자연어처리의 최신동향-3 : Transformer기반 사전학습모델(Pre-trained Models)

Hyopil Shin(Seoul National University)

hpshin@snu.ac.kr http://knlp.snu.ac.kr

자연어처리의 최신 동향:

1. 트랜스포머(Transformer)에 이르기까지의 자연어처리의 진화 : 언어 모델링(Language Modeling)관점에서의 진화(Evolutions): N-Grams, Word Embeddings, Sequence to Sequence Model, Attention, and Transformers

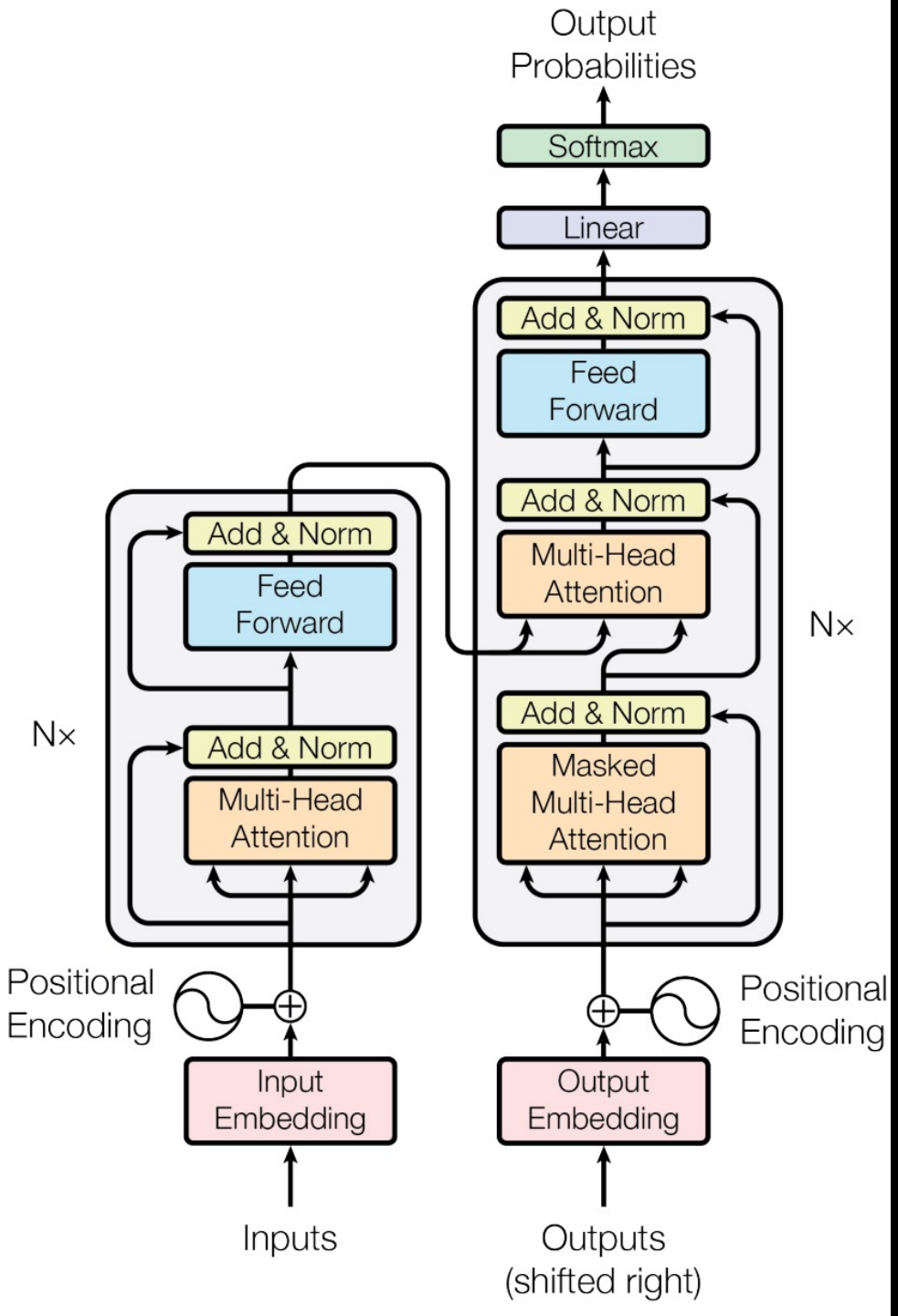
2. 트랜스포머(Transformer): 2017년 이후 자연어처리의 Game Changer가 된 Transformer

3. 트랜스포머기반 사전학습모델(Pre-Trained Models): Transformer Architecture를 활용한 다양한 종류의 사전학습모델(Autoregressive, Autoencoding, sequence to sequence, and multimodal Models)

4. 트랜스포머를 활용한 응용: Transformer와 사전학습모델의 Semantic Search, Sentence-Bert, Question Answering, Summarization, Chatbot, Dialogue 등 다양한 분야에서의 활용

Revisit To Transformer

Review What We talked about in Last Session



Revisit to Transformer

- Parallel Processing
- Tokenization
- Positional Embedding
- Three types of Attentions: Multi-head Self-Attention, Masked Multi-head Self-Attention, Cross Attention
- Layer Normalization
- Residual Connection

Reformer: The Efficient Transformer, 2020

https://arxiv.org/abs/2001.04451

REVISIT TO TRANSFORMER

Introduction

Large Transformer models can be prohibitively costly, especially on long sequences. To improve the efficiency of Transformers, authors introduce two techniques, locality sensitive hashing (LSH) attention, reversible transformer

- Reversible layers, introduced in Gomez et al. (2017), enable storing a single copy of activations in the whole model, so the N factor disappears.

Keywords

Axial positional encoding, LSH Attention, Chunking Feed Forward Layer, Reversible Residual Layer

- Splitting activations inside feed-forward layers and processing them in chunks removes the d_{ff} factor and saves memory inside feed-forward layers.
- Approximate attention computation based on locality-sensitive hashing replaces the $O(L^2)$ factor in attention layers with $O(LlogL)$ and allows operating on long sequences.

Axial Positional Encoding

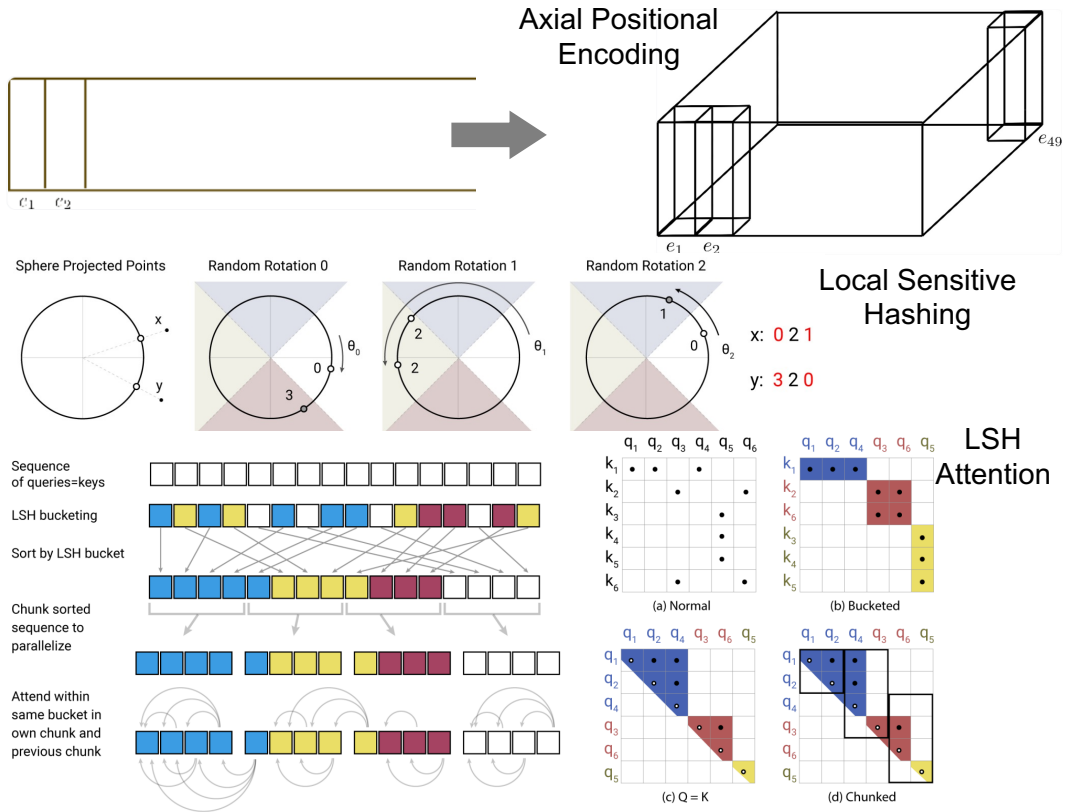
- Axial Positional Encodings are not explained in the official Reformer paper, but are extensively used in the official codebase
- The Reformer authors managed to drastically shrink the positional encodings in size by cutting the hidden size dimension in two and factorizing the dimension.

LSH Attention

- A shared QK, Query = Key, Using the same linear layer to go from A to Q and K, and a separate one for V. Sharing QK does not affect the performance of Transformer (Note. Focus on the keys in K that are closest q_i)
- Finding **nearest neighbors** quickly in high-dimensional spaces can be solved by local sensitive hashing (LSH). In this paper, the hashing scheme is that nearby vectors get the same hash with high probability and that hash-buckets are of similar size with high probability.
- Simplified depiction of LSH Attention showing the hash-bucketing, sorting, and chunking sorted sequence to parallelize and the resulting causal attentions.

Reversible Transformer

- Reversible Residual networks (RevNet) were introduced by Gomez et al. (2017). The main idea is to allow the activations at any given layer to be recovered from the activations at the following layer using the model parameters
- Reversible Transformer is applying the RevNet idea to the Transformer by combining the attention and feed-forward layers inside the RevNet block.
- Computations in feed-forward layers are completely independent across positions in a sequence, so the computation can be split into c chunks



Code

- From the paper: https://github.com/google/trax/tree/master/trax/models/reformer

https://huggingface.co/blog/reformer

Longformer: The Long-Document Transformer, 2020

https://arxiv.org/abs/2004.05150

REVISIT TO TRANSFORMER

Introduction

Transformer-based models are unable to process long sequences due to their self-attention operation, which scales quadratically with the sequence length. Longformer's attention mechanism is a drop-in replacement for the standard self-attention and combines a local windowed attention with a task motivated global attention.

Keywords

Sliding Window Attention,
Dilated Sliding Window Attention, Global Attention,
Bits Per Character (BPC), Longformer-Encoder-Decoder

Sliding Window Attention

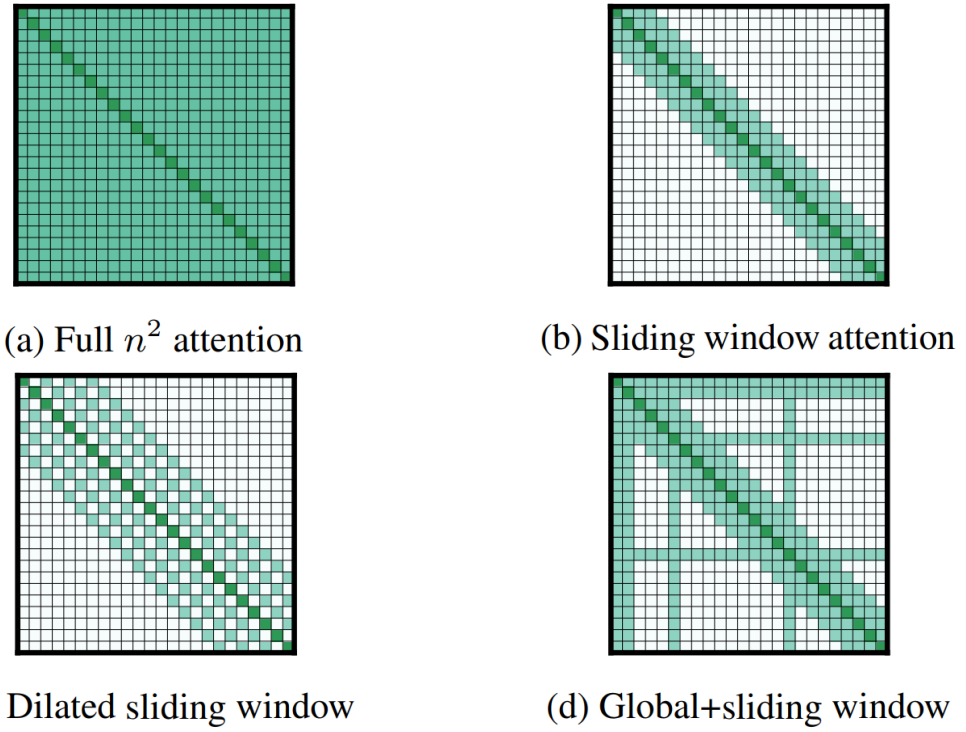
- A **fixed-size window** attention surrounding each token.
- Using multiple stacked layers of such windowed attention results in a large receptive field, where top layers have access to all input locations and have the capacity to build representations that incorporate information across the entire input, similar to CNNs

Dilated Sliding Window

- To further **increase the receptive field** without increasing computation, the sliding window can be "dilated"
- In multi-headed attention, settings with different dilation configurations per head improves performance by allowing some heads without dilation to focus on local context.

Global Attention

- The windowed and dilated attention are not flexible enough to learn task-specific representations.
- Accordingly, authors add "global attention" on few pre-selected input locations.
- For example, for classification global attention is used for the [CLS] token while in QA global attention is provided on all question tokens.



(a) Full n^2 attention (b) Sliding window attention

(c) Dilated sliding window (d) Global+sliding window

Results

- The resulting model can process sequences up to 4,096 tokens long.
- Longformer consistently outperforms the RoBERTa.

Model	*MLM BPC*	base	large
RoBERTa (seqlen: 512)		1.846	1.496
Longformer (seqlen: 4,096)		10.299	8.738
+ copy position embeddings		1.957	1.597
+ 2K gradient updates		1.753	1.414
+ 65K gradient updates		1.705	1.358
Longformer (train extra pos. embed. only)		1.850	1.504

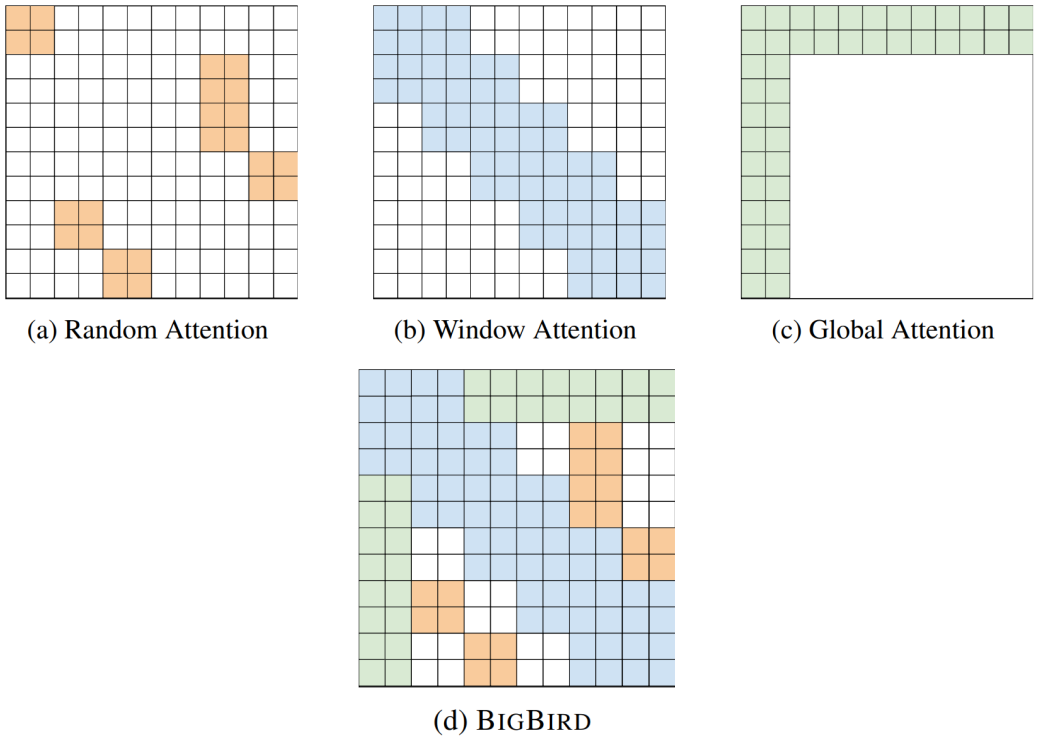
Code

- From the paper: https://github.com/allenai/longformer
- Convert RoBERTa to Longformer: https://github.com/allenai/longformer/blob/master/scripts/convert_model_to_long.ipynb

Big Bird: Transformers for Longer Sequences, 2020

https://arxiv.org/abs/2007.14062

REVISIT TO TRANSFORMER

Introduction	One of Transformers-based models core limitations is the **quadratic dependency** (mainly in terms of memory) on the sequence length due to their full attention mechanism. BIGBIRD, a sparse attention mechanism that reduces this quadratic dependency to linear.	***Keywords***	Sparse attention mechanism, Sparse random graph, Random Block, Local Window, Global Tokens
Key Questions	• Can we achieve the empirical benefits of a fully quadratic self-attention scheme using fewer inner-products? • Do these sparse attention mechanisms preserve the expressivity and flexibility of the original network?		
Model Concept	• BIGBIRD model uses the generalized attention mechanism, which is described by a **directed graph** whose vertex set is [n] and the set of arcs (directed edges) represent the set of inner products that the attention mechanism will consider. • The problem of reducing the quadratic complexity of self-attention can now be seen as a **graph sparsification problem**. Sparse random graph for attention mechanism should have two desiderata: small average path length between nodes and a notion of locality.		 (a) Random Attention (b) Window Attention (c) Global Attention (d) BIGBIRD
Random Block	• Each query block attends to r random key blocks. • All tokens attending to a set of r random tokens.		
Local Window	• While creating the block, we ensure that the number of query blocks and the number of key blocks are the same. This helps defining the block window attention. All tokens attending to a set of w local neighboring tokens.		
Global Tokens	• Global tokens attending on all parts of the sequence.		
Results	• The proposed sparse attention can handle sequences of length up to 8x of what was previously possible using similar hardware. • Max sequence length: 4096	***Code***	• From the paper: https://github.com/google-research/bigbird

Rethinking Attention with Performers, 2020

https://arxiv.org/abs/2009.14794

REVISIT TO TRANSFORMER

Introduction

Transformers are powerful neural network architectures. Regular Transformer **scales quadratically** with the number of tokens L in the input sequence, which is expensive for large L and precludes its usage with limited computational resources even for moderate values of L. Performers, capable of provably accurate and practical **estimation of regular (softmax) full-rank attention**, but of only linear space and time complexity and not relying on any priors such as sparsity or low-rankness.

