makes the language model **transfer** better

Universal Language Model Fine-tuning. ACL 2018.

ULMFiT: Three Steps

- Perform a language model training on a large corpus (called **LM pre-training**)
- Perform a fine-tuning on the trained language model using the down-stream task's dataset (called **LM fine-tuning**)
- Perform a fine-tuning on the language model using the task's classifier and update the model parameters using the classification objective function (called **Classifier fine-tuning**)

ULMFiT

- Why is LM fine-tuning important for transfer learning?
- The downstream dataset is usually different from the pre-training corpus
- Language modeling helps the model **adapt to the new dataset**
- LM fine-tuning can help the model transfer better and make full use of the new dataset

ULMFiT

- Experimental results on text classification

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

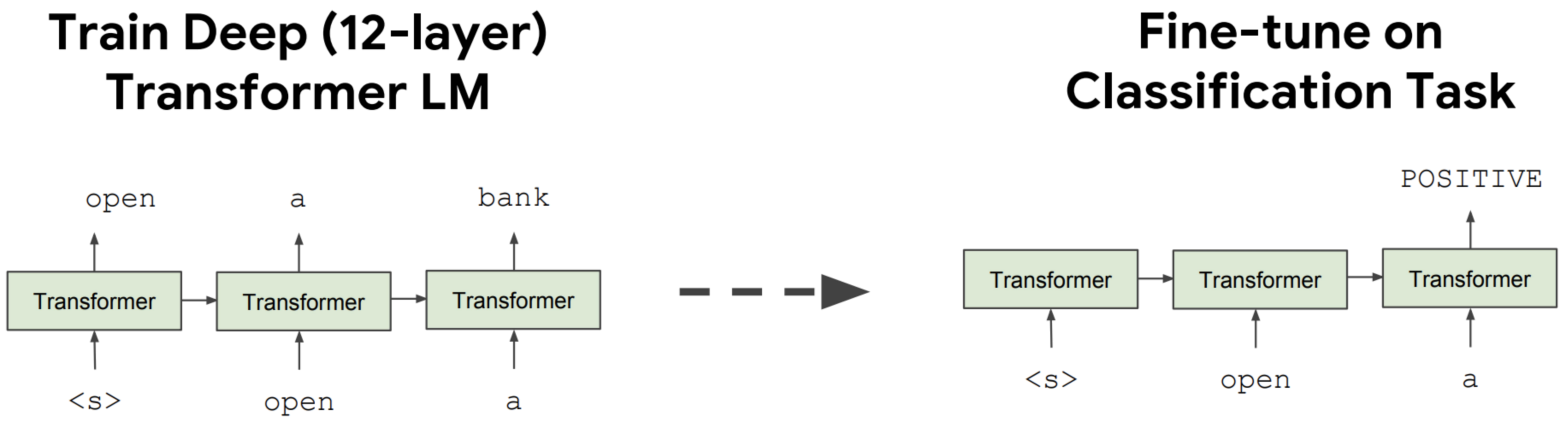
ULMFiT

- ULMFiT is an effective and extremely sample-efficient transfer learning method that can be applied to any NLP task

- In the original paper, there are several fine-grained fine-tuning strategies, which are also very interesting

GPT

- Inspired by the success of Transformers in different NLP tasks, GPT is the first work to pre-train a PLM based on **Transformer**
- Transformer + left-to-right LM
- Fine-tuned on downstream tasks



Improving Language Understanding by Generative Pre-Training. OpenAI 2018.

GPT-2

- A really big Transformer LM (also very good!)
- Trained on **40GB** of text
- SOTA perplexities on datasets it's not even trained on!

	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

Results of language modeling

More than LM

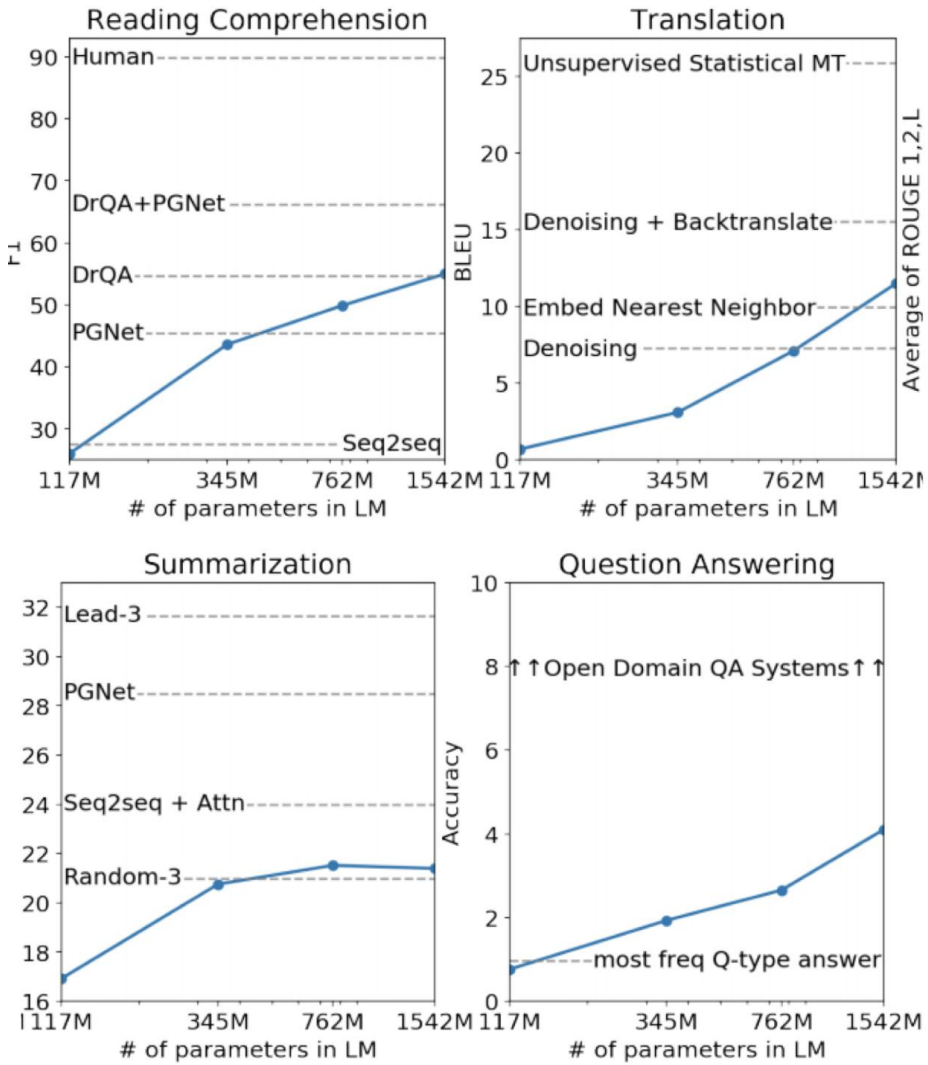
- **Zero-Shot Learning**
  - Ask LM to generate from a prompt
- Reading Comprehension:

  <context> <question> A:
- Summarization:

  <article> TL;DR:
- Question Answering:

  <question> A:

Zero-Shot Results



GPT

- A very powerful generative model
- Also achieve very good transfer learning results on downstream tasks
  - Outperform ELMo significantly

- The key to success
  - Big data (Large unsupervised corpus)
  - Deep neural model (Transformer)

BERT

- The **most popular** PLM!
- NAACL 2019 best paper
- Change the paradigm of NLP significantly!

BERT: Bi-directional Encoder Representations from Transformers. NAACL 2019.

Problem with Previous Methods

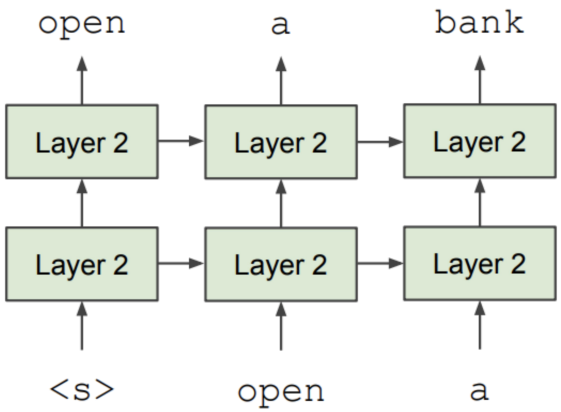
- **Problem:** Language models only use left context or right context, but language understanding is **bidirectional**

- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution
- Reason 2: Words can "see themselves" in a bidirectional encoder