Keywords

Kernelizable attention mechanisms,
Fast Attention Via positive Orthogonal Random features mechanism (FAVOR+)

Performer Attention Mechanism

- A new method for estimating softmax (and Gaussian) kernels with positive orthogonal random features which FAVOR+ leverages for the robust and unbiased **estimation of regular (softmax) attention**
- The **softmax-kernel** which defines regular attention matrix A is $\mathrm{SM}(\mathbf{x},\mathbf{y}) \stackrel{\mathrm{def}}{=} \exp(\mathbf{x}^\top \mathbf{y})$
- Positive Random Features for Softmax

$$\mathrm{SM}(\mathbf{x},\mathbf{y}) = \mathbb{E}_{\omega\sim\mathcal{N}(0,\mathbf{I}_d)}\left[\exp\left(\omega^\top \mathbf{x} - \frac{\|\mathbf{x}\|^2}{2}\right)\exp\left(\omega^\top \mathbf{y} - \frac{\|\mathbf{y}\|^2}{2}\right)\right] = \Lambda \mathbb{E}_{\omega\sim\mathcal{N}(0,\mathbf{I}_d)} \cosh(\omega^\top \mathbf{z})$$

Experiments

- A Performer replaces only the attention component with our method, while all other components are the same as for the regular Transformer
- Compared speed-wise the backward pass of the Transformer and the Performer.
- In terms of L, the Performer reaches nearly linear time and sub-quadratic memory consumption
- The Performer achieves nearly optimal speedup and memory efficiency possible.
- The combination of both memory and backward pass efficiencies for large L allows respectively, large batch training and lower wall clock time per gradient step

[Regular Attention Matrix]

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V}, \quad \mathbf{A} = \exp(\mathbf{Q}\mathbf{K}^\top/\sqrt{d}), \quad \mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_L)$$

[Kernel K, and Approximate attention]

$$\mathrm{K}(\mathbf{x},\mathbf{y}) = \mathbb{E}[\phi(\mathbf{x})^\top \phi(\mathbf{y})]$$

$$\widehat{\mathrm{Att}}_{\leftrightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \widehat{\mathbf{D}}^{-1}(\mathbf{Q}'((\mathbf{K}')^\top \mathbf{V})), \qquad \widehat{\mathbf{D}} = \mathrm{diag}(\mathbf{Q}'((\mathbf{K}')^\top \mathbf{1}_L))$$

[Approximation of the Regular Attention Mechanism]

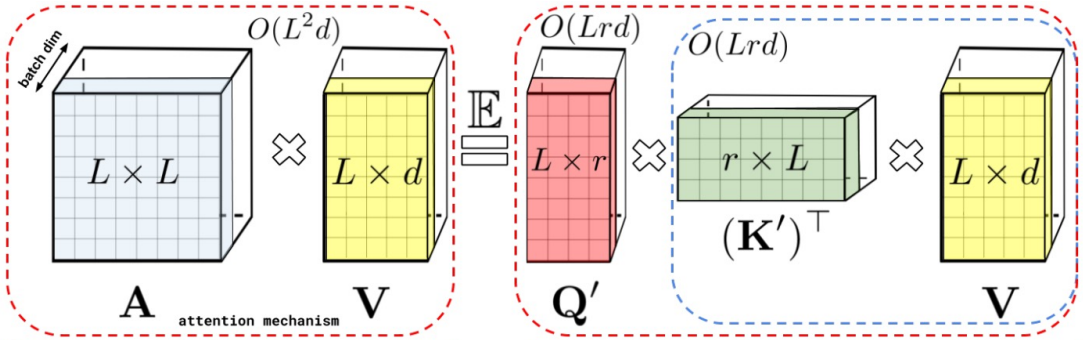


Figure 1: Approximation of the regular attention mechanism $\mathbf{AV}$ (before $\mathbf{D}^{-1}$-renormalization) via (random) feature maps. Dashed-blocks indicate order of computation with corresponding time complexities attached.

Broad Impact

- The Areas where Performer can be impactful: biological sequence analysis, lower energy consumption, research on efficient Transformers architectures

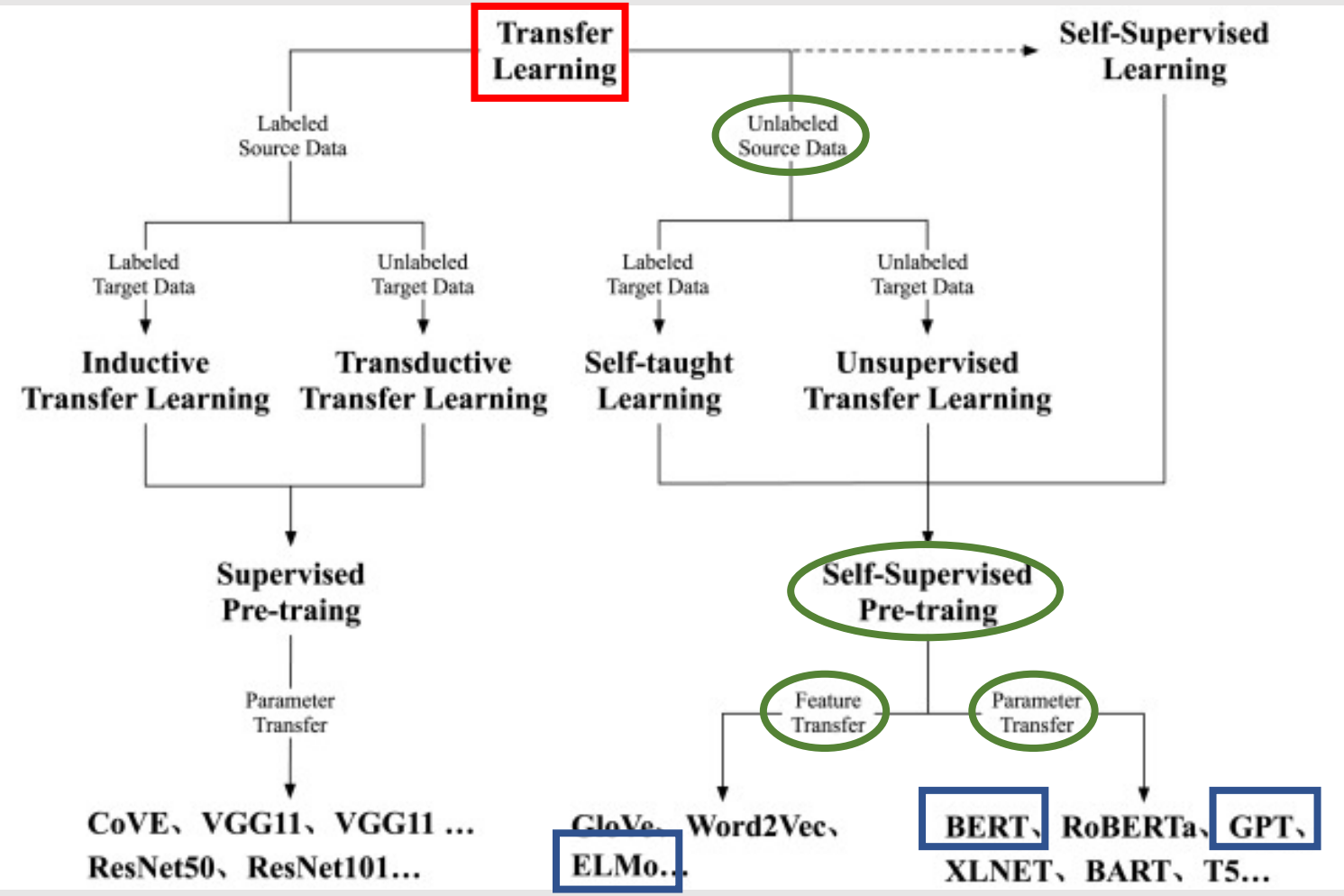
Code

- From the paper: https://github.com/google-research/google-research/tree/master/performer
- Port HuggingFace Weight to Performer: https://github.com/tenexcoder/huggingface-tutorials/tree/main/performer

BackGround: Transfer Learning

Transfer Learning and and Self-Supervised Learning

Background(Learning Methods): **Transfer Learning**



In NLP, we trained on a general language modeling (LM) task and then fine tuned on text classification (or other task). This would, in principle, perform well because the model would be able to use its **knowledge of the language semantics** acquired from the generative pre-training.

- capture **long-term dependencies** in language
- effectively incorporates **hierarchical relations**
- help the model learn **sentiments**
- **large data corpus** is easily available for LM

Revisit To Language Model(언어모델)

- 언어모델링: Assign a Probability to a sentence → Predict Next Word
  - 확률기반 언어모델: N-gram based - Unigram, Bigram, Trigram…

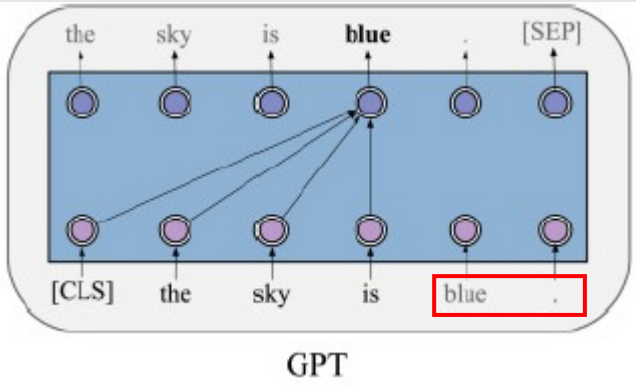
$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

  - Neural Network기반 언어모델: Word Embedding(Static)- Word2vec, Glove, Fasttext…
  - Transformer 기반 언어모델(Dynamic): BERT, GPT…

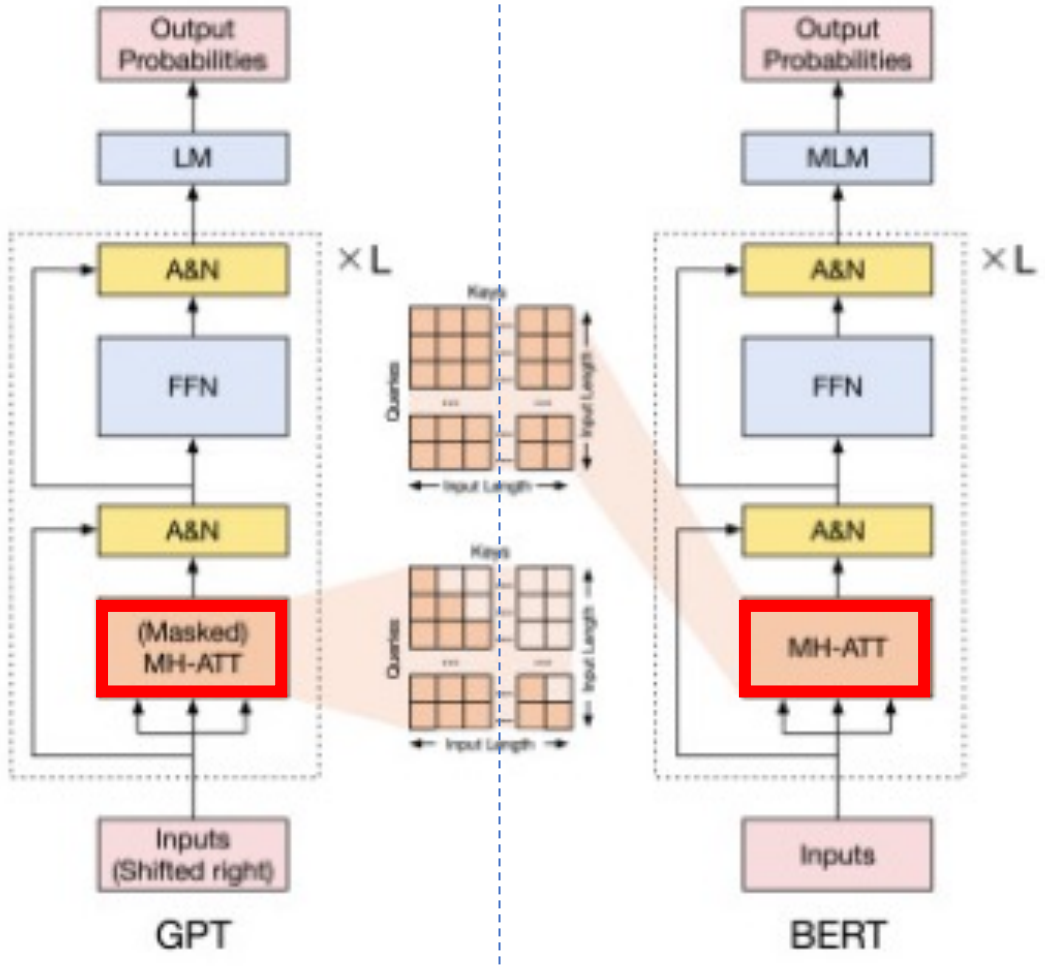
GPT

Generative Pre-trained Transformer

- Given large-scale corpora without labels, GPT optimizes a standard **autoregressive language modeling**, that is, maximizing the conditional probabilities of all the words by taking their previous words as contexts.
- The adaptation procedure of GPT to specific tasks is fine-tuning, by using the pre-trained parameters of GPT as a start point of downstream tasks.



GPT

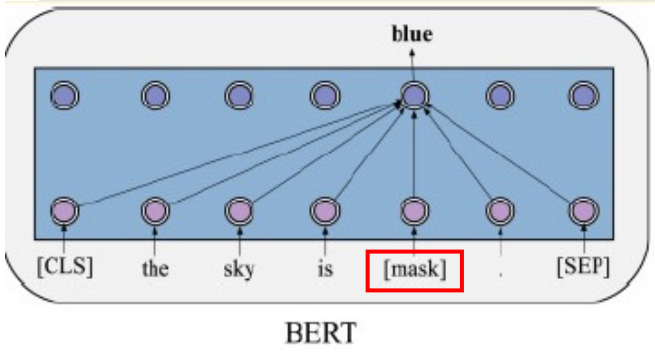


GPT BERT

BERT

Bidirectional Encoder Representations from **Transformers**

- In the pre-training phase, BERT applies **autoencoding language modeling** rather than autoregressive language modeling used in GPT. More specifically, inspired by cloze (Taylor, 1953), the objective masked language modeling (MLM) is designed.
- By modifying inputs and outputs with the data of downstream tasks, BERT could be fine-tuned for any NLP tasks.



BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018

https://arxiv.org/abs/1810.04805

자연어처리의 최신동향-3:
Transformer기반의 사전학습모델들

Motivation

There are two existing strategies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The major limitation is that standard LMs are unidirectional, and this limits the choice of architectures that can be used during pre-training. Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks.

Keywords

BERT (Bidirectional Encoder Representations from Transformer) ,
Autoencoding LM,
Masked Language Model (MLM), Next Sentence Prediction (NSP),
WordPiece embeddings

Introduction

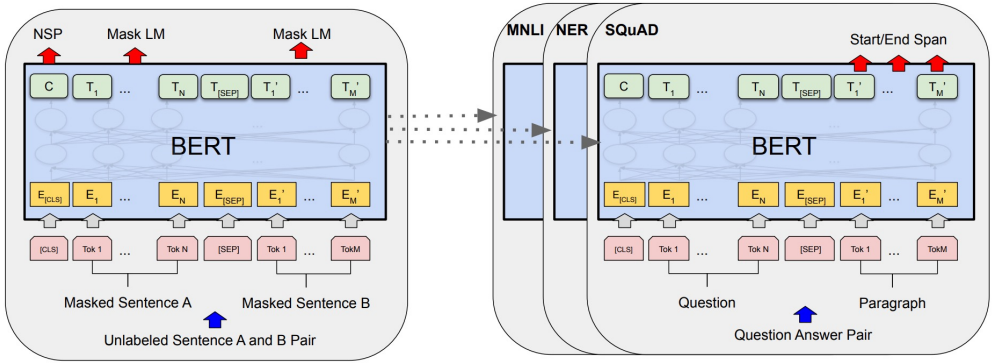
- BERT is designed to pre-train **deep bidirectional representations** from **unlabeled text** by jointly conditioning on both left and right context in all layers.
- Using a "**masked language model**" (MLM) pre-training objective and a "**next sentence prediction**" (NSP) task that jointly pre-trains text-pair representations.