Unidirectional vs. Bidirectional Models

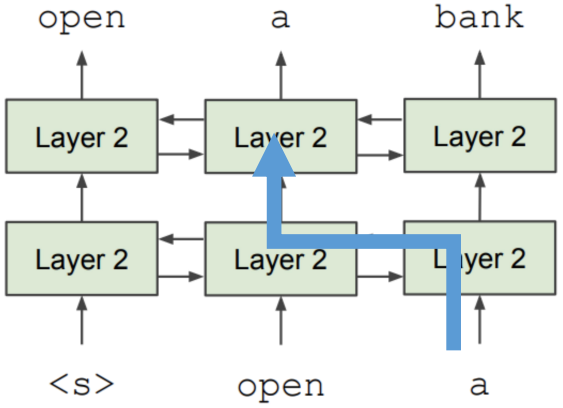
Unidirectional context

Build representation incrementally



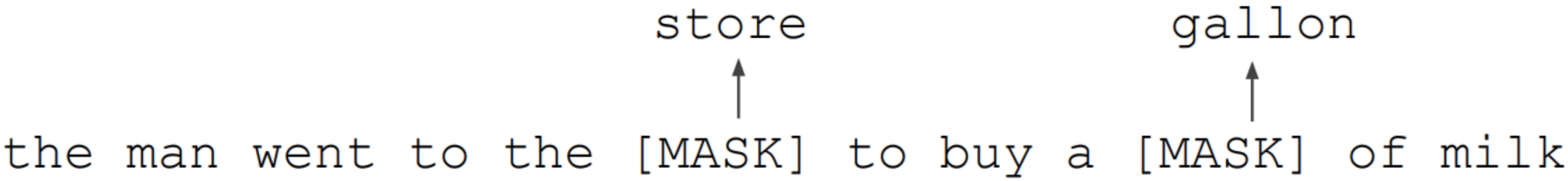
Bidirectional context

Words can “see themselves”



Masked LM

- Solution: **Mask** out k% of the input words, and then predict the masked words
  - k=15% in BERT



- Too little masking: Too expensive to train
- Too much masking: Not enough context

Masked LM

- Problem: [Mask] token **never seen** at fine-tuning
- Solution: 15% of the words to predict
- 80% of the time, replace with [MASK]

  went to the store → went to the [MASK]
- 10% of the time, replace with **a random word**

  went to the store → went to the running
- 10% of the time, keep the same

  went to the store → went to the store

Next Sentence Prediction

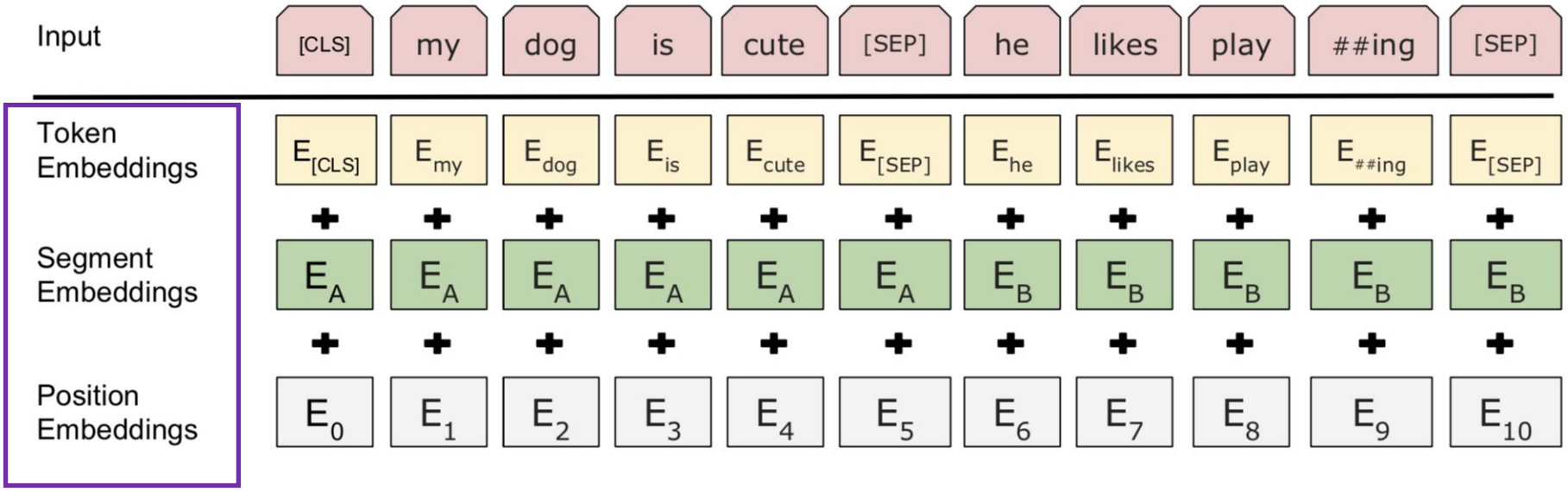
- To learn **relationships** between sentences, predict whether Sentence B is the actual sentence that **proceeds** Sentence A, or just a random sentence

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

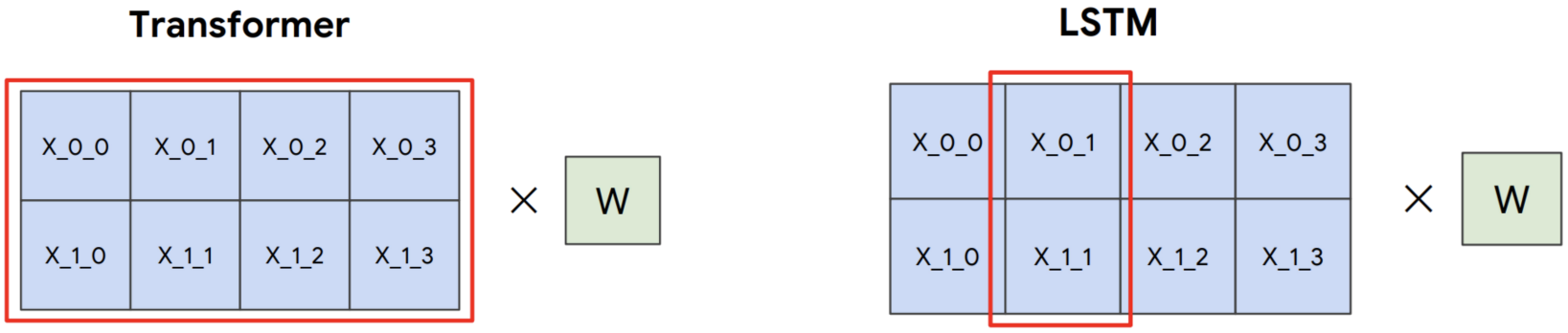
Input Representation

- Use 30,000 WordPiece vocabulary on input.
- Each token is the sum of three embeddings
- Single sequence is much more efficient.



Model Architecture

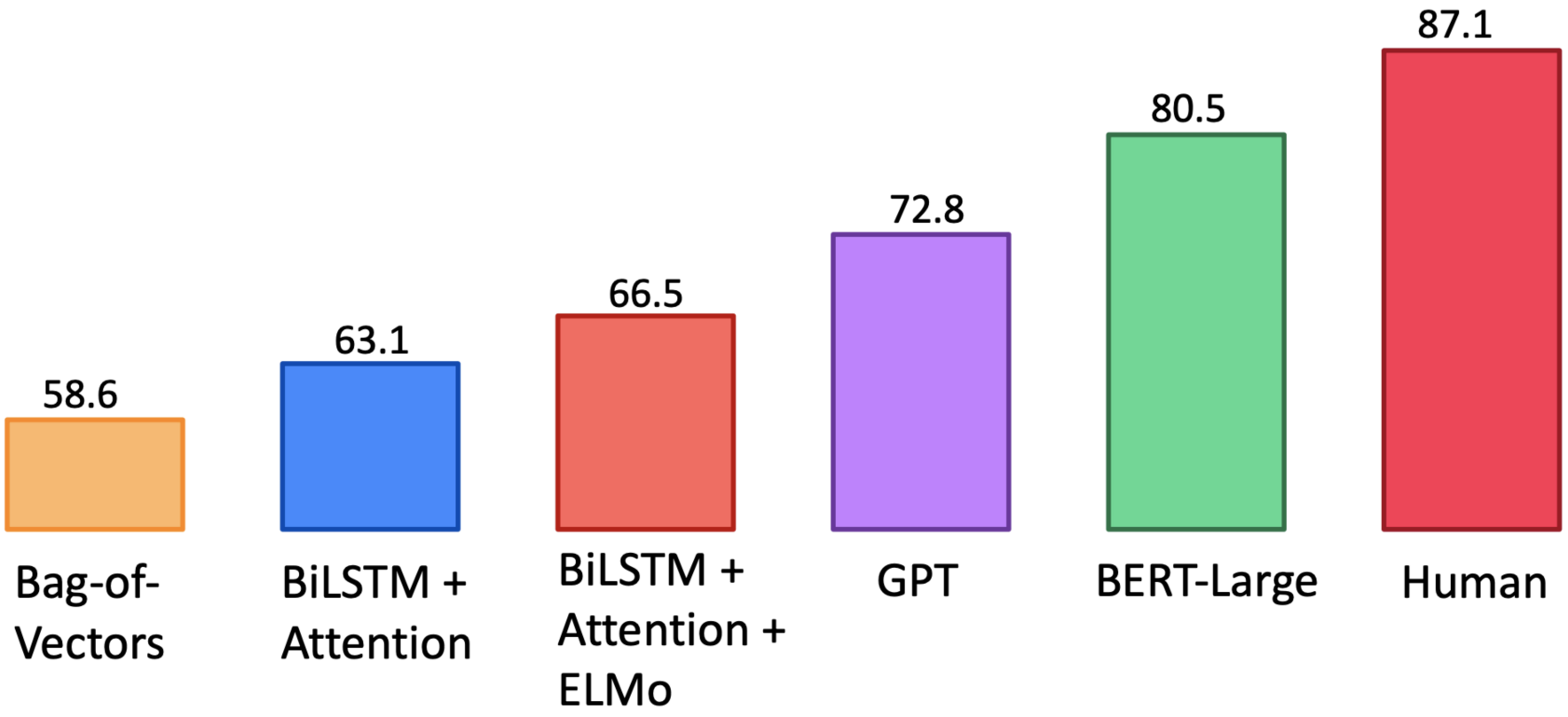
- Empirical advantages of Transformer vs. LSTM:
- Self-attention == no locality bias
  - Long-distance context has “equal opportunity”
- Single multiplication per layer == efficiency on GPU