Model Architecture

- A multi-layer bidirectional encoder from Transformer
- Base model: Layer L=12, Hidden size H=768, Attention Heads A=12, Total Parameters=110M
- Big model: Layers=24, Hidden size H=1024, Attention Heads A=16, Total Parameters=340M

Overall Procedures

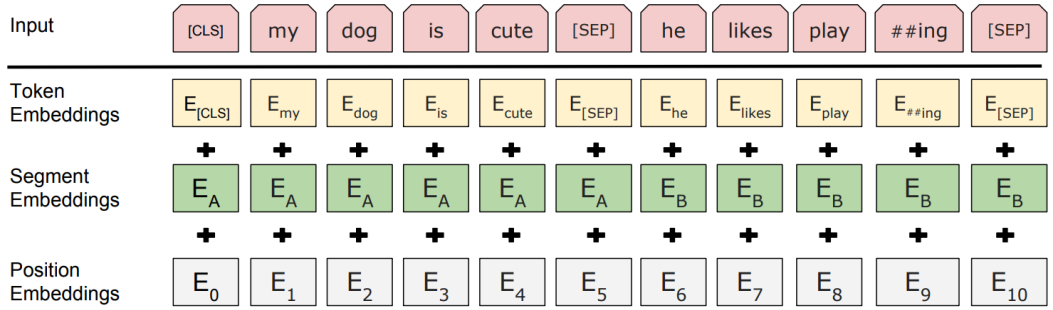
- There are two steps in the framework: pre-training and fine-tuning.
- Pre-training: the model is trained on unlabeled data over different pre-training tasks.
- Fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks.



Pre-training Fine-Tuning

Input/Output Representation

- A single sentence and a pair of sentences in one token sequence
  A **"sentence"** can be an arbitrary span of contiguous text, rather than an actual linguistic sentence.
- A **"sequence"** refers to the input token sequence to BERT
- Using **WordPiece embeddings** with a 30,000 token vocabulary
- A special classification token ([CLS]): the first token of every sequence
- A **special token** ([SEP]): Separating sentences
- Adding an embedding indicating whether it belongs to sentence A or B



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018

https://arxiv.org/abs/1810.04805

자연어처리의 최신동향-3:
Transformer기반의 사전학습모델들

Masked LM

- Masking some percentage of the input tokens at random, and then predict those masked Tokens
- Masking 15% of all WordPiece tokens in each sequence at random
- There is a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning.
- To mitigate this, replacing the i-th token (15% of tokens) with (1) the [MASK] token 80%, (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time.

Next Sentence Prediction

- When choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext).

Results

- GLUE Test results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
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	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

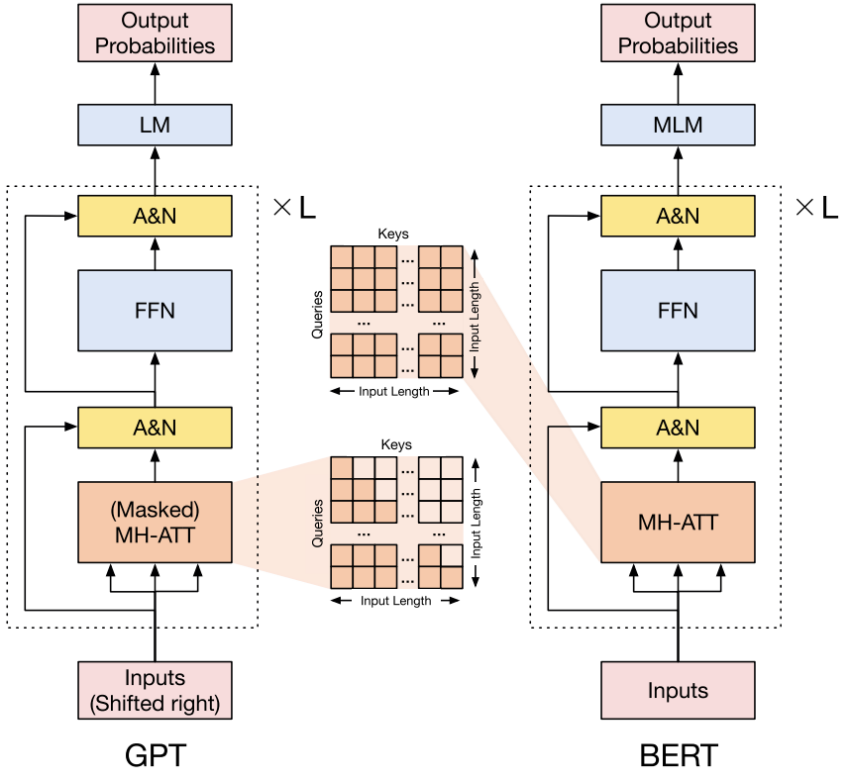
- Ablation Studies

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$BERT_{BASE}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Pre-training Data

- The BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words)
- For Wikipedia extracting only the text passages and ignore lists, tables, and headers
- Use a document-level corpus rather than a shuffled sentence-level corpus

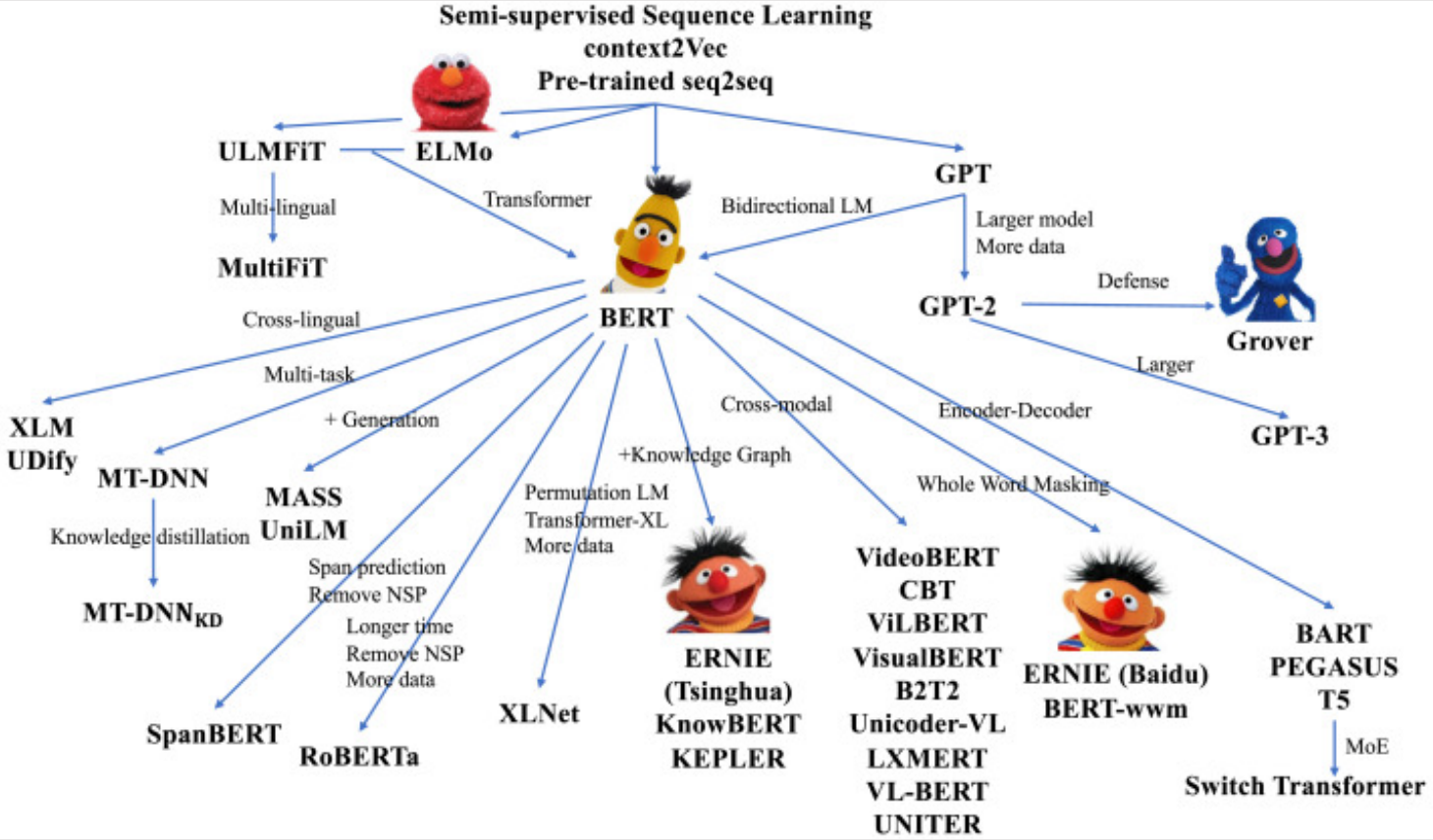
BERT vs. GPT



Code

- From the paper: https://github.com/ google-research/bert

After GPT and BERT



- **RoBERTa**(Liu et al., 2020d) is one of the success variants of BERT, which mainly has four simple and effective changes: (1) Removing the NSP task; (2) More training steps, with bigger batch size and more data; (3) Longer training sentences; (4) Dynamically changing the pattern.
- Some work improves the model **architectures** and explores novel **pre-training tasks**, such as XLNet (Yang et al., 2019), UniLM (Dong et al., 2019), MASS (Song et al., 2019), SpanBERT (Joshi et al., 2020) and ELECTRA (Clark et al., 2020).
- Besides, incorporating rich data sources is also an important direction, such as utilizing multilingual corpora, knowledge graphs, and images.
- Since the model scale is a crucial success factor of PTMs, researchers also explore to build larger models to reach over hundreds of billions of parameters, such as the series of GPT (Radford et al., 2019; Brown et al., 2020), Switch Transformer (Fedus et al.,2021), and meanwhile conduct computational efficiency optimization for training PTMs(Shoeybi et al., 2019; Rajbhandari et al., 2020; Ren et al., 2021).

Pre-trained Models: Past, Present, and Future

Based on the paper Han Xu et al. (2021), https://www.sciencedirect.com/science/article/pii/S2666651021000231

4 Important Directions of PTMs	The details of various latest PTMs	PTMs
Designing Effective Architecture	Combining Autoregressive and Autoencoding Modeling	XLNet, GLM
	Applying Generalized Encoder-Decoder	MASS, BART, PEGASUS, T5
	Cognitive-Inspired Architectures	LAMA,REALM
	More Variants of Existing PTMs: Masking Strategy	SpanBERT
Utilizing Multi-Source Data	Multilingual Pre-Training	Unicoder
	Multimodal Pre-training	ViLBERT, Unicoder-VL, CLIP, DALL-E, CogView, codeGen
	Knowledge-Enhanced Pre-Training	KEPLER, ERNIE, KnowBERT, KGLM, Commonsense story generation
Improving Computational Efficiency	System-Level Optimization	Megatron-LM
	Efficient Pre-Training	Train no Evil, Progressive layer dropout
	Model Compression	ALBERT, Compressing BERT, DistillBERT, MINILM
Interpretation and Theoretical Analysis	Knowledge of PTMs: Linguistic Knowledge	Word/contextual representation, BERT structure
	Knowledge of PTMs: World Knowledge	Evaluate commonsense
	Robustness of PTMs	Trick me if you can, BERT's robustness
	Structural Sparsity of PTMs	BERT's structure sparsity
	Theoretical Analysis of PTMs	Theoretical Analysis of Unsupervised PTM

01 Designing Effective Architecture

Improve the model architectures and explore novel pre-training tasks

Design effective architecture

Combing autoregressive and autoencoding modeling

- Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context
- XLNet: Generalized Autoregressive Pretraining for Language Understanding
- RETHINKING ATTENTION WITH PERFORMERS
- GLM: All NLP Tasks Are Generation Tasks: A General Pretraining Framework

Applying generalized encoder-decoder

- MASS: Masked Sequence to Sequence Pre-training for Language Generation
- T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer
- BART:Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
- PEGASUS: Pre-training with Extracted Gap sentences for Abstractive Summarization

Cognitive-inspired architectures

- Language Models as Knowledge Bases?
- REALM: Retrieval-Augmented Language Model Pre-Training

More variants of existing PTMs: masking strategy

- SpanBERT: Improving Pre-training by Representing and Predicting Spans

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, 2019

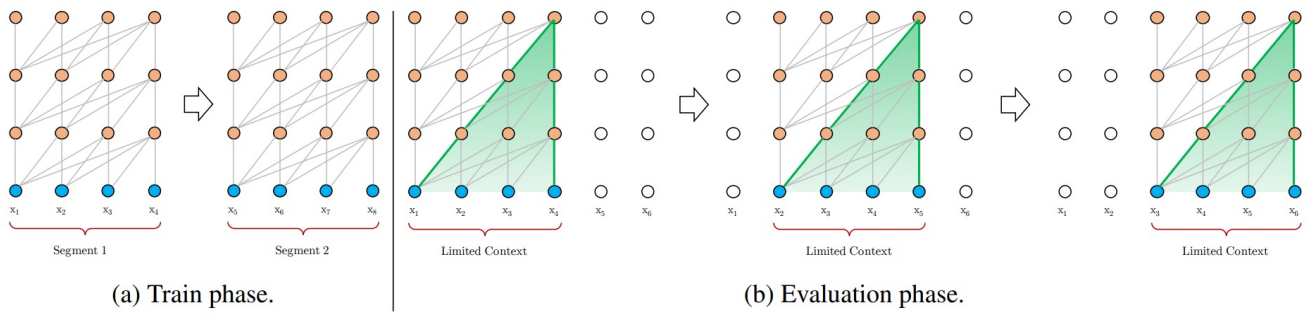
https://arxiv.org/abs/1901.02860

자연어처리의 최신동향-3:
Transformer기반의 사전학습모델들

Motivation	Transformers have a potential of learning longer-term dependency, but are **limited by a fixed-length context** in the setting of language Modeling As a consequence of the fixed context length, the model cannot capture any longer-term dependency beyond the predefined context length. (context fragmentation)	***Keywords***	Transformer-XL, Segment-level recurrence mechanism, Relative positional encoding,

Introduction

- Transformer-XL that enables learning dependency beyond a fixed length without disrupting temporal coherence
- **Segment-level recurrence mechanism**: Reusing the hidden states obtained in previous segments. The reused hidden states serve as memory for the current segment, which builds up a recurrent connection between the segments. As a result, modeling very long-term dependency becomes possible because information can be propagated through the recurrent connections.
- **Relative positional encoding**: formulation that generalizes to attention lengths longer than the one observed during training

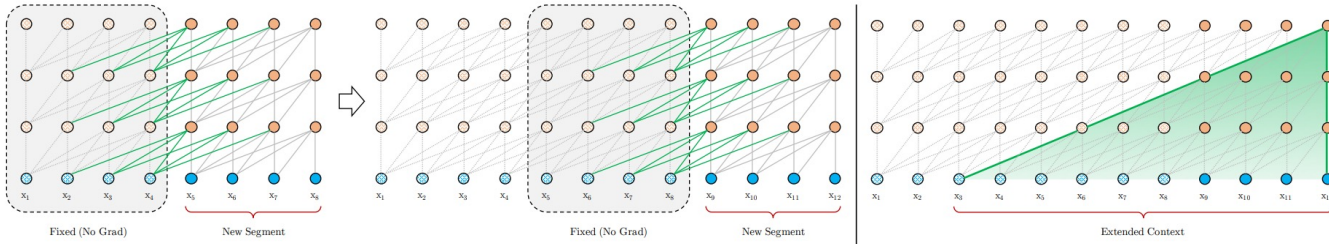


(a) Train phase. (b) Evaluation phase.

Figure 1: Illustration of the vanilla model with a segment length 4.

Segment-level Recurrence Mechanism

- With this recurrence mechanism applied to every two consecutive segments of a corpus, it essentially creates a segment-level recurrence in the hidden states.
- During training, the hidden state sequence computed for the previous segment is fixed and cached to be reused as an extended context when the model processes the next new segment.
- The concatenation of two hidden sequences along the length dimension.
- Consequently, the largest possible dependency length grows linearly w.r.t. the number of layers as well as the segment length.
- Besides achieving extra long context and resolving fragmentation, another benefit that comes with the recurrence scheme is significantly faster evaluation.



(a) Training phase. (b) Evaluation phase.

Figure 2: Illustration of the Transformer-XL model with a segment length 4.

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, 2019

https://arxiv.org/abs/1901.02860

자연어처리의 최신동향-3:
Transformer기반의 사전학습모델들

Relative Positional Encodings

- How can we keep the positional information coherent when we reuse the states?
- In the standard Transformer (Vaswani et al., 2017), the attention score between query q_i and key vector k_j within the same segment can be decomposed as
- Replace all appearances of the absolute positional embedding U_j for computing key vectors in term (b) and (d) with its relative counterpart R_{i-j} .
- Introduce a trainable parameter u, v ∈ R^d to replace the query in term (c), (d).

$$\mathbf{A}^{\text{abs}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.$$

[Standard Transformer]

$$\mathbf{A}^{\text{rel}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.$$

[Transformer-XL]

- Term (a) represents content-based addressing, (b) captures a content-dependent positional bias, (c) governs a global content bias, and (d) encodes a global positional bias.

Model

- The computational procedure for a N-layer Transformer-XL with a single attention head here.

$$\widetilde{\mathbf{h}}_\tau^{n-1} = \left[\text{SG}(\mathbf{m}_\tau^{n-1}) \circ \mathbf{h}_\tau^{n-1}\right]$$

$$\mathbf{q}_\tau^n, \mathbf{k}_\tau^n, \mathbf{v}_\tau^n = \mathbf{h}_\tau^{n-1}\mathbf{W}_q^{n\top}, \widetilde{\mathbf{h}}_\tau^{n-1}\mathbf{W}_{k,E}^{n\top}, \widetilde{\mathbf{h}}_\tau^{n-1}\mathbf{W}_v^{n\top}$$

$$\mathbf{A}^n_{\tau,i,j} = \mathbf{q}^{n}_{\tau,i}{}^\top \mathbf{k}^n_{\tau,j} + \mathbf{q}^{n}_{\tau,i}{}^\top \mathbf{W}^n_{k,R}\mathbf{R}_{i-j} + u^\top \mathbf{k}_{\tau,j} + v^\top \mathbf{W}^n_{k,R}\mathbf{R}_{i-j}$$

$$\mathbf{a}_\tau^n = \text{Masked-Softmax}(\mathbf{A}_\tau^n)\mathbf{v}_\tau^n$$
$$\mathbf{o}_\tau^n = \text{LayerNorm}(\text{Linear}(\mathbf{a}_\tau^n) + \mathbf{h}_\tau^{n-1})$$
$$\mathbf{h}_\tau^n = \text{Positionwise-Feed-Forward}(\mathbf{o}_\tau^n)$$

Results

- improve the state-of-the-art results of bpc/perplexity to 0.99 on enwiki8, 1.08 on text8, 18.3 on WikiText-103, 21.8 on One Billion Word, and 54.5 on Penn Treebank (without finetuning).
- Applying Transformer-XL to a variety of datasets on both word-level and character-level language modeling

Model	#Param	PPL
Grave et al. (2016b) - LSTM	-	48.7
Bai et al. (2018) - TCN	-	45.2
Dauphin et al. (2016) - GCNN-8	-	44.9
Grave et al. (2016b) - LSTM + Neural cache	-	40.8
Dauphin et al. (2016) - GCNN-14	-	37.2
Merity et al. (2018) - QRNN	151M	33.0
Rae et al. (2018) - Hebbian + Cache	-	29.9
Ours - Transformer-XL Standard	151M	**24.0**
Baevski and Auli (2018) - Adaptive Input◇	247M	20.5
Ours - Transformer-XL Large	257M	**18.3**

Table 1: Comparison with state-of-the-art results on WikiText-103. ◇ indicates contemporary work.

Model	#Param	bpc
Cooijmans et al. (2016) - BN-LSTM	-	1.36
Chung et al. (2016) - LN HM-LSTM	35M	1.29
Zilly et al. (2016) - RHN	45M	1.27
Krause et al. (2016) - Large mLSTM	45M	1.27
Al-Rfou et al. (2018) - 12L Transformer	44M	1.18
Al-Rfou et al. (2018) - 64L Transformer	235M	1.13
Ours - 24L Transformer-XL	277M	**1.08**

Table 3: Comparison with state-of-the-art results on text8.

Method	PPL
Ours	**25.2**
With Shaw et al. (2018) encodings	25.7
Without recurrence	27.1

Table 7: Ablation study on One Billion Word, a dataset without long-term dependency.

Code

- From the paper: https://github.com/kimiyoung/ transformer-xl

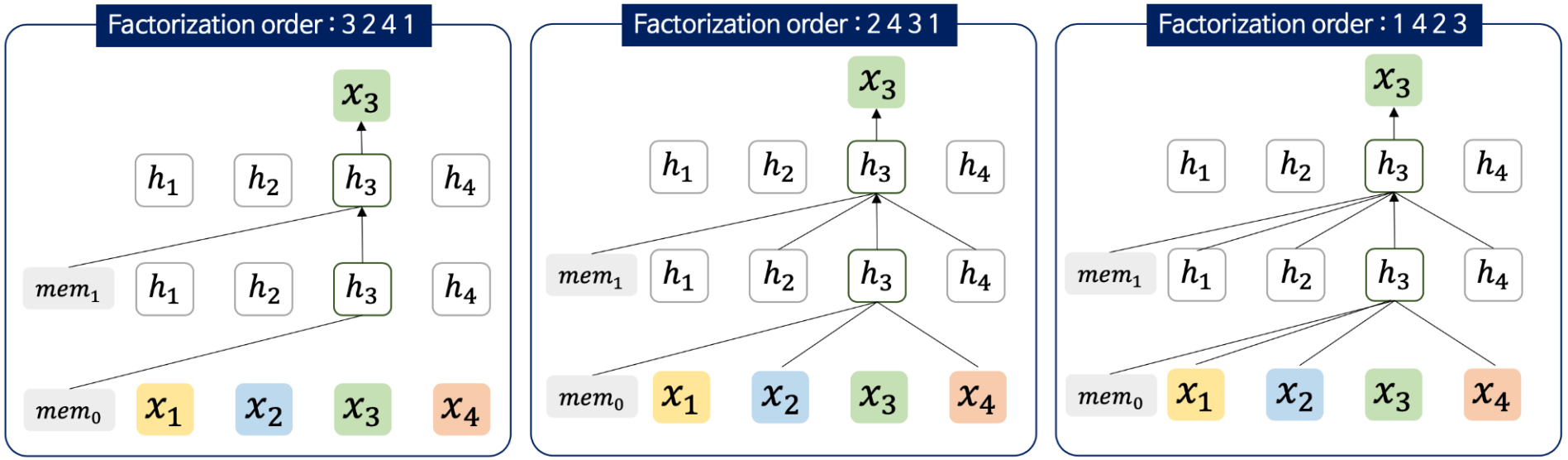
XLNET(Generalized Autoregressive Pretraining for Language Understanding

1. Permutation Language Modeling Object

1. Permutation Language Modeling Objective

Input sequence (T) x_1 x_2 x_3 x_4

모든 permutation 집합 (Z_T) [[3,2,4,1], [2,4,3,1], [1,4,2,3] … [4,3,1,2]]



XLNET(Generalized Autoregressive Pretraining for Language Understanding

2. Target-Aware Representation for Transformer

2. Target-Aware Representation for Transformer

- 새로운 objective function는 standard Transformer에서 작동하지 않음
- 따라서 Transformer에 XLNet의 Objective function을 적용하기위해 **Target-Aware Representation** 제안

$$p_\theta(x_{z_t} \mid x_{z<t}) = \frac{exp(e(x)^T h_\theta(x_{z<t})}{\Sigma_{x'} exp(e(x')^T h_\theta(x_{z<t}))}$$

XLNet의 AR model

x_1 x_2 x_3 ?

Permutation을 통해 ? 예측

학습시 Permutation하여
예측할 token이 명확하지 않음

Ex	[1,2,3,4]	$p(x_3 \mid x_1, x_2)$	$h_\theta(x_1, x_2)$
	[1,2,4,3]	$p(x_4 \mid x_1, x_2)$	$h_\theta(x_1, x_2)$

동일한 representation으로 다른 target 맞춰야 함

⬇

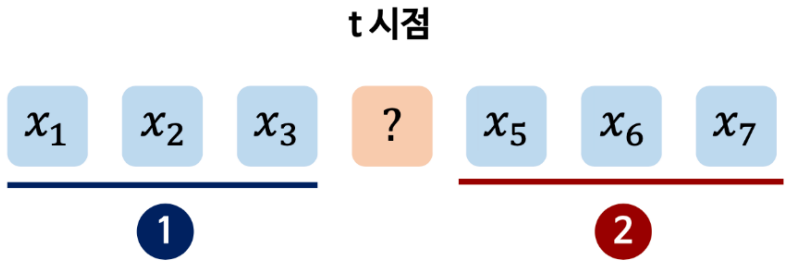
Target Position-Aware Representation

$$h_\theta(x_{z<t}) \rightarrow g_\theta(x_{z<t}, z_t)$$

XLNET(Generalized Autoregressive Pretraining for Language Understanding

3. Two-Stream Self-Attention

3. Two-Stream Self-Attention



1. T 시점에서 Target token을 예측하기위해 $g(x_{z<t}, z_t)$ 는 T 시점 이전의 context와 target position을 이용해야 함
2. T 시점 이후의 token을 예측하기위해 $g(x_{z<t}, z_t)$ T 시점의 context도 가지고 있어야함

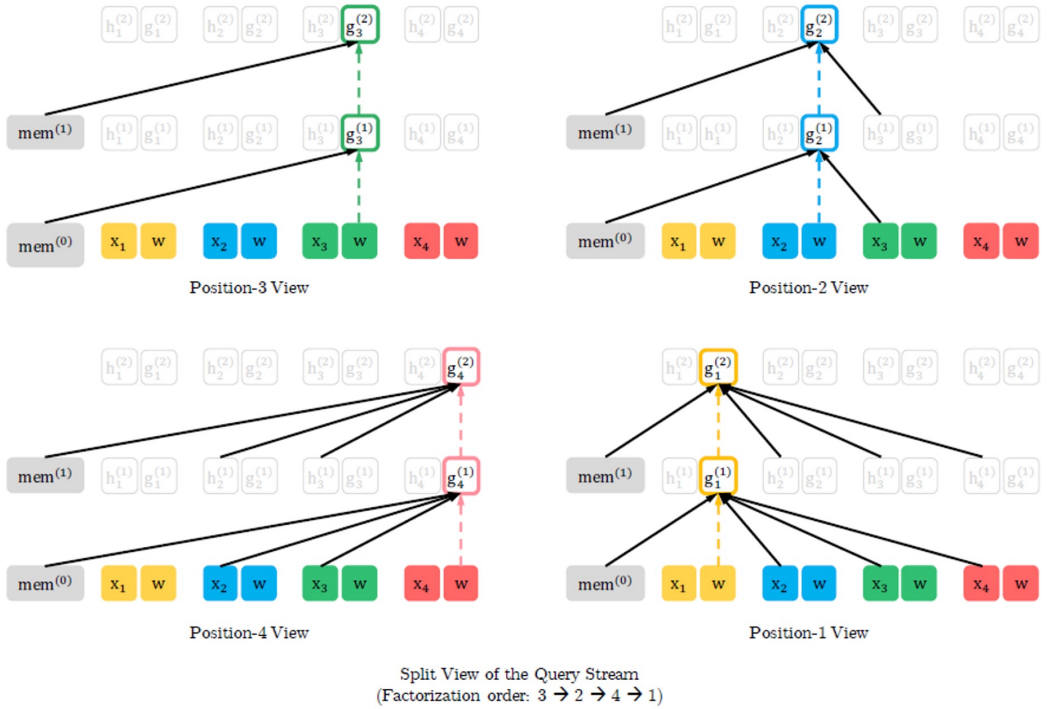
⬇

1 2 모두 고려하도록 2가지 hidden representation 사용

2개의 hidden representation 사용할 수 있는 transformer 구조 제안

XLNET(Generalized Autoregressive Pretraining for Language Understanding

- Query representation

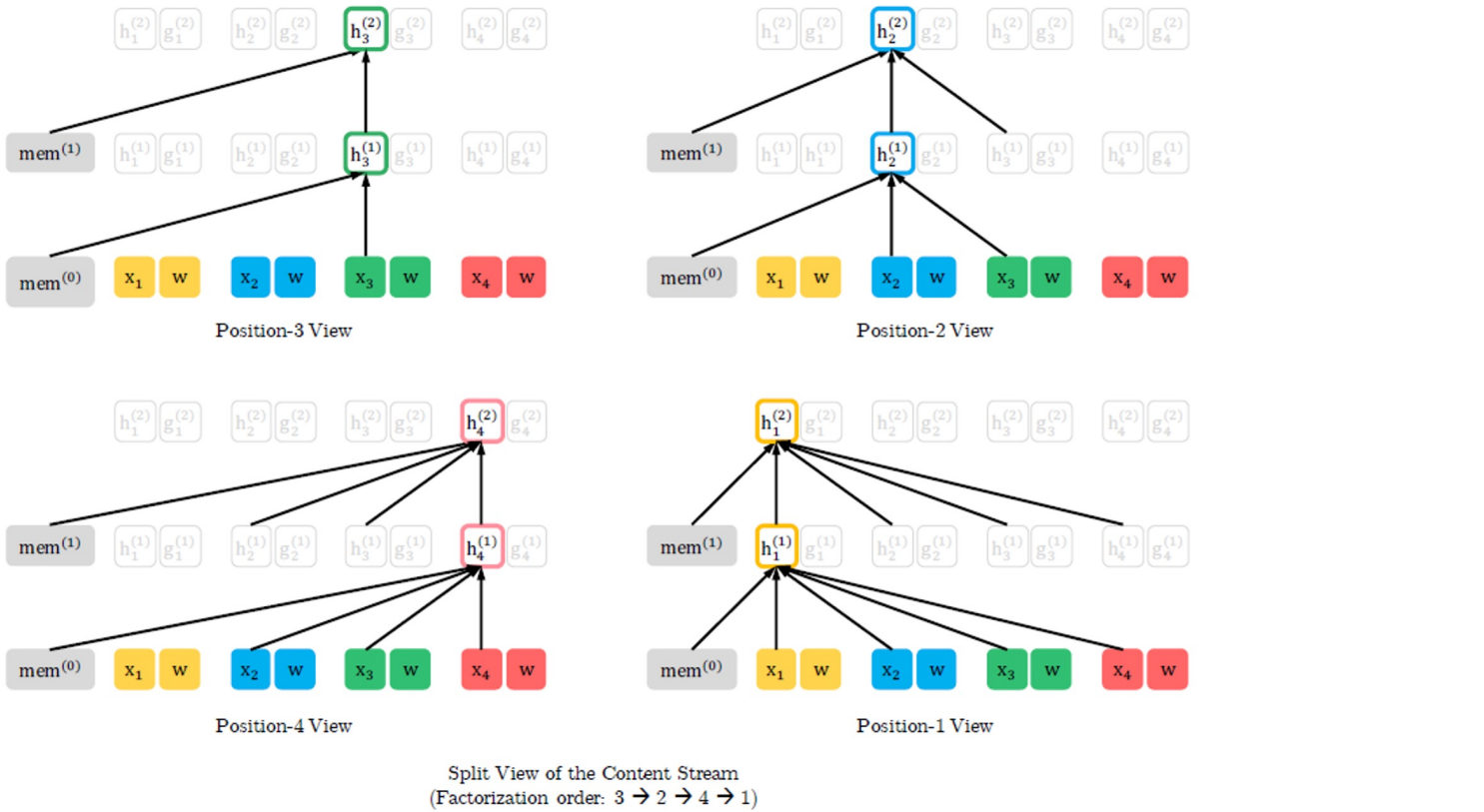


Split View of the Query Stream
(Factorization order: 3 → 2 → 4 → 1)

$$Query\ Stream: g_{z_t}^{(m)} \leftarrow Attention(Q = g_{z_t}^{(m-1)}, KV = h_{z_{<t}}^{(m-1)}; \theta)$$