Model Detail

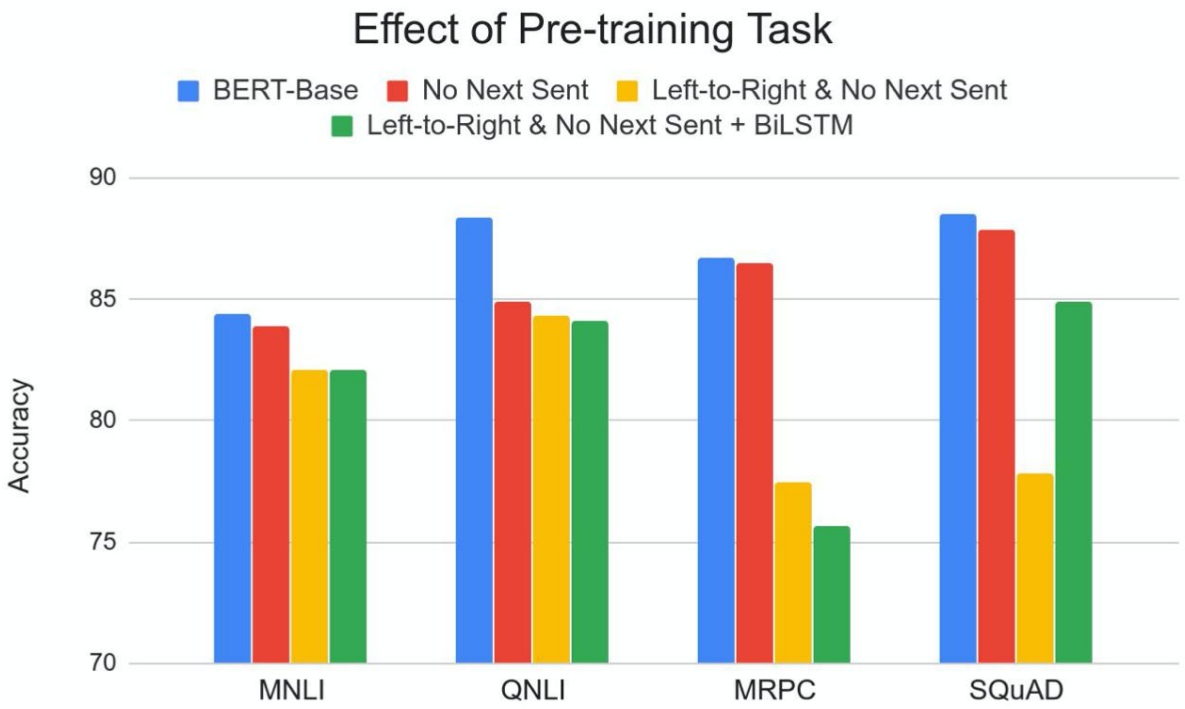
- Data: Wikipedia (2.5B words) + BookCorpus (800M words)
- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- BERT-Base: 12-layer, 768-hidden, 12-head
- BERT-Large: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPUs for 4 days

GLUE Results



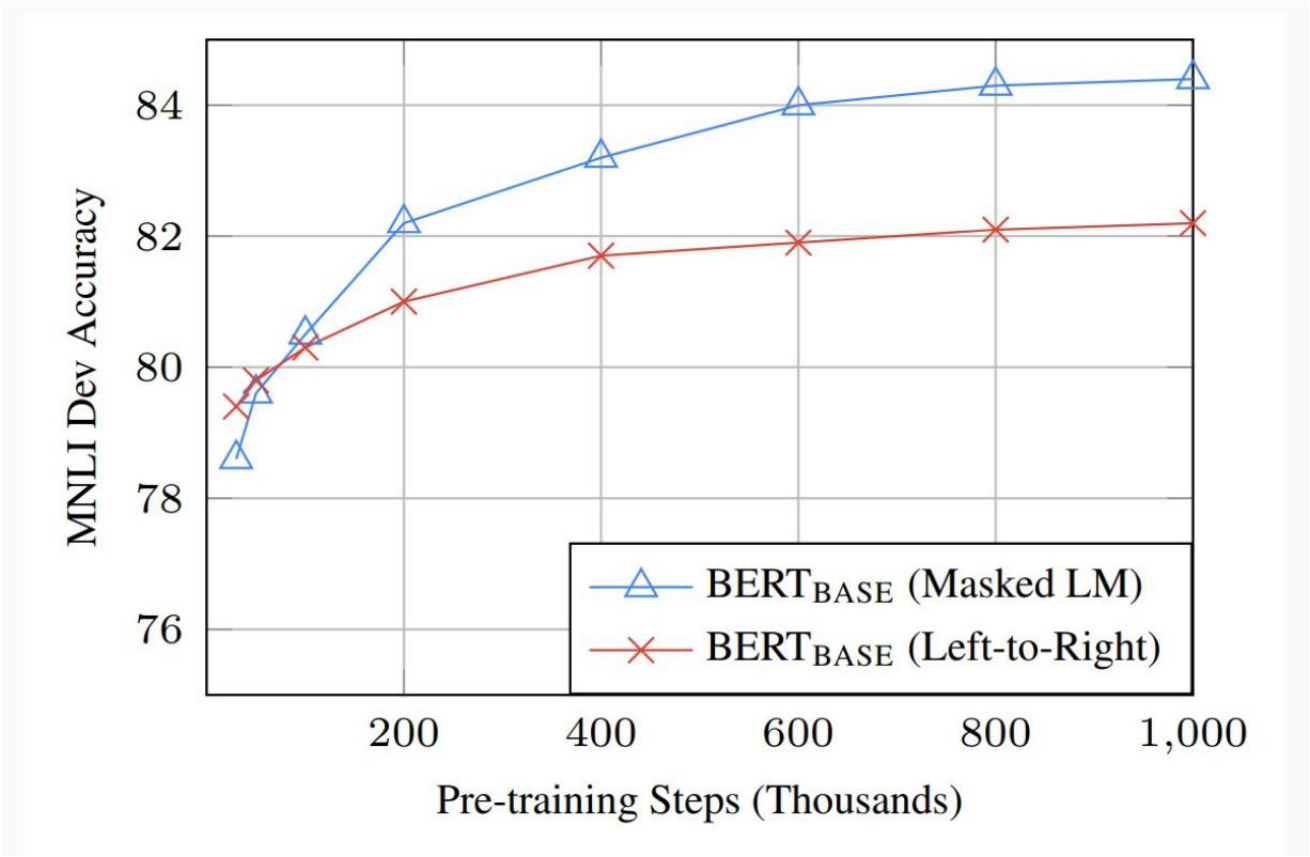
Effect of Pre-training Task

- Masked LM (compared to left-to-right LM) is very important on some tasks
- Next Sentence Prediction is important for other tasks



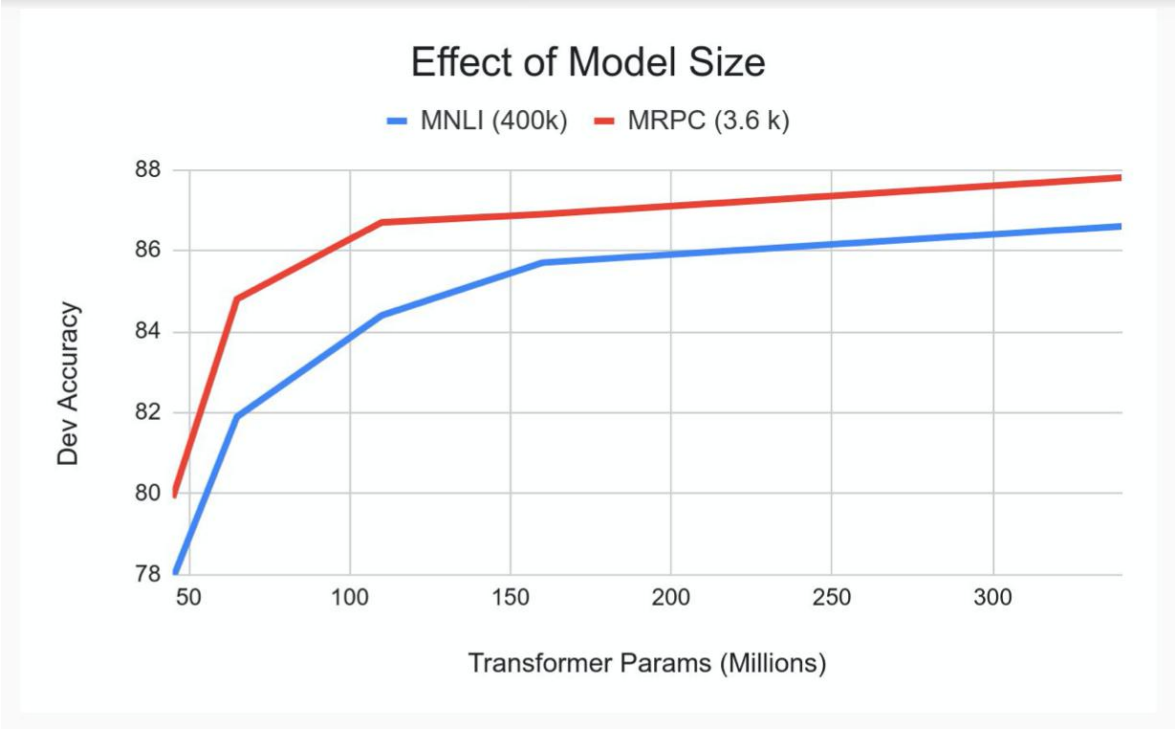
Effect of Directionality and Training Time

- Masked LM takes slightly longer to converge because it only predicts 15% instead of 100%



Effect of Model Size

- **Big** models help *a lot*
- Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples



Effect of Masking Strategy

- Masking 100% of the time hurts on feature-based approach
- Using random word 100% of time hurts slightly

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

BERT

- Empirical results from BERT are great, but biggest impact on the field is:
  - With pre-training, **bigger == better**, without clear limits (so far)

- Good for people and companies building NLP systems

Summary

- Feature-based approaches transfer the **contextualized** word embeddings for downstream tasks
- Fine-tuning approaches transfer the **whole model** for downstream tasks
- Experimental results show that fine-tuning approaches are better than feature-based approaches
- Hence, current research mainly focuses on **fine-tuning approaches**

PLMs after BERT

THUNLP

Is BERT really perfect?

- Any optimized pre-training paradigm?
- The **gap** between pre-training and fine-tuning
  - [MASK] token will not appear in fine-tuning
- The **efficiency** of Masked Language Model
  - Only predict 15% words

RoBERTa

- Explore several pre-training approaches for a more robust BERT
  - Dynamic Masking
  - Model Input Format
  - Next Sentence Prediction
  - Training with Large Batches
  - Text Encoding
- Massive experiments

RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2019.

Dynamic Masking

- The original BERT implementation performed masking once during data preprocessing
  - Static
  - Each training sequence was seen with the same mask many times

- Dynamic Masking: **generate the masking** pattern **every time** we feed a sequence to the model

Dynamic Masking

- Reference is the original result of BERT
- Static masking performs similarly to the original BERT model
- Dynamic masking is comparable or slightly better than static masking

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

Model Input Format and Next Sentence Prediction

- SEGMENT-PAIR+NSP (from the original BERT)
  - A pair of segments as input
- SENTENCE-PAIR+NSP
  - A pair of sentences as input
- FULL-SENTENCES
  - No NSP
- DOC-SENTENCES
  - No NSP
  - From the same document

Model Input Format and Next Sentence Prediction

- Using **individual sentences** hurts performance
- Removing the **NSP loss** matches or slightly improves performance

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

More Data

- The pre-training approach is better
- Large pre-training dataset is also important

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

RoBERTa

- The best training strategy of BERT
- Data and V100 are also needed
  - 160 GB of text
  - 1024 V100 GPUs for approximately one day

- Enjoy the power of RoBERTa
  - https://github.com/pytorch/fairseq

XLNet

- Recall: the gap between pre-training and fine-tuning
  - Back to **generative** model!
- Beat BERT on 20 tasks, especially reading comprehension
- Three factors
  - **Permutation** Language Modeling (PLM)
  - Transformer-XL
  - Larger Dataset

XLNet: Generalized Autoregressive Pretraining for Language Understanding. NeurIPS 2019.

PLM: two kinds of pretraining objectives

- Given a text sequence $\mathbf{X} = [x_1, \ldots, x_T]$
- Autoregressive (**AR**) language modeling

$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x')\right)},$$

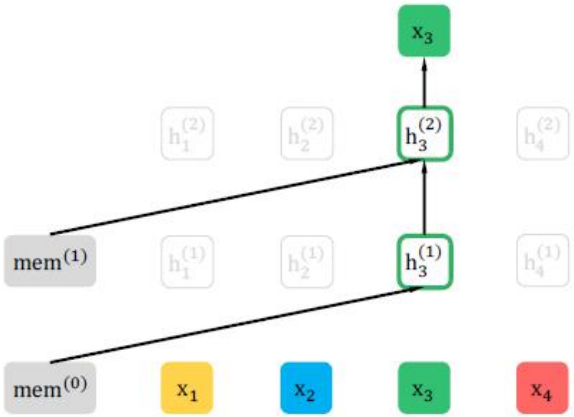
- Autoencoding (**AE**)
  - Predict the masked tokens $\bar{\mathbf{x}}$ with the corrupted sequence $\hat{\mathbf{x}}$

$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}}) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x')\right)},$$

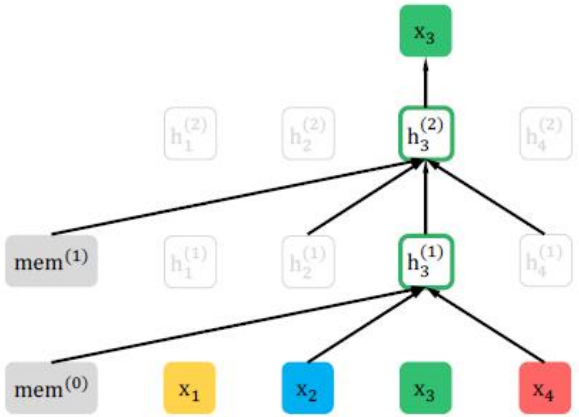
PLM: two kinds of pretraining objectives

- Two weaknesses of AE (BERT)
  - **Independency assumption:** Independently predict each masked token
  - **Input noise:** Downstream tasks do not have the [mask]
- AR does not have these weaknesses
- How to consider bidirectional context in AR formulation?
  - Permutation Language Modeling