XLNET(Generalized Autoregressive Pretraining for Language Understanding

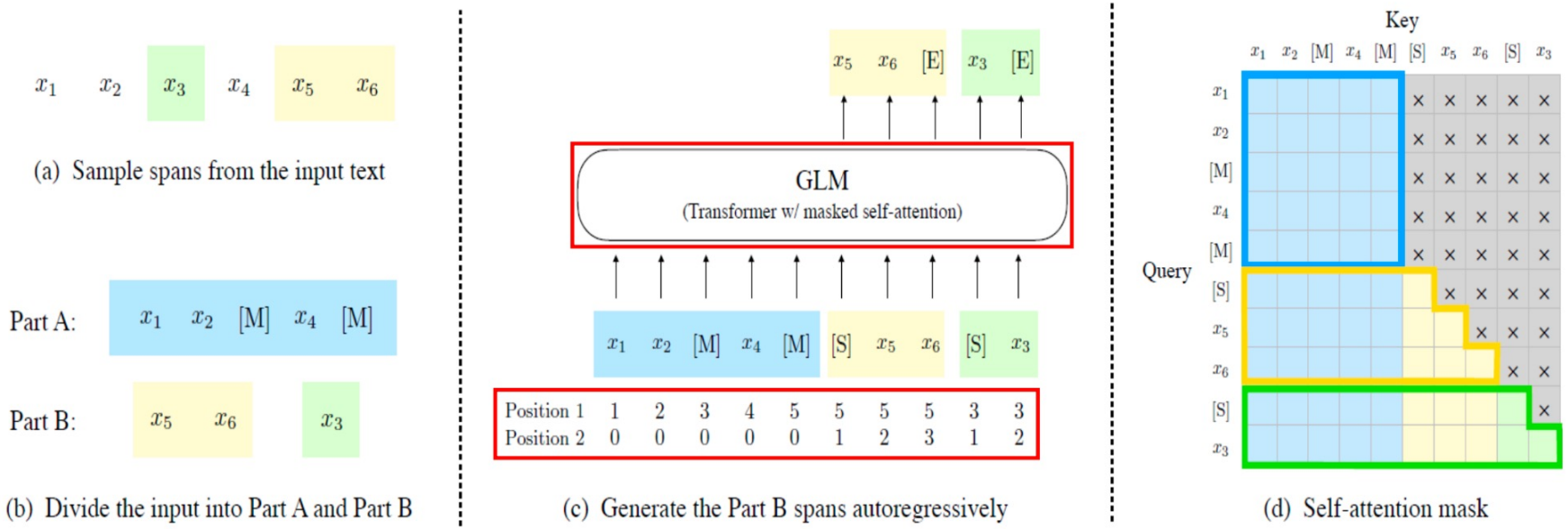
- Content representation



$$Content\ Stream: h_{z_t}^{(m)} \leftarrow Attention(Q = h_{z_t}^{(m-1)}, KV = h_{z \le t}^{(m-1)}; \theta)$$

GLM (All NLP Tasks Are Generation Tasks: A General Pre-training Framework)

GLM - Framework



- (c): Given a **variable-length** masked span, GLM autoregressively generates the masked tokens
- (d): Combining **autoencoding and autoregressive** model

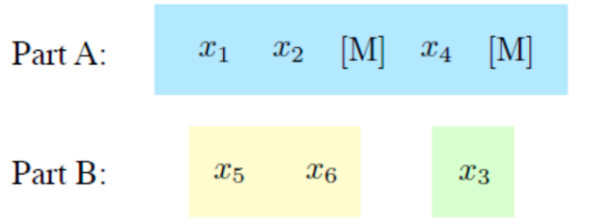
GLM (All NLP Tasks Are Generation Tasks: A General Pre-traing Framework)

2D Positional Encoding

- Preserve information of [MASK]'s number
  => 2D Positional Encoding

- Position 1: inter-span position
- Position 2: intra-span position

=> breaks **independent assumption**



(b) Divide the input into Part A and Part B

	x_1	x_2	[M]	x_4	[M]	[S]	x_5	x_6	[S]	x_3
Position 1	1	2	3	4	5	5	5	5	3	3
Position 2	0	0	0	0	0	1	2	3	1	2

GLM (All NLP Tasks Are Generation Tasks: A General Pre-training Framework)

Finetuning

- Different according to the tasks:
  => inconsistency between pre-training(MLM) and finetuning
- Following PET(Pattern Exploit Training),

<PVP(Pattern-Verbalizer Pair) examples>

Question & Answering

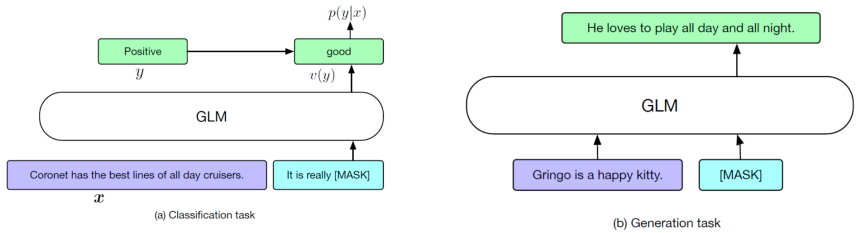
Where is the capital of Korea? → The capital of Korea is <MASK>

Sentiment Classification

This movie is terrible. → This movie is terrible. So I feel <MASK>.

Finetune

- All downstream tasks => generation tasks of blank filling
- Resolve **inconsistency between pretraining and finetuning**



MASS (Masked Sequence to Sequence Pre-training for Language Generation)

- Proposed architecture "MASS"
  - Sequence to sequence learning framework
  - Designed to pre-train the encoder and decoder jointly in two steps
    (1) Predicting the fragment of the sentence that is masked on the encoder side
    (2) Masking the input tokens of the decoder that are unmasked in the source side

- MASS just needs to pre-train one model and then fine-tune on a variety of downstream tasks

- 3.1 Sequence to Sequence Learning
  - Sentence pair notation
    - $(x, y) \in (X, Y)$
      where $x = (x_1, x_2, \dots, x_m)$ *is the source sentence with m tokens*
      $y = (y_1, y_2, \dots, y_n)$ *is the target sentence with n tokens,*
      X, Y *are the source and target domains*
  - Objective function
    - $L(\theta; (X, Y)) = \Sigma_{(x,y)\in(X,Y)} log\, P(y|x; \theta)$
    - $P(y|x; \theta) = \prod_{t=1}^{n} P(y_t|y_{<t}, x; \theta)$,
      where $y_{<t}$ is the preceeding tokens before position t

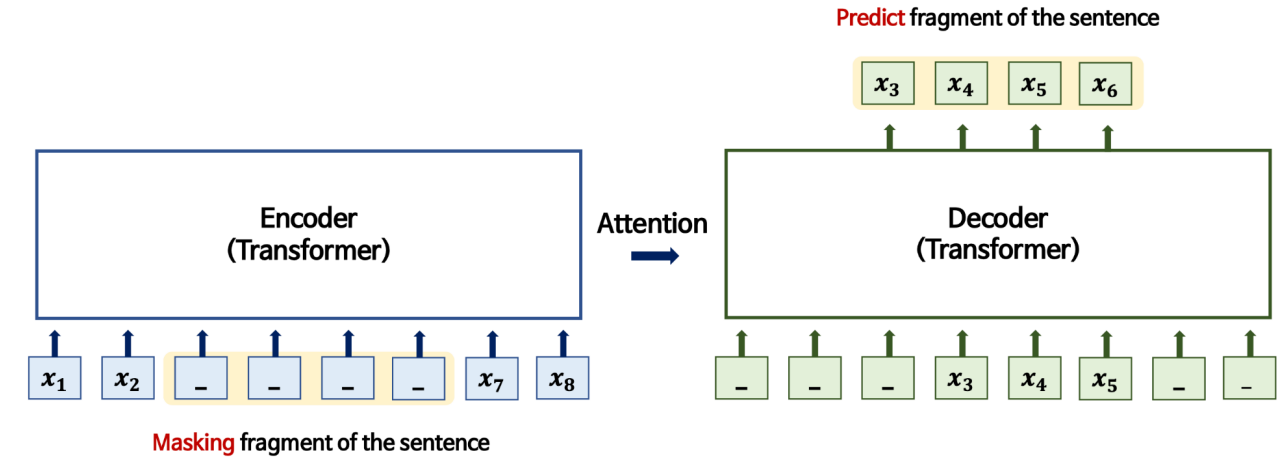
- 3.2 Masked Sequence to Sequence Pre-training
  - Some notations
    - $x^{\backslash u:v}$: modified x masked from u to v, $0 < u < v < m$
      (m *is the number of tokens of original sentence* x)
    - $k := v - u + 1$ (k *is the number of masked tokens*)
    - [𝕄] : special symbol replaced by maksed token
    - $x^{u:v}$: sentence fragment of x from u to v

- 3.2 Masked Sequence to Sequence Pre-training
  - Objective function
    - $L(\theta; X) = \frac{1}{|X|} \Sigma_{x\in X} \log P(x^{u:v}|x^{\backslash u:v}; \theta)$

      $= \frac{1}{|X|} \Sigma_{x\in X} \log \prod_{t=u}^{v} P(x_t^{u:v}|x_{<t}^{u:v}, x^{\backslash u:v}; \theta)$ $\because$ *chain rule*

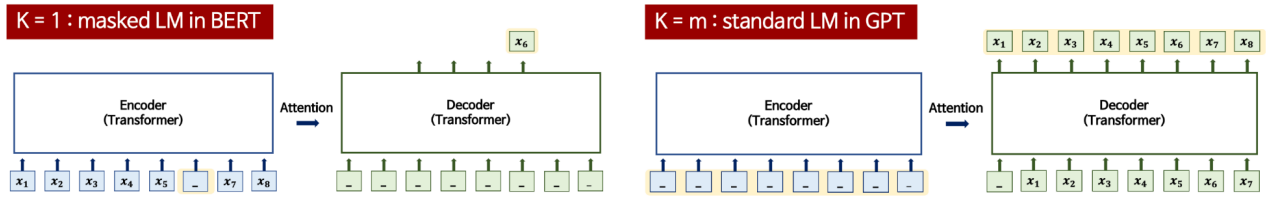
MASS (Masked Sequence to Sequence Pre-training for Language Generation)

Mass model structure



: Encoder와 Decoder의 input을 다르게 함으로서 Decoder가 Encoder에 더 의존할 수 있도록 함

K : length of the masked fragment of the sentence



BART (Bi-directional and Autoregressive Transformer)

Interpretation of the Name of Language Model

BART

Bidirectional and Auto-Regressive Transformer

① ②

① **Bidirectional:** BERT structure (Auto-encoder)

② **Auto-Regressive:** GPT (Decoder)

+

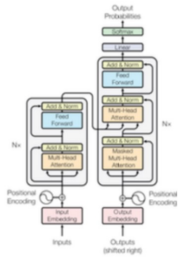
③ **Noise Flexibility:** various noise generation methods / masking schemes during the pre-training phase

➔ Well-rounded tasks for NLG and NLU

BART (Bi-directional and Autoregressive Transformer)

What Does This All Mean?

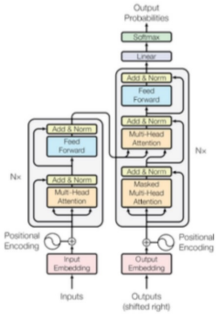
① Bidirectional **② Auto-Regressive** **③ Noise Flexibility**



➔ Well-rounded tasks for NLG and NLU

BART Architecture

Compared with BERT and GPT



Different from BERT,

1. BART has no additional FFN before word-prediction
2. BART's last encoder layer conducts cross-attention with each decoder layer

Similar to GPT,

1. ReLU Activations → GeLU Activations
2. Parameters initialized from N(0, 0.02)

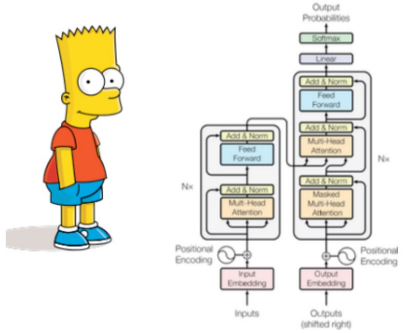
Size of Models:

1. Base : 6x encoder, 6x decoder
2. Large : 12x encoder, 12x decoder

BART (Bi-directional and Autoregressive Transformer)

BART Architecture

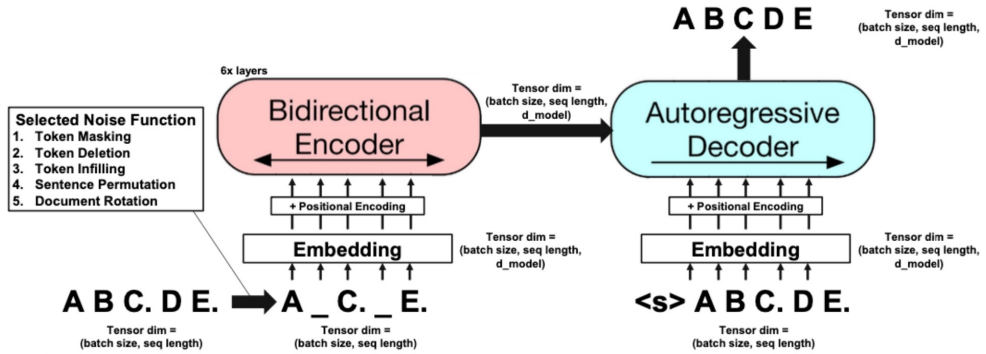
Goals to Achieve



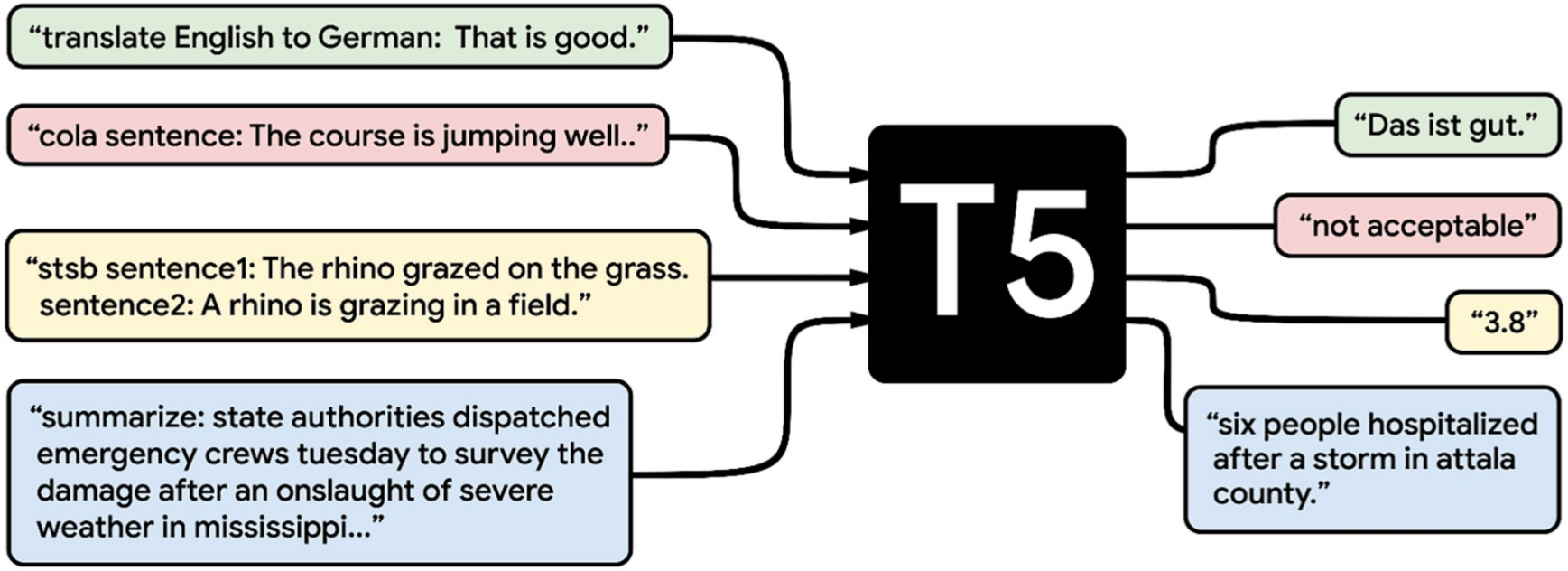
1. **Solves weakness of BERT:**
   According to Yang et al. (2019) BERT's problems are
   1. Independence assumption
   2. Pretrain-finetune discrepency

2. **A versatile model for variety of NLP tasks :**
   Different from other end-task focused MLM variants, create a high-performing model capable of solving a variety of tasks.