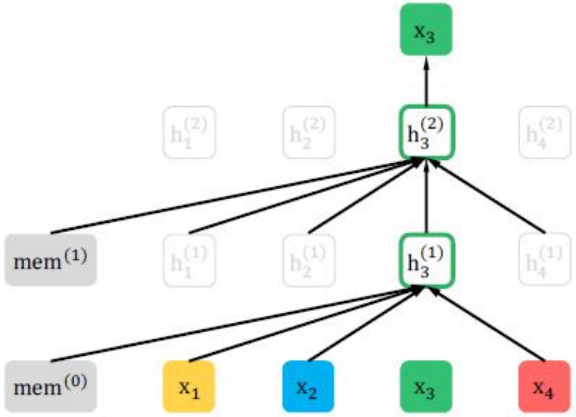
Permutation Language Modeling



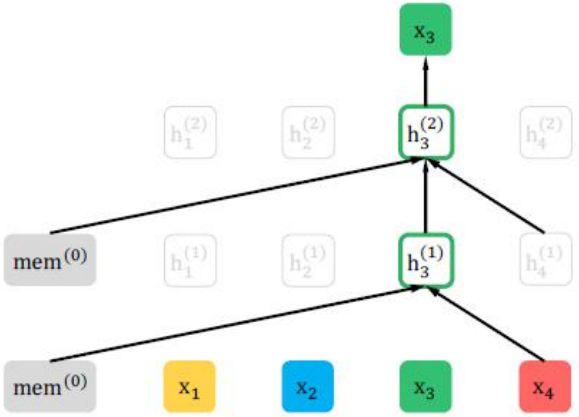
Factorization order: 3 → 2 → 4 → 1



Factorization order: 2 → 4 → 3 → 1

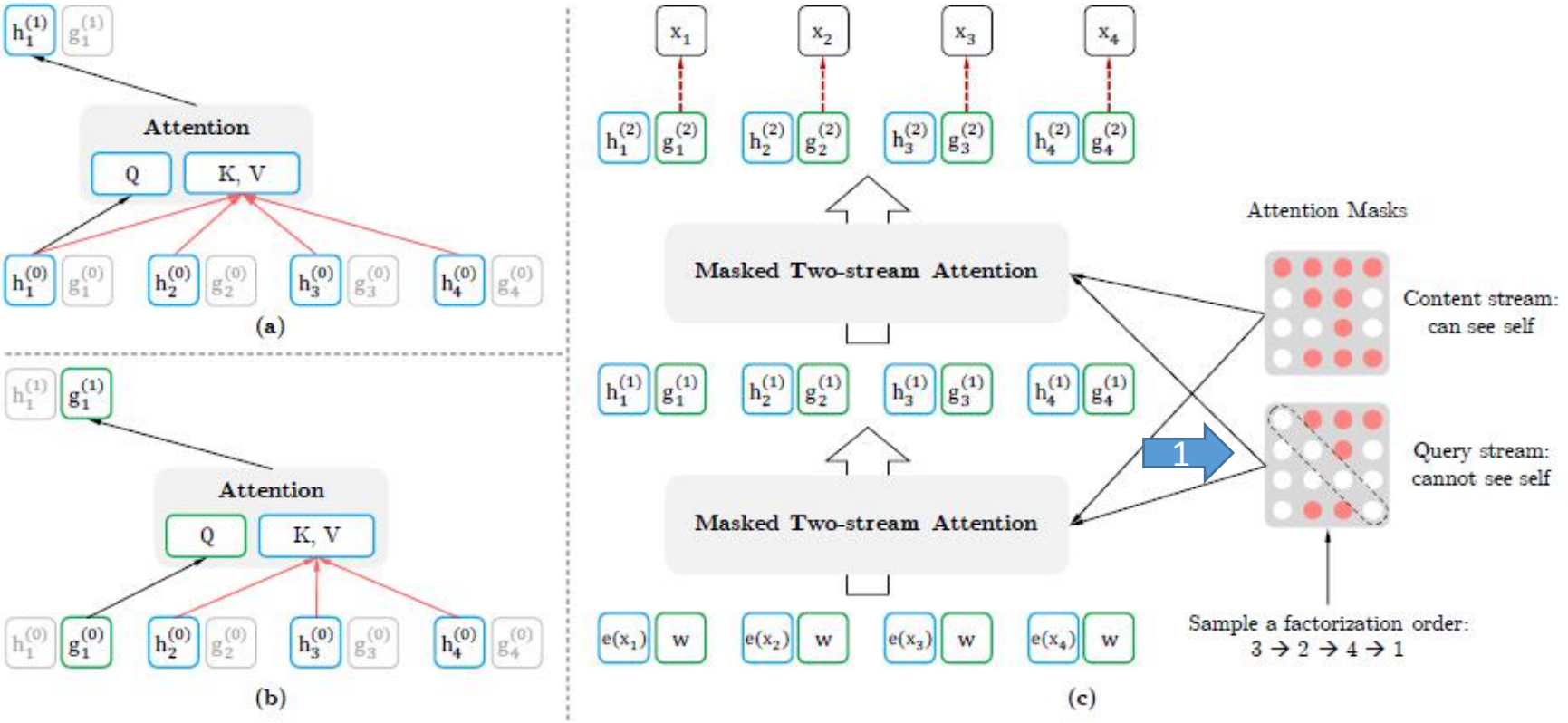


Factorization order: 1 → 4 → 2 → 3



Factorization order: 4 → 3 → 1 → 2

Two-stream self-Attention



Training Datail

- All the permutation of a sequence?
  - **Randomly sample** a permutation for each sentence
- Predict all the tokens?
  - Only predict **last 1/K tokens**
  - K=6, 7 in experiments

XLNet

- Better results than BERT
  - All models are trained using the same data and hyperparameters as in BERT
  - 3 BERT variants: the original BERT, BERT with whole word masking, and BERT without next sentence prediction

Model	SQuAD1.1	SQuAD2.0	RACE	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
BERT-Large (Best of 3)	86.7/92.8	82.8/85.5	75.1	87.3	93.0	91.4	74.0	94.0	88.7	63.7	90.2
XLNet-Large-wikibooks	88.2/94.0	85.1/87.8	77.4	88.4	93.9	91.8	81.2	94.4	90.0	65.2	91.1

ELECTRA

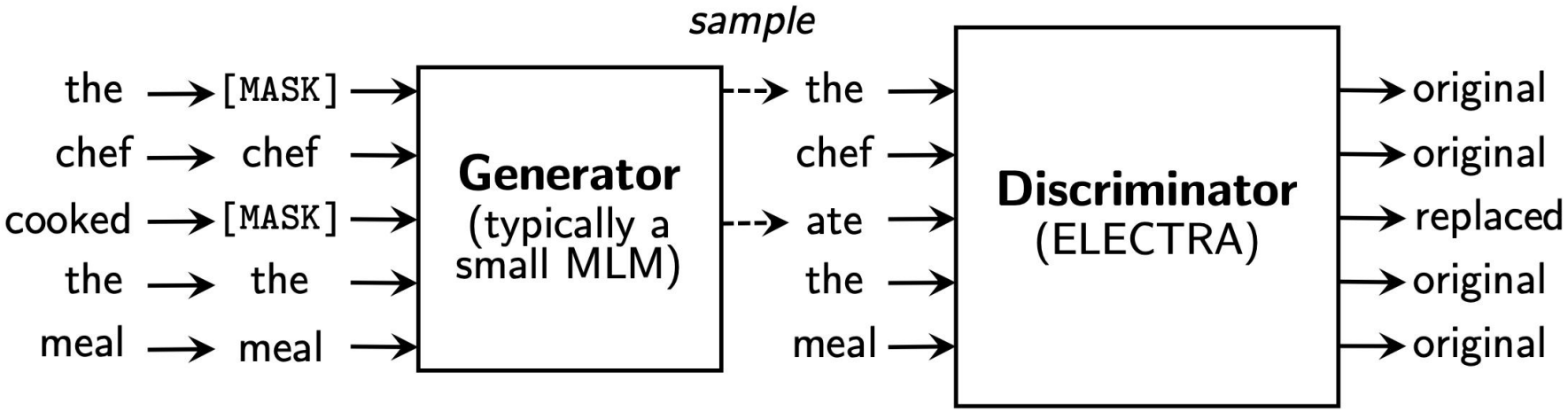
- Recall: the efficiency of bi-directional pre-training
  - Masked LM: 15% prediction
  - Premutation LM: 1/6~1/7 prediction
- Traditional LM: 100% prediction
  - Single direction

- Replaced Token Detection
  - A new bi-directional pre-training task
  - **100% prediction**

ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. ICLR 2020.

Replaced Token Detection

- The generator is trained to perform masked LM
- The generated sequence is used for Replaced Token Detection
- The discriminator predicts whether the token is “real”



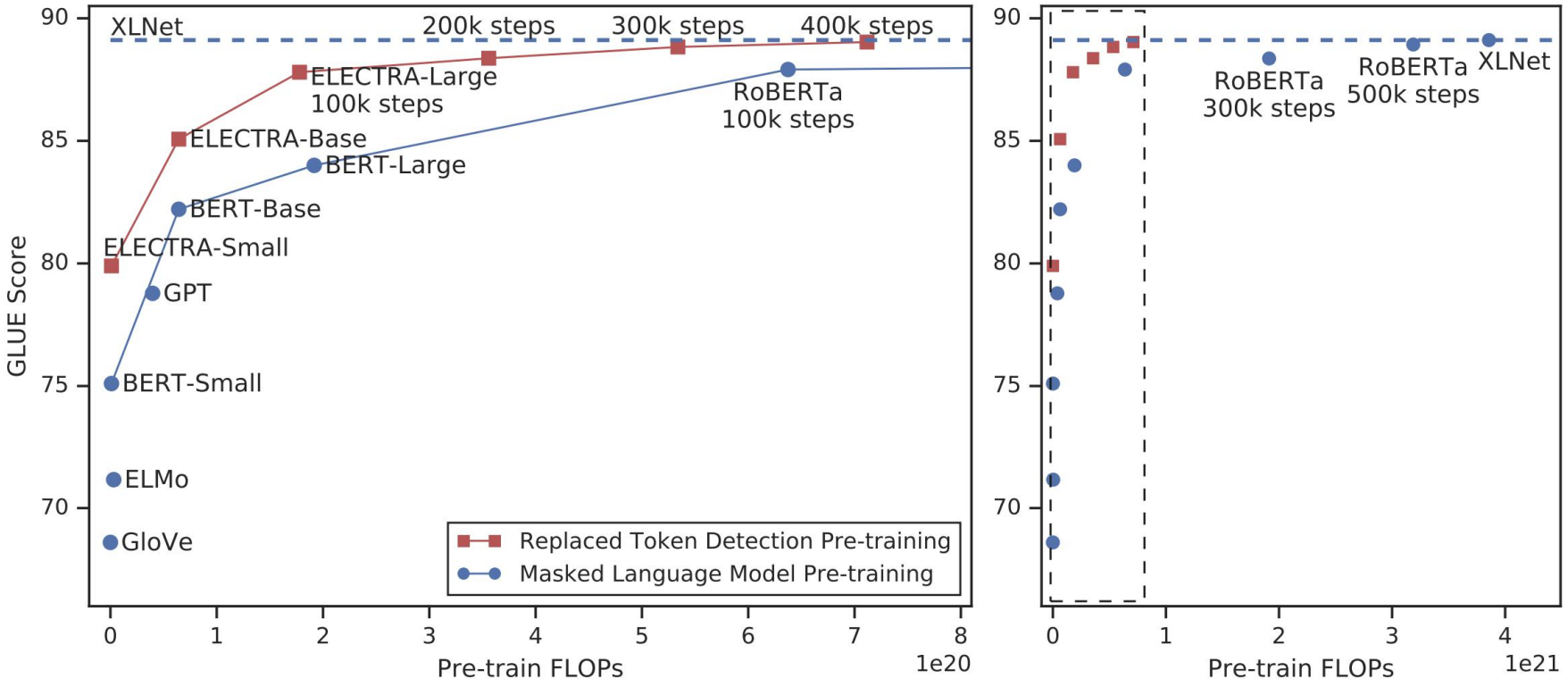
Replaced Token Detection

- **Generator**
  - Generate a particular token with a softmax layer
  - As same as conventional LM

- **Discriminator** $D(\boldsymbol{x}, t) = \text{sigmoid}(w^T h_D(\boldsymbol{x})_t)$

  - Whether the token is “real”
  - w^T is the weight matrix
  - $h_D(\boldsymbol{x})_t$ is the output hidden state of token x_t
  - All tokens will be predicted

ELECTRA


XLNet
200k steps
300k steps
400k steps
ELECTRA-Large
100k steps
ELECTRA-Base
BERT-Large
BERT-Base
ELECTRA-Small
GPT
BERT-Small
ELMo
GloVe
RoBERTa
100k steps
Replaced Token Detection Pre-training
Masked Language Model Pre-training
GLUE Score
Pre-train FLOPs
1e20
RoBERTa
300k steps
RoBERTa
500k steps
XLNet
Pre-train FLOPs
1e21

ELECTRA

- Make full use of the input sequence
- 100% prediction leads to a more sample-efficient pre-training task
- The pre-training task doesn’t need to directly model the language distribution (e.g., language modeling)

Applications of Masked LM

THUNLP

Masked LM

- Basic idea: to use **bi-direction** information to predict the target token
- Beyond token: use **multi-modal** or **multi-lingual** information together by masking
- Input the objects from different domains together and predict the target object based on the input objects

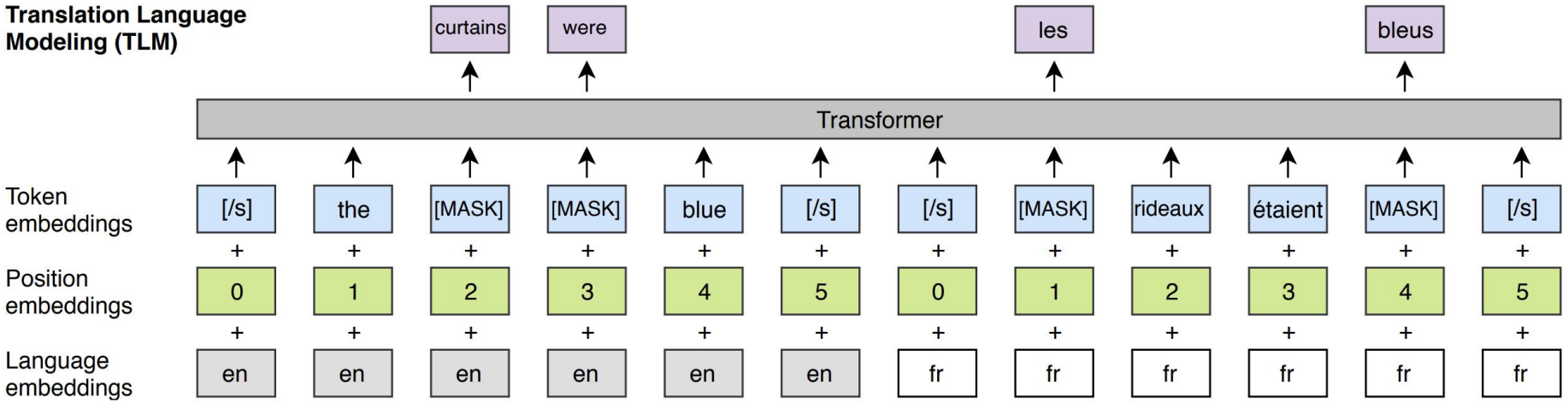
Cross-lingual LM Pre-training

- Translation Language Modeling (TLM)
- The TLM objective extends MLM to pairs of parallel sentences (e.g., English-French)
- To predict a masked English word, the model can attend to both the English sentence and its French translation, and is encouraged to **align English and French representations**

Cross-lingual Language Model Pretraining. NeurIPS 2019.

Cross-lingual LM Pre-training

- Mask both English and French tokens
- Share sub-word vocabulary
  - BPE is widely used in NMT



Cross-lingual LM Pre-training

- The translation language modeling (TLM) objective improves cross-lingual language model pretraining by leveraging **parallel data**

	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur	Δ
Machine translation baselines (TRANSLATE-TRAIN)																
Devlin et al. (2018)	81.9	-	77.8	75.9	-	-	-	-	70.7	-	-	76.6	-	-	61.6	-
XLM (MLM+TLM)	85.0	80.2	80.8	80.3	78.1	79.3	78.1	74.7	76.5	76.6	75.5	78.6	72.3	70.9	63.2	76.7
Machine translation baselines (TRANSLATE-TEST)																
Devlin et al. (2018)	81.4	-	74.9	74.4	-	-	-	-	70.4	-	-	70.1	-	-	62.1	-
XLM (MLM+TLM)	85.0	79.0	79.5	78.1	77.8	77.6	75.5	73.7	73.7	70.8	70.4	73.6	69.0	64.7	65.1	74.2
Evaluation of cross-lingual sentence encoders																
Conneau et al. (2018b)	73.7	67.7	68.7	67.7	68.9	67.9	65.4	64.2	64.8	66.4	64.1	65.8	64.1	55.7	58.4	65.6
Devlin et al. (2018)	81.4	-	74.3	70.5	-	-	-	-	62.1	-	-	63.8	-	-	58.3	-
Artetxe and Schwenk (2018)	73.9	71.9	72.9	72.6	73.1	74.2	71.5	69.7	71.4	72.0	69.2	71.4	65.5	62.2	61.0	70.2
XLM (MLM)	83.2	76.5	76.3	74.2	73.1	74.0	73.1	67.8	68.5	71.2	69.2	71.9	65.7	64.6	63.4	71.5
XLM (MLM+TLM)	**85.0**	**78.7**	**78.9**	**77.8**	**76.6**	**77.4**	**75.3**	**72.5**	**73.1**	**76.1**	**73.2**	**76.5**	**69.6**	**68.4**	**67.3**	**75.1**

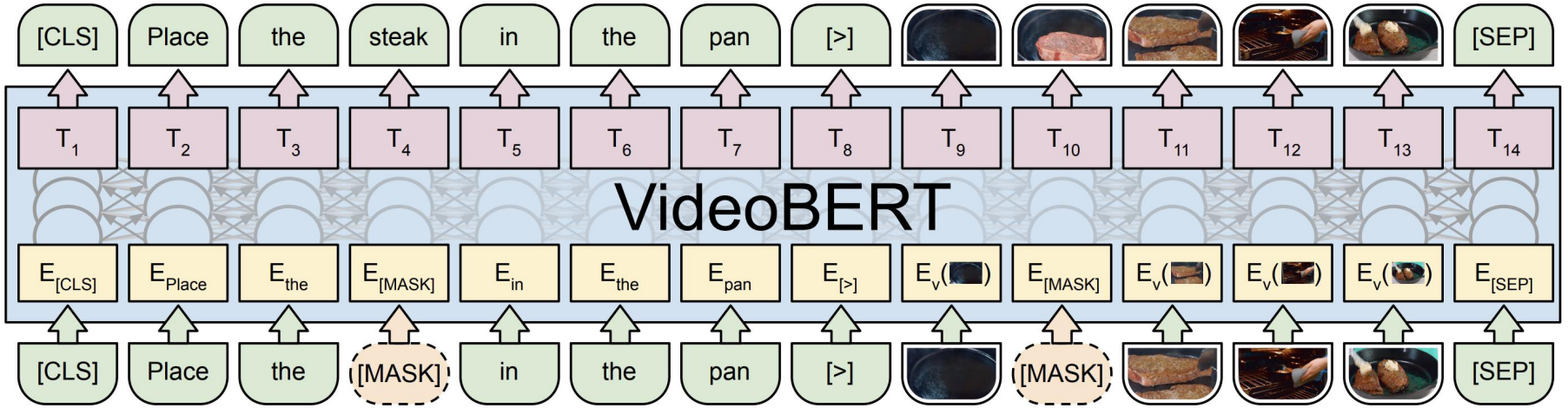
Cross-Modal LM Pre-training

- Pairs of videos and texts from automatic speech recognition (ASR)
- Generate a sequence of “visual words” by applying hierarchical vector quantization to features derived from the video using a pre-trained model
- Encourages the model to focus on high-level semantics and longer-range temporal dynamics in the video

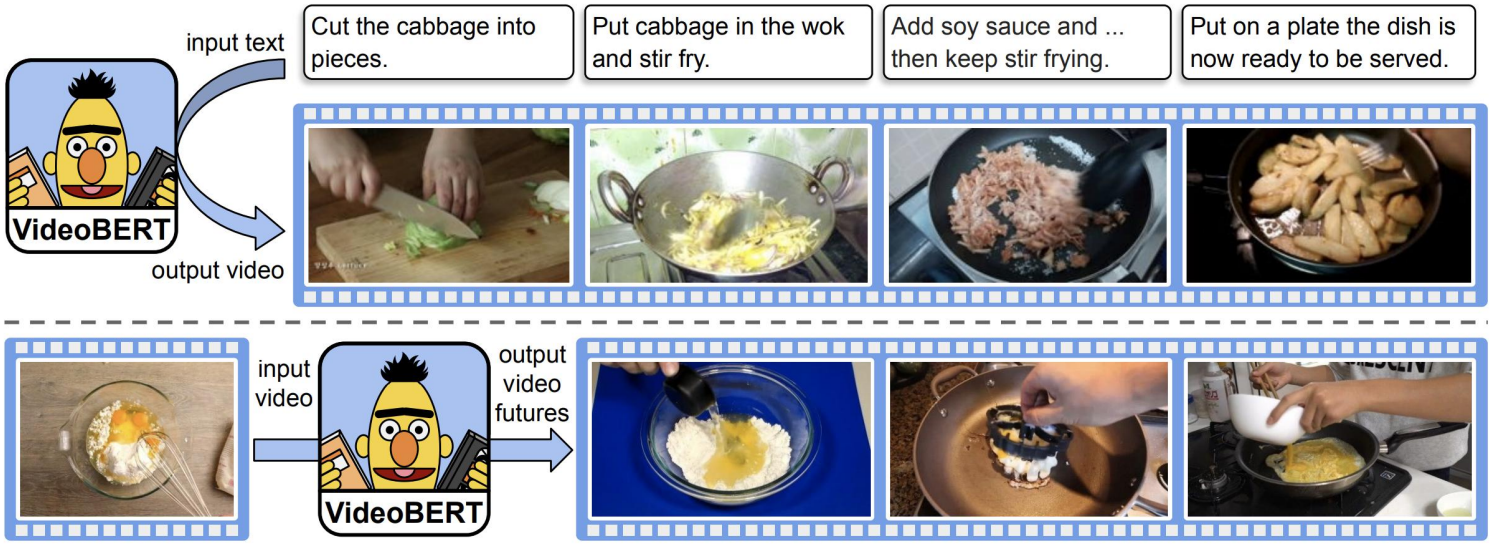
VideoBERT: A Joint Model for Video and Language Representation Learning. ICCV 2019.