Overview of the Pre-training Process



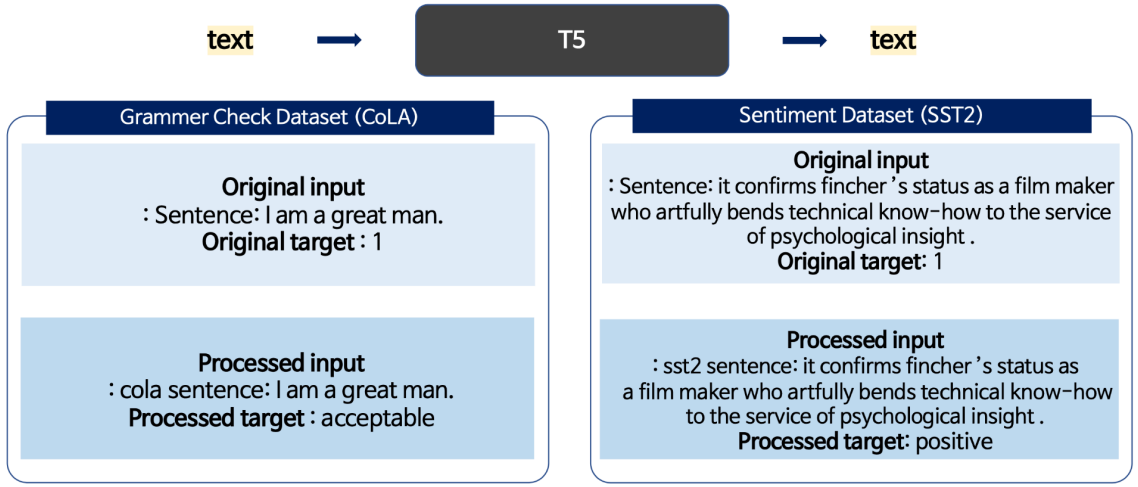
T5 (Exploring the Limits of Transformer Learning with a Unified Text-to-Text Transformer)



"Unified framework that converts every language problem into a text-to-text format"

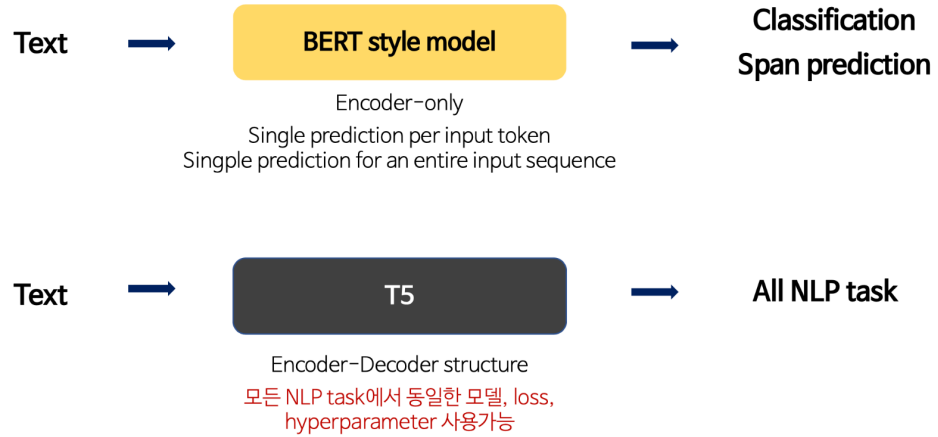
T5 (Exploring the Limits of Transformer Learning with a Unified Text-to-Text Transformer)

1. What is text to text ?



: text 형태로 주어진 문제에서 text 정답 찾기

2. Transfer learning in NLP



T5 (Exploring the Limits of Transformer Learning with a Unified Text-to-Text Transformer)

2. Transfer learning in NLP (T5에 적용된 방법론들)

1) Model Architecture

Encoder , Decoder only 모델 보다
Basic transformer 구조가 높은 성능을 보임

2) Pretraining Objectives

Pretraining에서 Noising 된 input을
Denoising하며 단어를 예측하는 방식이
가장 효율적인 방법임

3) Unlabeled datasets

Domain specific data는 task에 도움이 되지만
데이터의 크기가 작은경우 overfitting을 야기함

4) Training strategies

multitask learning이
unsupervised pre-training과 비슷한 성능 보임
학습시 task별 적절한 proportion이 필요함

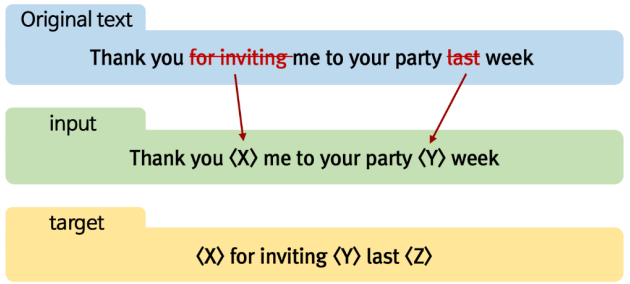
5) Scaling

모델 크기를 늘리거나 ,앙상블을 시도하며 실험 진행.
작은모델을 큰 데이터로 학습하는게 효과적이라는것 발견함

6) Pushing the limits

110억개 파라미터를 가지는 모델을 훈련하여 SOTA 달성함
1 trillon 개가 넘는 token에 대해 훈련 진행함

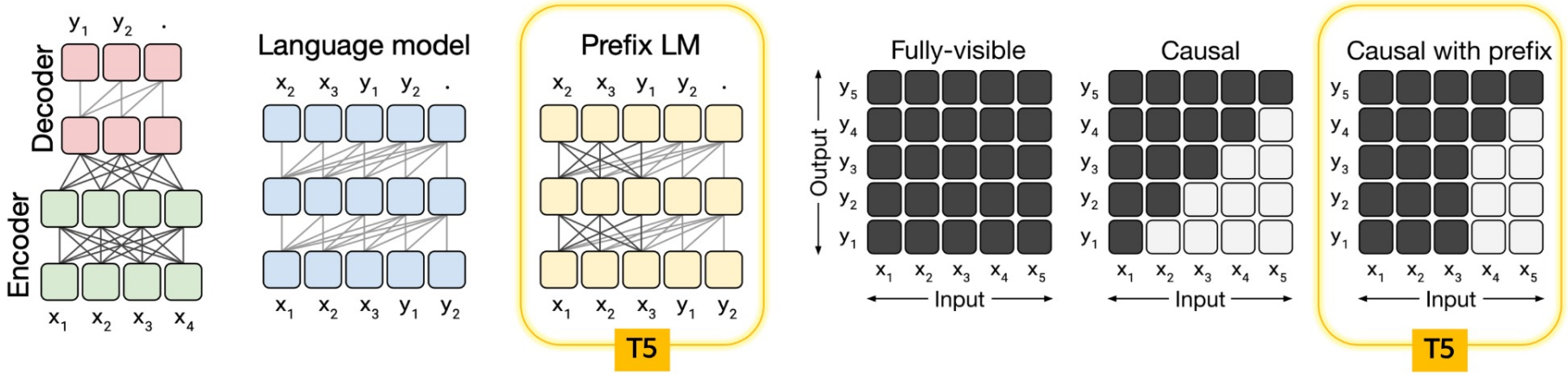
3. Training objective : Modified MLM



- MLM은 bidirectional model구조를 가짐
- BERT는 하나의 token에 masking을 하지만 T5 연속된 token을 하나의 mask로 바꿈 BART와 비슷
- Encoder-Decoder 구조로 input과 Target을 가지고 있음
- Input에서 mask 되지 않은 부분을 target에서 맞춰야 함 MASS와 비슷
- Output level에서 FFNN + Softmax을 통해 시퀀스 생성

T5 (Exploring the Limits of Transformer Learning with a Unified Text-to-Text Transformer)

4. Model structure



Model structure

1) Encoder - Decoder

Encoder : fed an input sequence
Decoder : produces a new output sequence

2) Language model

autoregressively produce an output sequence

3) Prefix Language model

Source data : bidirectional attention
Generation part : unidirectional attention

More Variants of Existing PTMs: Masking Strategy: SpanBERT (Improving Pre-training by Representing and Predicting Spans)

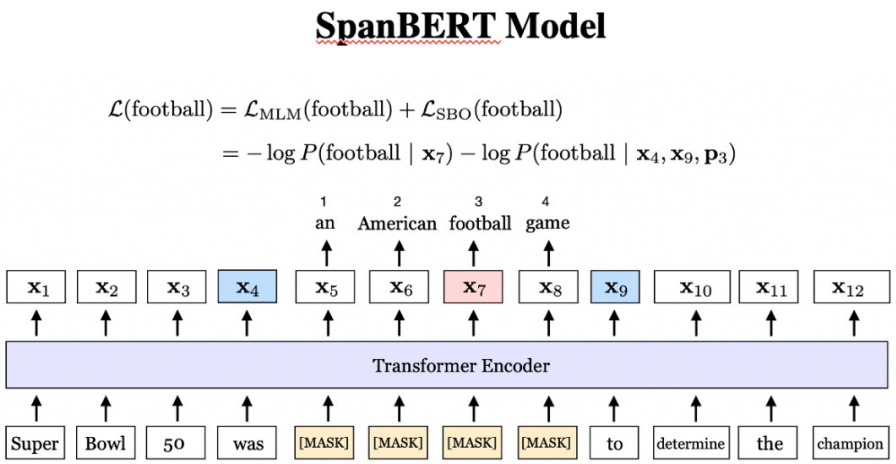
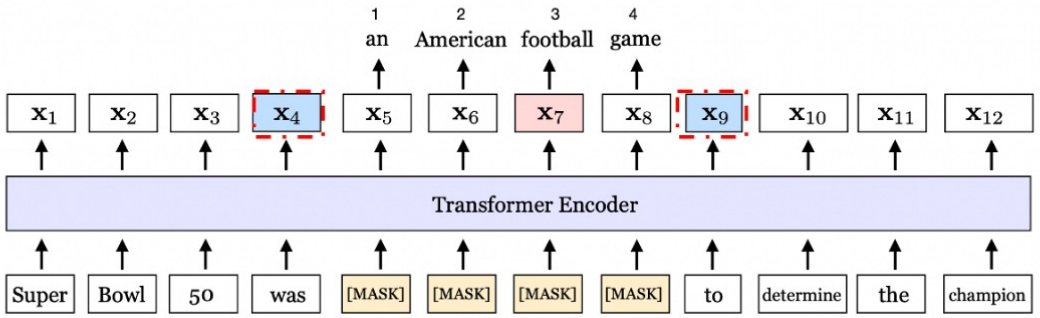


Figure 1: An illustration of SpanBERT training. The span *an American football game* is masked. The span boundary objective (SBO) uses the output representations of the boundary tokens, $\mathbf{x}_4$ and $\mathbf{x}_9$ (in blue), to predict each token in the masked span. The equation shows the MLM and SBO loss terms for predicting the token, *football* (in pink), which as marked by the position embedding $\mathbf{p}_3$, is the *third* token from x_4.

1. different random process to mask spans of tokens, rather than individual ones.
2. **SBO** – trying to predict the entire masked span using only the representations of the tokens at the span's boundary.
3. Samples a single contiguous segment of text for each training example. (**No NSP**)

Span Boundary Objective(SBO)

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$
$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$



Sequence: $(x_1, \ldots, x_n)$
Given a masked span of tokens
$(x_s, \ldots, x_e) \in Y$ (where (s,e) indicates its start and end positions)

we represent each token x_i in the span using the output encodings of the *external* boundary tokens x_s-1 and x_e+1, as well as the position embedding of the target token p_i-s+1:

$y_i = f(x_s-1, x_e+1, p_i-s+1)$
*PE: Relative positions with respect to the start SBO token
*f(·): 2 layers FFNN with GeLU activations and Layer Normalization

Span selection models typically create a fixed-length representation of a span using its boundary tokens(start and end).

→ To support such models, we use the representations for the **end of the span** to summarize as much of the internal span content as possible.

RoundUp: Evolution of Transformer Architecture

02 Utilizing Multi-Source Data

Multilingual, Multimodal,
Knowledge Enhanced Pretraining

Utilizing multi-source data

- Multilingual pre-training
  - Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks
- Multimodal pre-training
  - Unicoder-VL: A Universal Encoder for Vision and Language by Cross-modal Pre-training
  - ViLBERT: Pretraing Task-Agnostic Visionlinguistic Representations for Vision-and-Language Tasks
  - Zero-Shot Text-to-Image Generation
- Knowledge enhanced pre-training
  - KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation
  - KGLM: Using Knowledge-Graphs for Fact-Aware Language Modeling
  - ERNIE(1.0): Enhanced Representation through Knowledge Integration
  - ERINIE 2.0: A Continual Pre-training Framework for Language Understanding
  - A Knowledge-Enhanced Pretraining Model for Commonsense Story Generation

Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

- Summary
  - We present Unicoder, a universal language encoder that is insensitive to different languages.
  - Given an arbitrary NLP task, a model can be trained with Unicoder using training data in one language and directly applied to inputs of the same task in other languages. Comparing to similar efforts such as Multilingual BERT (2018) and XLM (2019), three new crosslingual pre-training tasks are proposed, including **cross-lingual word recovery, crosslingual paraphrase classification and crosslingual masked language model**.
  - Experiments are performed on two tasks: cross-lingual natural language inference (XNLI) and cross-lingual question answering (XQA), where XLM is our baseline.
  - On XNLI, 1.8% averaged accuracy improvement (on 15 languages) is obtained.
  - On XQA, which is a new cross-lingual dataset built by us, 5.5% averaged accuracy improvement (on French and German) is obtained.

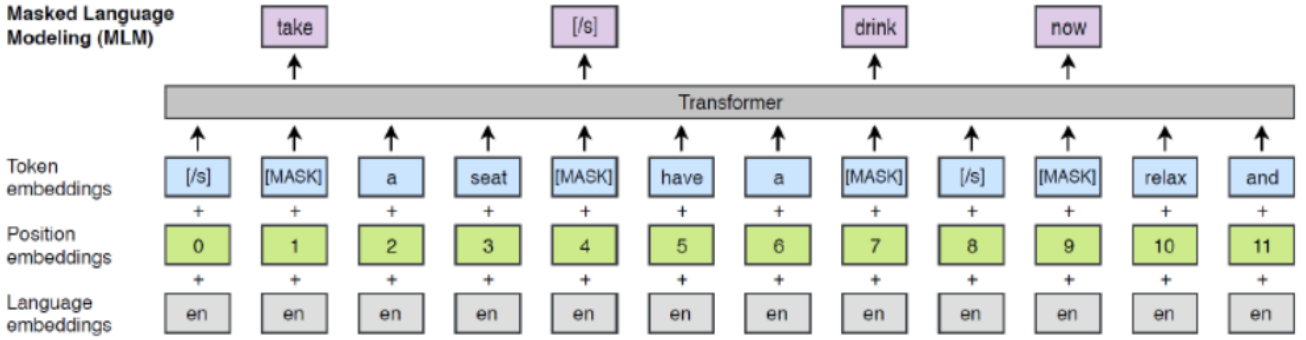
Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

- Approach : model structure
  - Model structure based on XLM
  - Shared sub-word vocabulary over multiple languages
    - Using byte pair encoding (BPE)
  - Downsampling rich resource languages corpus
    - Preventing words of low-resource languages from being split too much

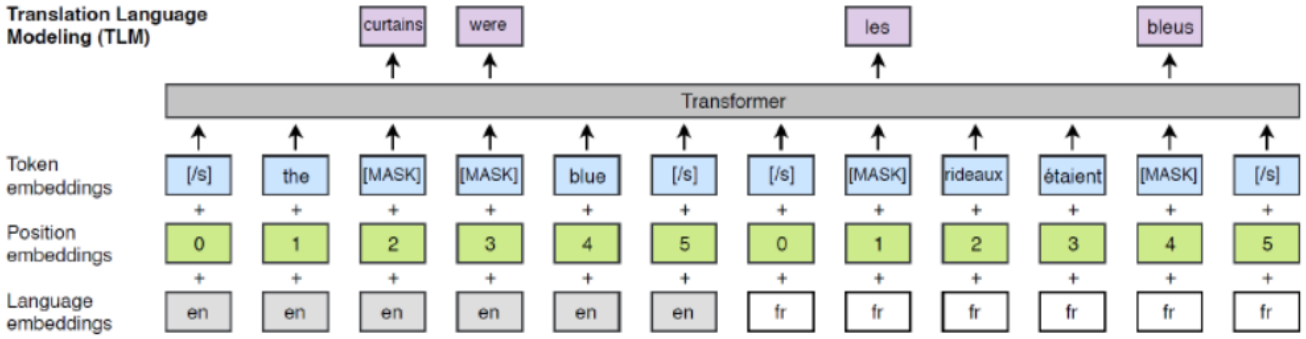
- Approach : pre-training tasks in Unicoder
  - Masked Language Modeling from BERT
  - Translation Language Modeling from XLM
  - Cross-lingual Word Recovery new
  - Cross-lingual paraphrase classification new
  - Cross-lingual masked language model new

Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

- Approach : pre-training tasks in Unicoder
  - Masked Language Modeling from BERT



- Translation Language Modeling from XLM
  - Concatenating parallel sentences → predicting masked words



Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

- Approach : pre-training tasks in Unicoder
  - Cross-lingual word recovery
    - Word alignment between two languages

$X = (x_1, x_2, \dots, x_m)$ from language s
$Y = (y_1, y_2, \dots, y_n)$ from language t

$x_i^t = \sum_{j=1}^{n} softmax(A_{ij}) y_j^t$

x_i^s: word embeddings of x_i
y_j^t: word embeddings of y_j

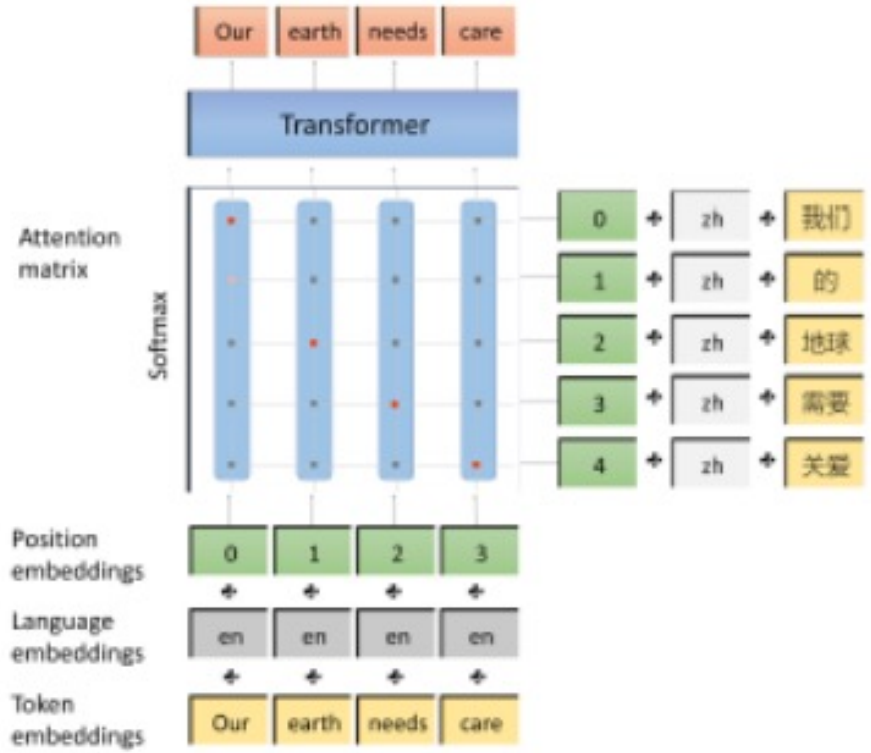
- $A_{ij} = W[x_i^s, y_j^t, x_i^s \odot y_j^t]$ (Attention Matrix)