Cross-Modal LM Pre-training

- Two tasks
  - A linguistic-visual alignment task ([CLS])
  - Mask prediction task ([MASK])



Cross-Modal LM Pre-training



GT: add some chopped basil leaves into it

VideoBERT: chop the basil and add to the bowl

GT: cut the top off of a french loaf

VideoBERT: cut the bread into thin slices

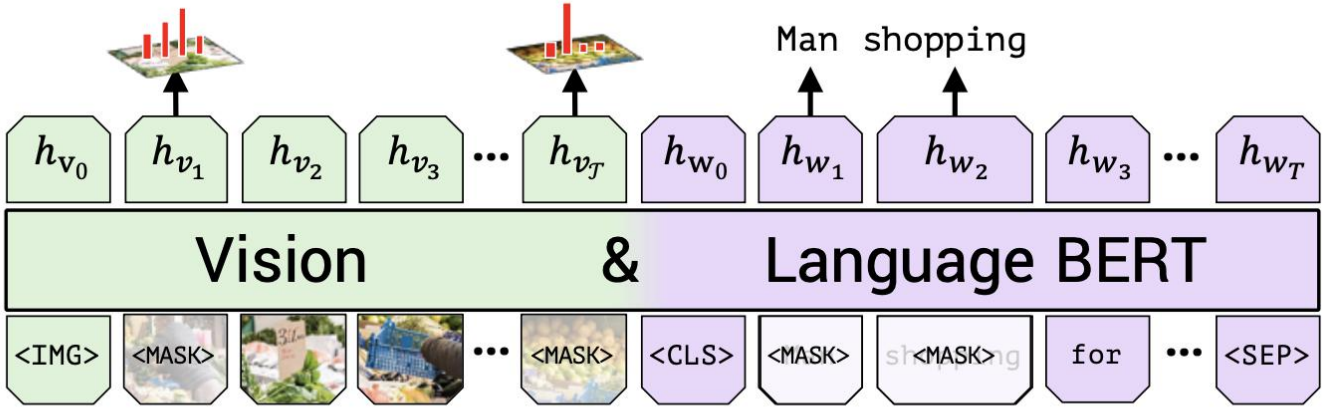
Video captioning examples

Cross-Modal LM Pre-training

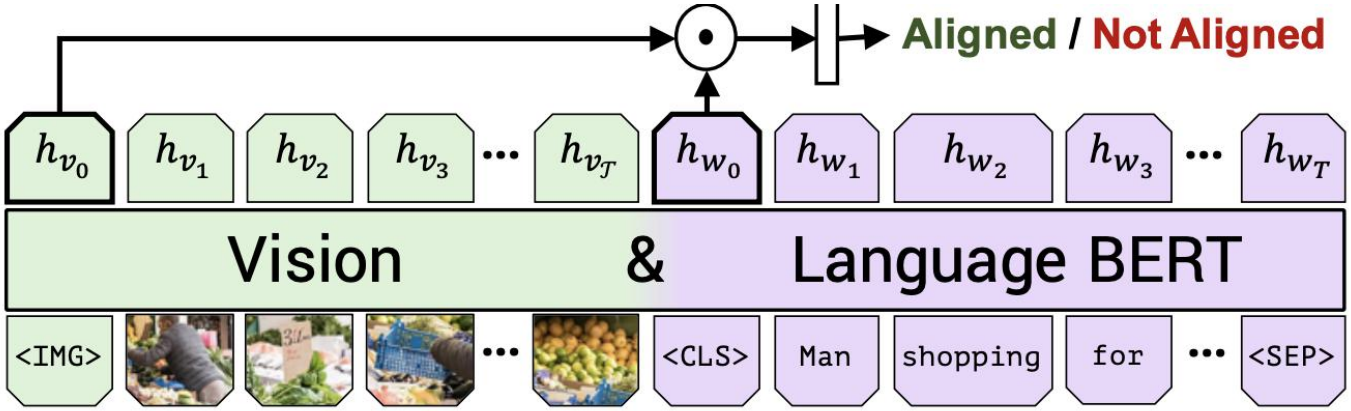
- Pairs of images and captions
- Use a pre-trained object detection network to generate an object sequence for a given image
- Visual features from the pre-trained object detection network are called “image **region** features”
- Learn task-agnostic joint representations of image content and natural language

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. NeurIPS 2019.

Cross-Modal LM Pre-training



(a) Masked multi-modal learning



(b) Multi-modal alignment prediction

Summary

- Masked LM inspired a variety of new pre-training tasks
- What's your idea about transferring Masked LM?

Thanks

Zhiyuan Liu

liuzy@tsinghua.edu.cn

THUNLP

Analysis of PLMs

THUNLP

Linguistic Knowledge in PLMs

- What features of language do these vectors capture, and what do they miss?
- How and why does transferability vary across representation layers in contextualizers?
- How does the choice of pretraining task affect the vectors' learned linguistic knowledge and transferability?

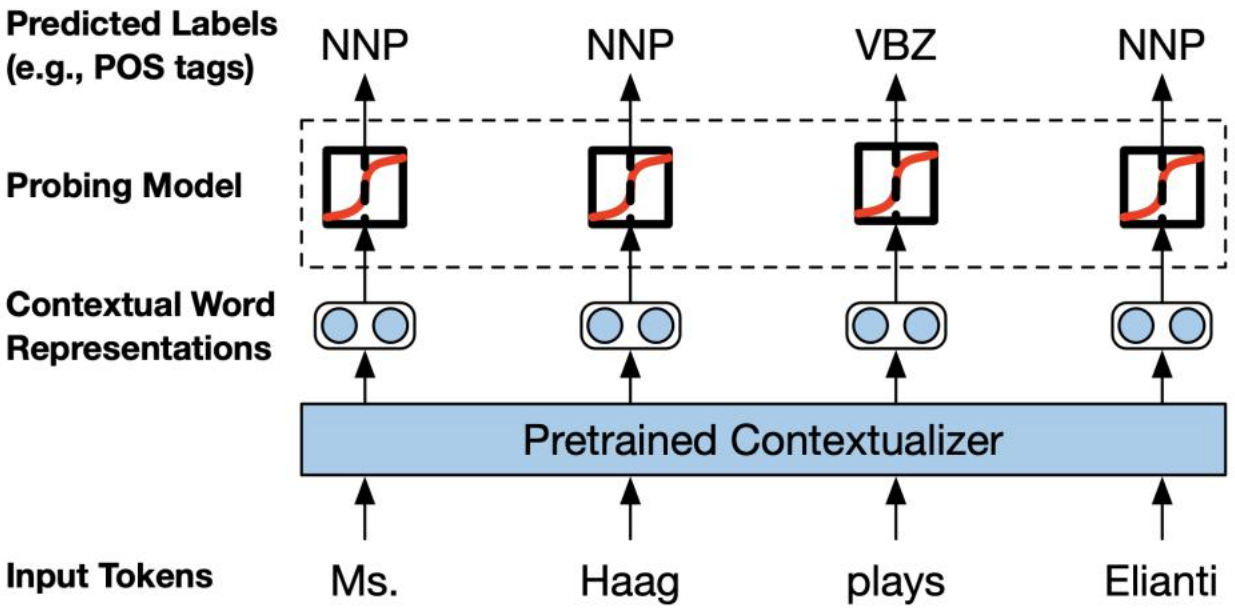
Linguistic Knowledge and Transferability of Contextual Representations. NAACL 2019.

Probing Tasks

- Token Labeling
  - Part-of-speech, CCG super-tagging, …
- Segmentation
  - Syntactic chunking, named entity recognition, …
- Pairwise Relations
  - Arc prediction, arc classification, …

Probing Model

- Input the contextualized word representations
- Token Labeling & Segmentation: single representation
- Pairwise Relation: a pair of representations



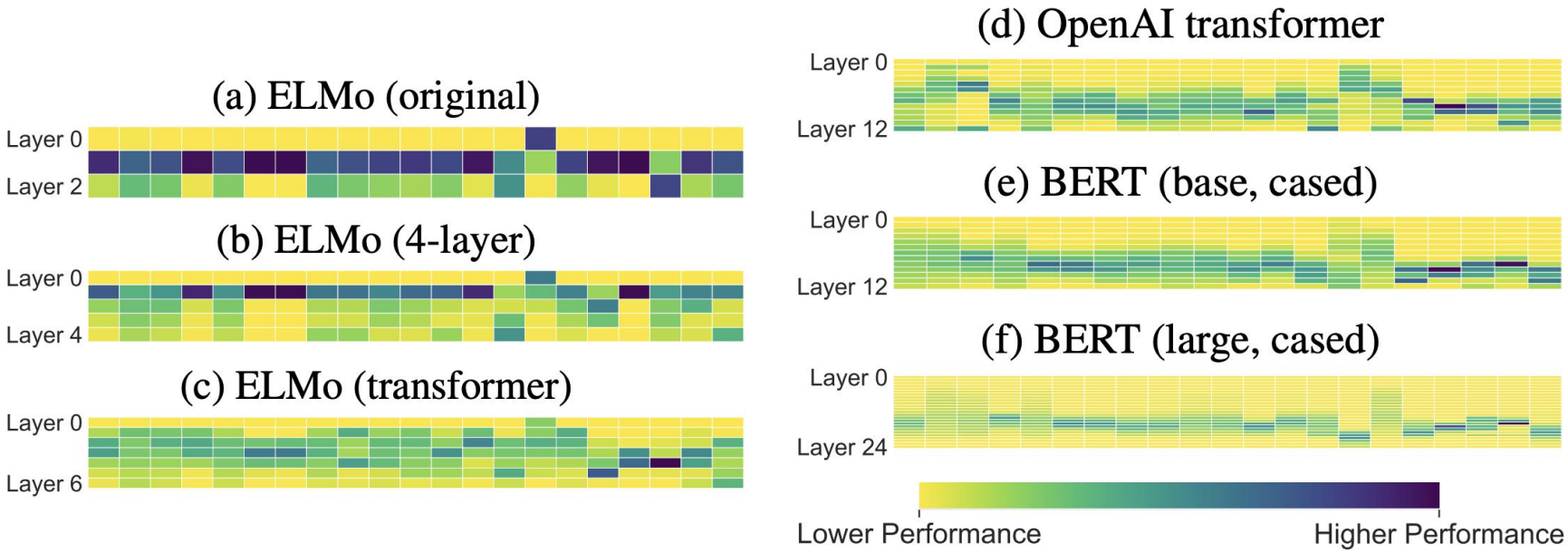
Capture & Miss

- Contextualized vectors are better than GloVe
- BERT > ELMo > GPT
- Fail on NER and grammatical error detection

Pretrained Representation			POS						Supersense ID		
	Avg.	CCG	PTB	EWT	Chunk	NER	ST	GED	PS-Role	PS-Fxn	EF
ELMo (original) best layer	81.58	93.31	97.26	95.61	90.04	82.85	93.82	29.37	75.44	84.87	73.20
ELMo (4-layer) best layer	81.58	93.81	**97.31**	95.60	89.78	82.06	**94.18**	29.24	74.78	85.96	73.03
ELMo (transformer) best layer	80.97	92.68	97.09	95.13	93.06	81.21	93.78	30.80	72.81	82.24	70.88
OpenAI transformer best layer	75.01	82.69	93.82	91.28	86.06	58.14	87.81	33.10	66.23	76.97	74.03
BERT (base, cased) best layer	84.09	93.67	96.95	95.21	92.64	82.71	93.72	43.30	**79.61**	87.94	75.11
BERT (large, cased) best layer	**85.07**	**94.28**	96.73	**95.80**	**93.64**	**84.44**	93.83	**46.46**	79.17	**90.13**	**76.25**
GloVe (840B.300d)	59.94	71.58	90.49	83.93	62.28	53.22	80.92	14.94	40.79	51.54	49.70
Previous state of the art (without pretraining)	83.44	94.7	97.96	95.82	95.77	91.38	95.15	39.83	66.89	78.29	77.10

Transferability of Layers

- Higher layers achieve lower perplexities
- The representations that are better-suited for language modeling are also those that exhibit worse probing task performance



Effect of Pre-training Tasks

- See the same tokens
- Different supervision

- BiLM is the best
- Need large corpora

Pretraining Task	Layer Average Target Task Performance			
	0	1	2	Mix
CCG	56.70	64.45	63.71	66.06
Chunk	54.27	62.69	63.25	63.96
POS	56.21	63.86	64.15	65.13
Parent	54.57	62.46	61.67	64.31
GParent	55.50	62.94	62.91	64.96
GGParent	54.83	61.10	59.84	63.81
Syn. Arc Prediction	53.63	59.94	58.62	62.43
Syn. Arc Classification	56.15	64.41	63.60	66.07
Sem. Arc Prediction	53.19	54.69	53.04	59.84
Sem. Arc Classification	56.28	62.41	61.47	64.67
Conj	50.24	49.93	48.42	56.92
BiLM	66.53	65.91	65.82	66.49
GloVe (840B.300d)	60.55			
Untrained ELMo (original)	52.14	39.26	39.39	54.42
ELMo (original) (BiLM on 1B Benchmark)	64.40	79.05	77.72	78.90

Summary

- The technique of PLMs is very important for NLP (from word2vec to BERT)
- Fine-tuning approaches are widely used after BERT
- The idea of Masked LM inspired the research on unsupervised learning
- Consider PLMs first when you plan to construct a new NLP system

Reading Material

a. Pre-Trained Language Models

- **Must-read Papers**

· Semi-supervised Sequence Learning. [link]

· (ELMo) Deep contextualized word representations. [link]

· BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [link]

- **Further Reading**

· Introduction of Pre-trained LM. [link]

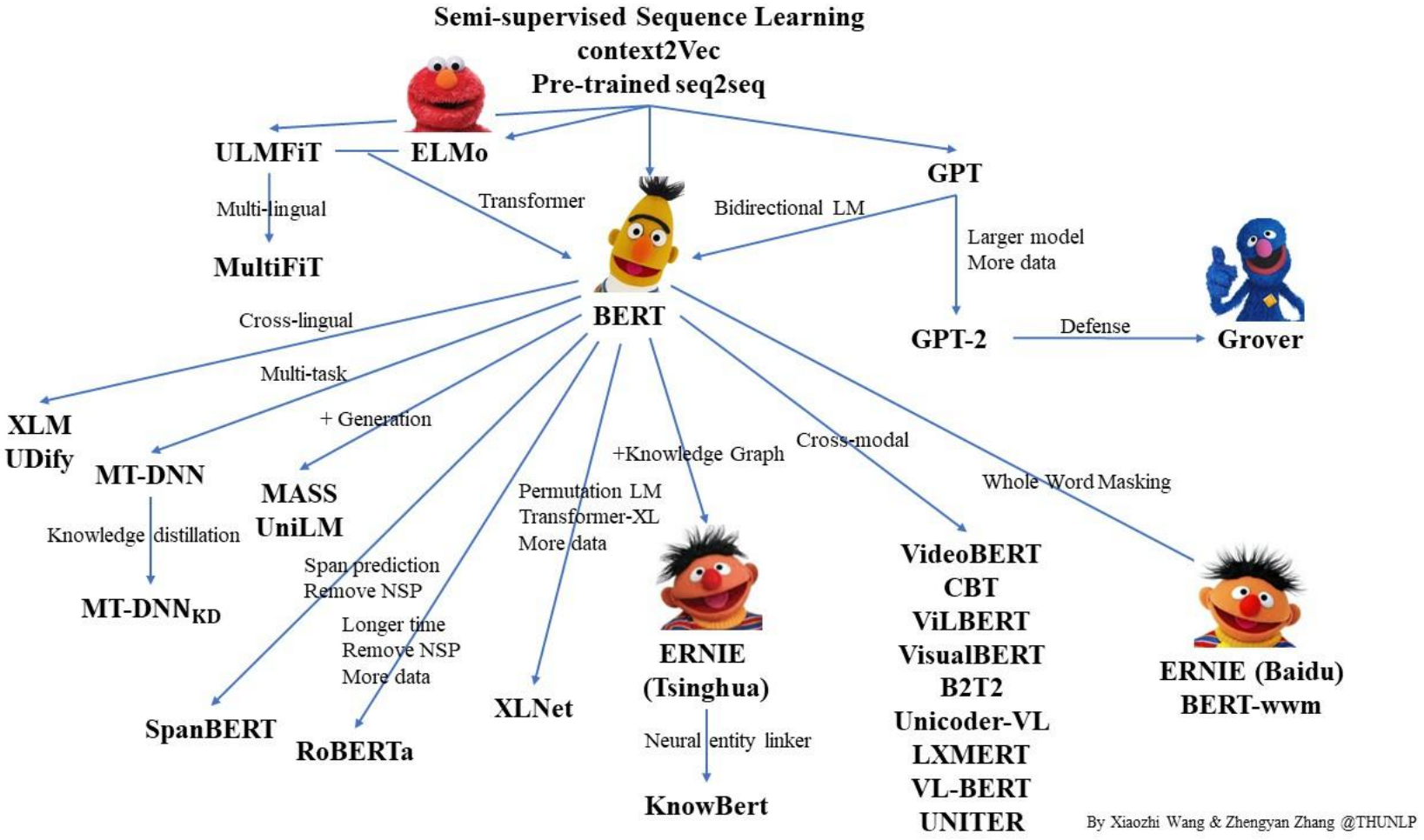
· Transformer code repo. [link]

· Transfer Learning in Natural Language Processing. Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, Thomas Wolf. NAACL 2019 [link]

· PLM paper list. [link]

For reading material recommendation of this course, please refer to our github.

Paper List



https://github.com/thunlp/PLMpapers