→ Attention b/w language s and t

- $x_i^t = \sum_{j=1}^{n} softmax(A_{ij}) y_j^t$

→ Words x_i aligned to y_j

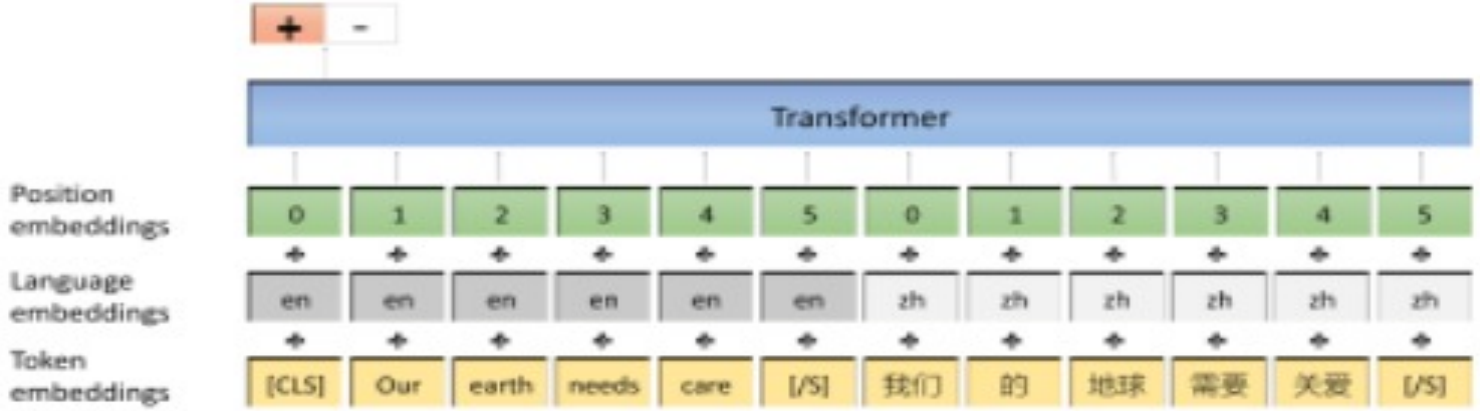
- Recover original text X from x_i^t



(a) Cross-lingual word recovery

Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

- Approach : pre-training tasks in Unicoder
  - Cross-lingual paraphrase classification : whether two sentence same meaning
  - Positive sample : obtained from machine translation dataset
  - Negative sample : select with high similarity to X but not equal to Y



(b) Cross-lingual paraphrase classification

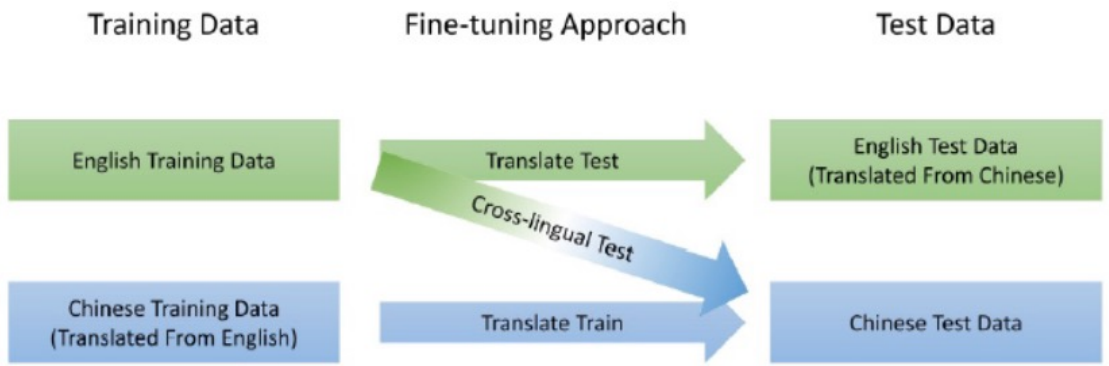
Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

- Approach : pre-training tasks in Unicoder
  - Cross-lingual masked language model
  - Sequence of sentences written with different languages

(c) Cross-lingual masked language model

Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

- Approach : fine-tuning
  - Typical cross-lingual fine-tuning settings
    - Only one language has training data
    - Test is conducted on other languages
  - Three baseline approaches
  - Cross-lingual test
    - Pretrain a model on training data in source
    - Fine-tune the model on test in target
  - Translate-train
    - Translate the training data into the target
    - Pre-train a model on the translated training data
    - Fine-tune the model on test in target
  - Translate-test
    - Translate the test data into the source
    - Pretrain a model on training data in source
    - Fine-tune the model on the translate test data



Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

Experiments

	en	fr	de	average
Machine translate at training (TRANSLATE-TRAIN)				
XLM (Lample and Conneau, 2019)	80.2	65.1	63.3	64.2
Unicoder	81.1	66.2	66.5	66.4
Evaluation of cross-lingual sentence encoders (Cross-lingual TEST)				
XLM (Lample and Conneau, 2019)	80.2	62.3	61.7	62.0
Unicoder	81.1	64.1	63.7	63.9
Multi-language Fine-tuning				
BERT (Devlin et al., 2018)	76.4	61.6	64.6	63.1
XLM (Lample and Conneau, 2019)	80.7	67.1	68.2	67.7
Unicoder	**81.4**	**69.3**	**70.1**	**69.7**

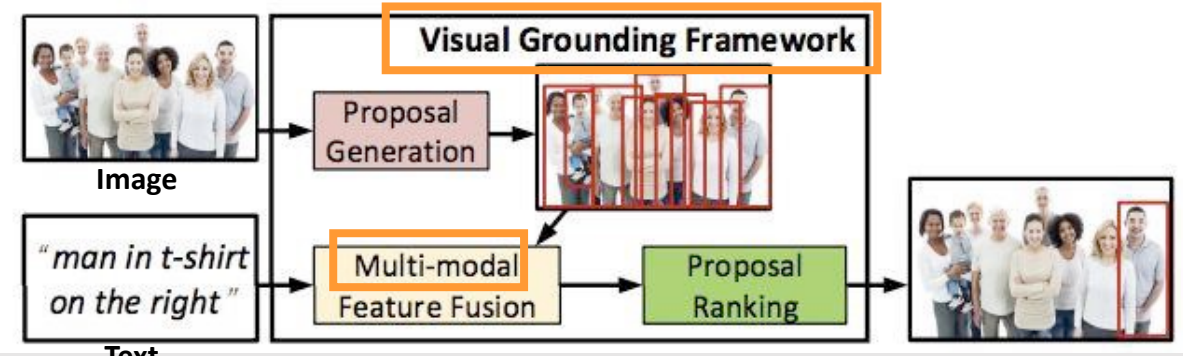
Table 3: Results on the XQA. The average column is the average of fr and de result.

	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur	average
Machine translate at training (TRANSLATE-TRAIN)																
Conneau et al. (2018)	73.7	68.3	68.8	66.5	66.4	67.4	66.5	64.5	65.8	66.0	62.8	67.0	62.1	58.2	56.6	65.4
Multilingual BERT (Devlin et al., 2018)	81.9	-	77.8	75.9	-	-	-	-	70.7	-	-	76.6	-	-	61.6	-
Multilingual BERT from Wu and Dredze 2019	82.1	76.9	78.5	74.8	72.1	75.4	74.3	70.6	70.8	67.8	63.2	76.2	65.3	65.3	60.6	71.6
XLM (Lample and Conneau, 2019)	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
Unicoder	85.1	80.0	81.1	79.9	77.7	80.2	77.9	75.3	76.7	76.4	75.2	79.4	71.8	71.8	64.5	76.9
Machine translate at test (TRANSLATE-TEST)																
Conneau et al. (2018)	73.7	70.4	70.7	68.7	69.1	70.4	67.8	66.3	66.8	66.5	64.4	68.3	64.2	61.8	59.3	67.2
Multilingual BERT (Devlin et al., 2018)	81.4	-	74.9	74.4	-	-	-	-	70.4	-	-	70.1	-	-	62.1	-
XLM (Lample and Conneau, 2019)	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
Unicoder	85.1	80.1	80.3	78.2	77.5	78.0	76.2	73.3	73.9	72.8	71.6	74.1	70.3	65.2	66.3	74.9
Evaluation of cross-lingual sentence encoders (Cross-lingual TEST)																
Conneau et al. (2018)	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
Multilingual BERT (Devlin et al., 2018)	81.4	-	74.3	70.5	-	-	-	-	62.1	-	-	63.8	-	-	58.3	-
Multilingual BERT from Wu and Dredze 2019	82.1	73.8	74.3	71.1	66.4	68.9	69	61.6	64.9	69.5	55.8	69.3	60.0	50.4	58.0	66.3
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 (Lample and Conneau, 2019)	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
Unicoder	85.1	79.0	79.4	77.8	77.2	77.2	76.3	72.8	73.5	76.4	73.6	76.2	69.4	69.7	66.7	75.4
Multi-language Fine-tuning																
XLM (Lample and Conneau, 2019)	85.0	80.8	81.3	80.3	79.1	80.9	78.3	75.6	77.6	78.5	76.0	79.5	72.9	72.8	68.5	77.8
Unicoder w/o Word Recovery	85.2	80.5	81.8	80.9	79.7	81.1	79.3	76.2	78.2	78.5	76.4	79.7	73.4	73.6	68.8	78.2
Unicoder w/o Paraphrase Classification	85.5	81.1	82.0	81.1	80.0	81.3	79.6	76.6	78.2	78.2	75.9	79.9	73.7	74.2	69.3	78.4
Unicoder w/o Cross-lingual Language Model	85.5	81.9	81.8	80.5	80.5	81.0	79.3	76.4	78.1	78.3	76.3	79.6	72.9	73.0	68.7	78.3
Unicoder	**85.6**	**81.1**	**82.3**	**80.9**	**79.5**	**81.4**	**79.7**	**76.8**	**78.2**	**77.9**	**77.1**	**80.5**	**73.4**	**73.8**	**69.6**	**78.5**

Table 2: Test accuracy on the 15 XNLI languages. This table is organized by fine-tuning and test approaches. TRANSLATE-TRAIN is to machine translate English training data to target language and fine-tune with this translated data; TRANSLATE-TEST is machine translate target language test data to English, the fine-tuning is conducted on English; Cross-lingual TEST is to fine-tune on English and directly test on target language; Multi-language Fine-tune is to fine-tune on machine translated training data on all languages.

Multimodal: Modality

What is **Modality**?
The “way” for a certain event(Data)
=> composing “source”(text, speech, audio, image..) shares a modality

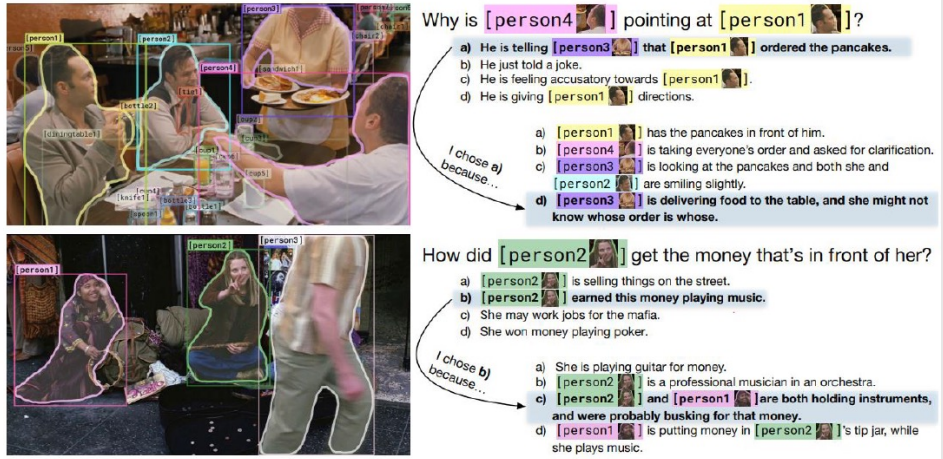


Main issues of Vision Language multimodal Learning:
Learning the “**Aligning**” between Visual Stimuli and Natural Language

Previous works lack a unified foundation, and instead starting with separate pretrained models for each tasks and merge them often results in poor generalizations when paired data is limited or biased

- VQA(Visual Question Answering)

- Visual Commonsense Reasoning



Multimodal: Vision & Language

- Pre-Trained Multi-modal Models: Model Architecture Comparison for Vision

	Method	Architecture	Visual Token	Pre-train Datasets	Pre-train Tasks
Works Under Review / Just Got Accepted	VideoBERT (Sun et al., 2019b)	single cross-modal Transformer	video frame	Cooking312K (Sun et al., 2019b)	1) sentence-image alignment 2) masked language modeling 3) masked visual-words prediction
	CBT (Sun et al., 2019a)	two single-modal Transformer (vision & language respectively) + one cross-modal Transformer	video frame	Cooking312K (Sun et al., 2019b)	1) sentence-image alignment 2) masked language modeling 3) masked visual-feature regression
	ViLBERT (Lu et al., 2019)	one single-modal Transformer (language) + one cross-modal Transformer (with restricted attention pattern)	image RoI	Conceptual Captions (Sharma et al., 2018)	1) sentence-image alignment 2) masked language modeling 3) masked visual-feature classificatio
	B2T2 (Alberti et al., 2019)	single cross-modal Transformer	image RoI	Conceptual Captions (Sharma et al., 2018)	1) sentence-image alignment 2) masked language modeling
	LXMERT (Tan & Bansal, 2019)	two single-modal Transformer (vision & language respectively) + one cross-modal Transformer	image RoI	‡ COCO Caption + VG Caption + VG QA + VQA + GQA	1) sentence-image alignment 2) masked language modeling 3) masked visual-feature classificatio 4) masked visual-feature regression 5) visual question answering
	VisualBERT (Li et al., 2019b)	single cross-modal Transformer	image RoI	COCO Caption (Chen et al., 2015)	1) sentence-image alignment 2) masked language modeling
	Unicoder-VL (Li et al., 2019a)	single cross-modal Transformer	image RoI	Conceptual Captions (Sharma et al., 2018)	1) sentence-image alignment 2) masked language modeling 3) masked visual-feature classificatio
	Our VL-BERT	single cross-modal Transformer	image RoI	Conceptual Captions (Sharma et al., 2018) + BooksCorpus (Zhu et al., 2015) + English Wikipedia	1) masked language modeling 2) masked visual-feature classificatio

X ERT is pre-trained on COCO Caption (Chen et al., 2015), VG Caption (Krishna et al., 2017), VG QA (Zhu et al., 2016), VQA (Antol et al.

ViLBERT: Pretraing Task-Agnostic Visionlinguistic Representations for Vision-and-Language Tasks

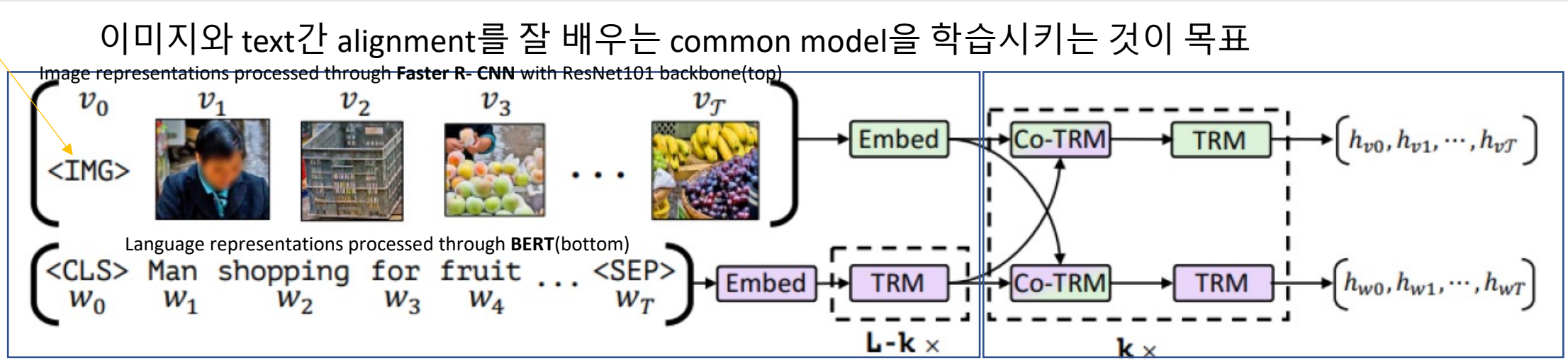
- Summary
  - We present ViLBERT, a model for learning task-agnostic joint representations of image content and natural language.
  - We extend the popular BERT architecture to a multi-modal two-stream model, processing both visual and textual inputs in separate streams that interact through **co-attentional** transformer layers.
  - We pretrain our model through two proxy tasks on the large, automatically collected Conceptual Captions dataset and then transfer it to multiple established vision-and-language tasks – visual question answering, visual commonsense reasoning, referring expressions, and caption-based image retrieval – by making only minor additions to the base architecture.
  - We observe significant improvements across tasks compared to existing task-specific models – achieving state-of-the-art on all four tasks.
  - Our work represents a shift away from learning groundings between vision and language only as part of task training and towards treating visual grounding as a pretrainable and transferable capability

Multimodal Pre-Training: ViLBERT

Vision Language BERT

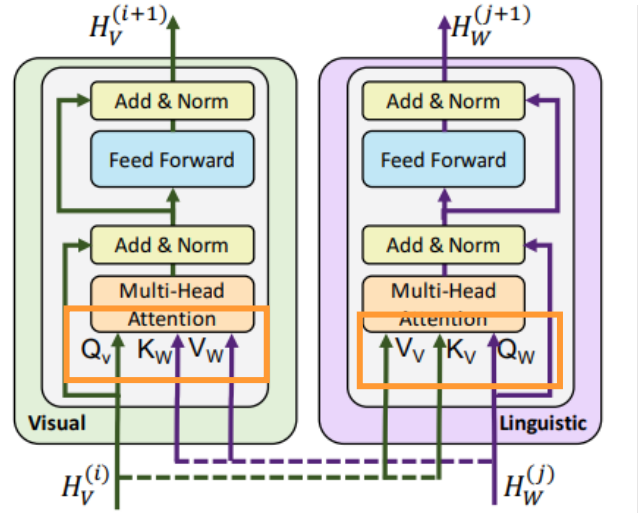
Image representations are marked with a new special token <IMG> (이미 BERT에는 special token을 사용-> 이미지에만 더 붙여준다)

이미지와 text간 alignment를 잘 배우는 common model을 학습시키는 것이 목표



Individual paths for image(top)/language(bottom) representation processing

1. Self supervised Learning: BERT, GPT와 같은 learning 방식
2. Separation of streams for Vision/Language processing, and use **Co-attention** for the interaction between two modalities(modalities 사이의 interaction을 구하는 요소)
3. Used Visual Language pretraining tasks for training objectives
   1. **Masked multi modal modelling:** Predicts a **distribution over semantic classes** for the masked image regions, instead of direct regression
      –--> Minimize the KL divergence with the distribution from the feature extractor(R-CNN에서 얻게 되는 distribution), Unlikely to reconstruct " image features, but will learn to identify the high-level semantics of visual content
   2. **Multi modal alignment prediction** ≈ Next Sentence Prediction in BERT
      predict whether the image and text are aligned



(b) Our co-attention transformer layer

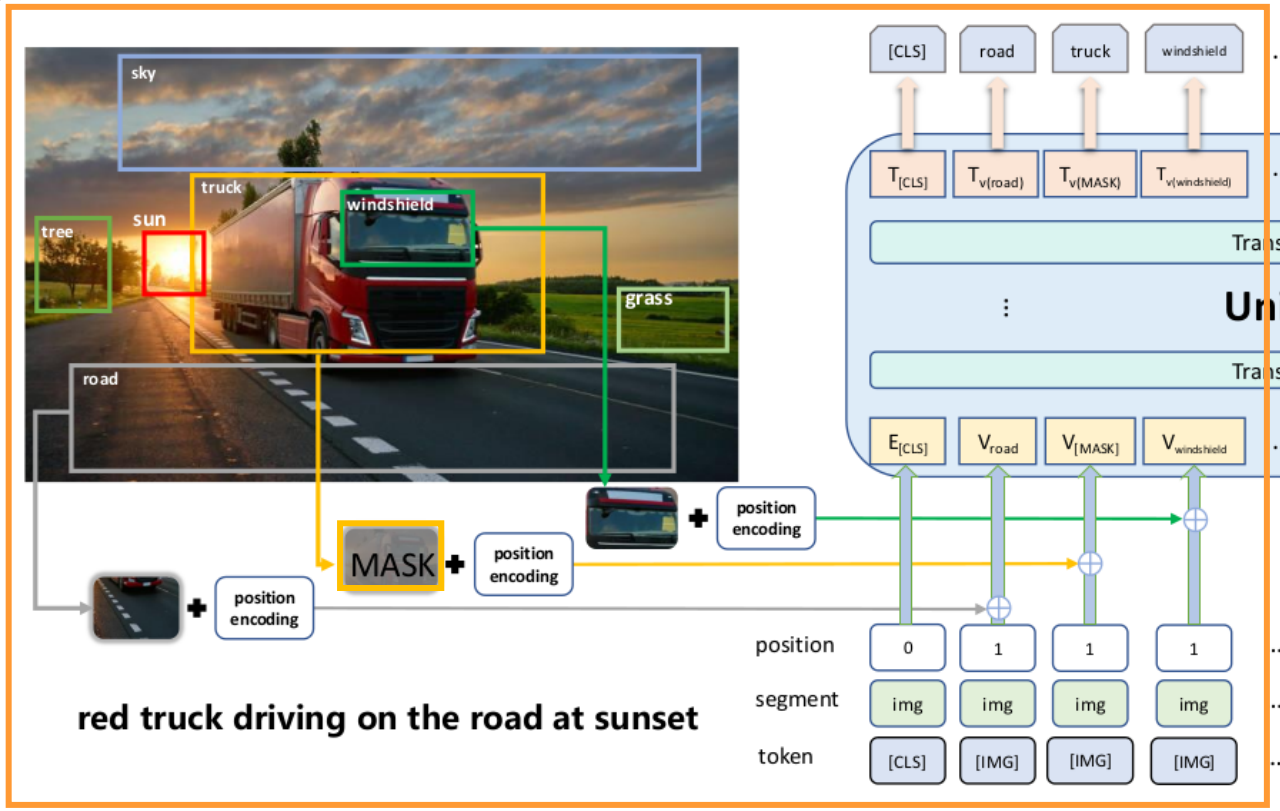
Unicoder-VL: A Universal Encoder for Vision and Language by Cross-modal Pre-training

- Summary
  - We propose Unicoder-VL, a universal encoder that aims to learn joint representations of vision and language in a pre-training manner.
  - Borrow ideas from cross-lingual pretrained models, such as XLM (2019) and Unicoder (2019), both visual and linguistic contents are fed into a multi-layer Transformer (2017) for the cross-modal pre-training, where three pre-trained tasks are employed, including Masked Language Modeling (MLM), Masked Object Classification (MOC) and Visual-linguistic Matching (VLM).
  - The first two tasks learn context-aware representations for input tokens based on linguistic and visual contents jointly.
  - The last task tries to predict whether an image and a text describe each other.
  - After pretraining on large-scale image-caption pairs, we transfer Unicoder-VL to caption-based image-text retrieval and visual commonsense reasoning, with just one additional output layer.
  - We achieve state-of-the-art or comparable results on both two tasks and show the powerful ability of the cross-modal pre-training

Multimodal Pre-Training: Unicoder-VL

A Universal Encoder for Vision and Language by Cross-modal Pre-training

Fine Tuning Task :
- Image, Text Retrieval
- Visual Common Reasoning



$$L = (L_{MLM} + L_{MOC}) \cdot I[y = 1] + L_{VLM}$$

- MLM (Masked Language Modeling)
- MOC (Masked Object Classification)
- VLM (Visual Language Matching)

MLM, MOC 는 positive pair 로만 학습하고 , VLM 은 negative pair 도 사용

Self attention : cross modality 를 학습

Input Embedding

- Linguistic : Bert와 동일 $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_T\}$
- Image : Faster-RCNN의 ROI feature
  - Visual feature $\mathbf{v} = \{v_1, \ldots, v_I\}$
  - Location feature : $\frac{x_1}{W}, \frac{y_1}{H}, \frac{x_2}{W}, \frac{y_2}{H}, \frac{(x_2-x_1)(y_2-y_1)}{HW}$
    - 5-d vector
  - 두 feature의 합을 최종 Image Embedding으로 사용

$$L_{MOC}(\theta) = -E_{(\mathbf{w},\mathbf{v})\sim D \sum_{i=1}^{M} CE\left(c\left(\mathbf{v}_\mathbf{m}^{(i)}\right), g_\theta\left(\mathbf{v}_\mathbf{m}^{(i)}\right)\right)}$$

Masked Visual Embedding: 90% zero vector로 초기화, 10%는 불변 $\mathbf{v}_m$
Goal : $\mathbf{v}_m$ 을 한번 더 FC Layer+Softmax에 통과시켜 **class id**를 예측

- $c\left(\mathbf{v}_\mathbf{m}^{(i)}\right)$: Pseudo GT (one-hot vector)
- $g_\theta\left(\mathbf{v}_\mathbf{m}^{(i)}\right)$: predicted class id

$$L_{MLM}(\theta) = -E_{(\mathbf{w},\mathbf{v})\sim D} \log P_\theta\left(\mathbf{w}_\mathbf{m} || \mathbf{w}_{\setminus m}, \mathbf{v}\right)$$

$$L_{VLM}(\theta) = -E_{(\mathbf{w},\mathbf{v})\sim D}[y \log s_\theta(\mathbf{w},\mathbf{v}) + (1-y)\log(1 - s_\theta(\mathbf{w},\mathbf{v}))]$$

Scoring function $s_\theta(\mathbf{w},\mathbf{v})$: word embedding, visual embedding간의 유사도

y : word embedding, visual embedding이 matched=1, unmatched=0

즉, positive pair negative pair를 모두 학습에 활용

Multimodal Pre-Training: DALL-E

Zero-Shot Text-to-Image Generation

DALL-E라는 이름은 초현실주의 화가 살바도르 달리(Salvador Dali)와 로봇 애니메이션 속 로봇 캐릭터 윌-E(WALL-E)에서 영감을 받아 이름을 DALL-E라고 지었다고 합니다.

- Transformer 기반 language model: **text to image generation** process 를 autoregressively train
- GPT 3 와 유사한 형태로 , Text, Image tokens 를 **single stream** 으로 학습
- 이미지 x,텍스트 y, 인코딩 된 이미지 token z의 **joint probability distribution**
- 학습 후 MS COCO dataset 에 대해 Zero shot(Training set이 주어지지 않는 상황)에서도 우수한 성능을 보임
- (GPT 170억 개 대비)120 억 개의 parameters 를 가지고 있으며 , 2.5 억 개의 text image pairs 로 학습
- Text to image generation 은 MS COCO, CUB 200 과 같은 상대적으로 소규모의 dataset 을 사용

Multimodal Pre-Training: DALL-E

ELBO(**E**vidence of **L**ower **Bo**und, ELB): Latent variable(z) model에 Bayes Theorem을 적용

Prior probability(사전 확률)

X <- Z (Latent variable $\boldsymbol{z}$, observed variable $\boldsymbol{x}$)

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Evidence

$$p(x) = \int p(x|z)p(z)dz$$

적분 계산이 어려우므로 (Gaussian) Normalizing할 수 있는 $\boldsymbol{q}$를 선택

Posterior probability(사후 확률)

$$q(z) \approx p(z|x)$$

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

Divergence 란 두 확률 분포의 다름의 정도를 나타내는 것

$$D_{KL}(q(z)||p(z|x)) = E_q[\log q(z)] - E_q[\log p(z|x)]$$

Bayes Theorem

$$= E_q[\log q(z)] - E_q[\log \frac{p(x|z)p(z)}{p(x)}]$$

Joint Probability

$$= E_q[\log q(z)] - E_q[\log \frac{p(x,z)}{p(x)}]$$

$$= E_q[\log q(z)] - E_q[\log p(x,z) - \log p(x)]$$

적분 시 1이 된다

$$= E_q[\log q(z)] - E_q[\log p(x,z)] + \log p(x)$$

- log p(x) 는 **상수**이므로 ELBO 가 증가하면 $D(KL)$은 감소
- **ELBO 를 최대화**하여 q(z) 와 p(z|x) 의 **차이 최소화** 가능

Evidence

$$\therefore \log p(x) = D_{KL}(q(z)||p(z|x)) + E_q[\log p(x,z)] - E_q[\log q(z)]$$

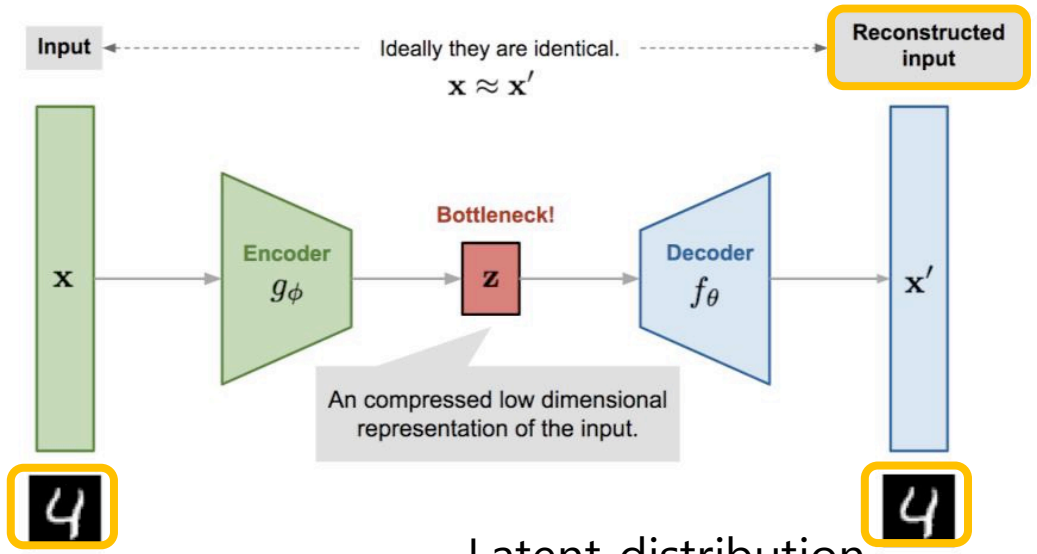
Jensen's inequality에 의하여 양수 — ELBO

$$L(\theta,\phi) = -E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$$

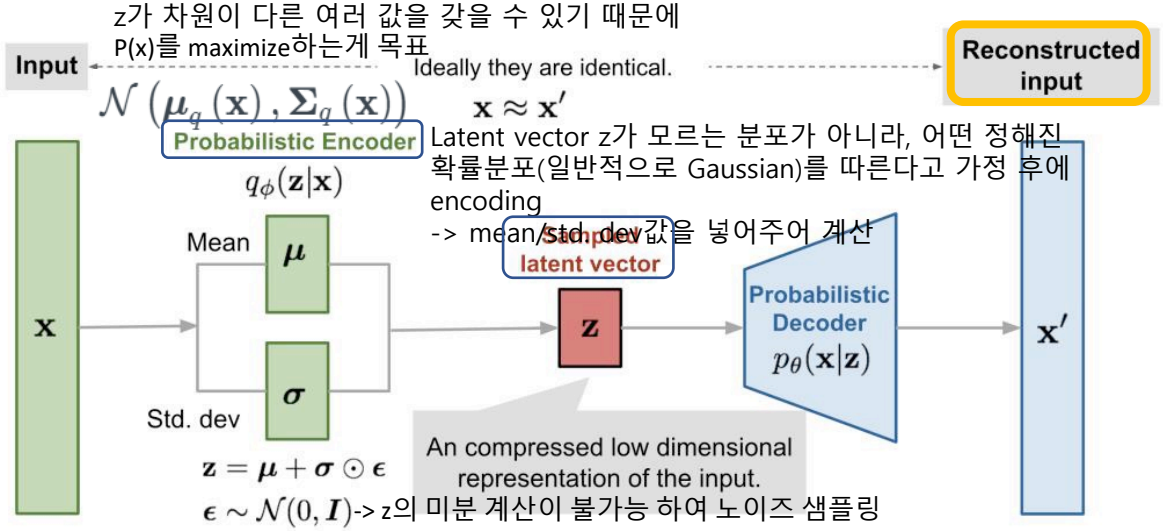
음수를 곱해 최소화 문제로 변형하여 loss function 으로 정의 후 stochastic gradient descent 로 최적화

Multimodal Pre-Training: DALL-E

AutoEncoder(AE)



Variational AutoEncoder(VAE)



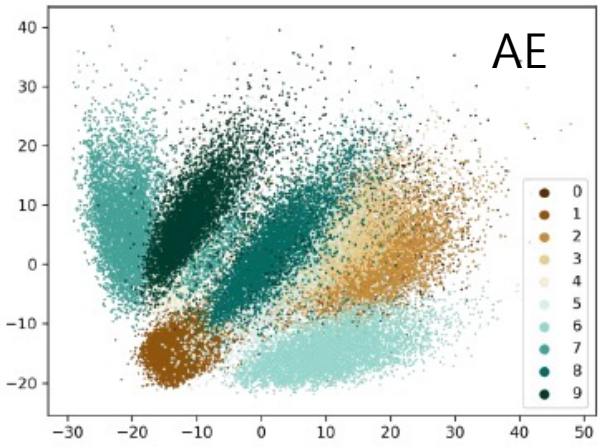
z가 차원이 다른 여러 값을 갖을 수 있기 때문에 P(x)를 maximize하는게 목표

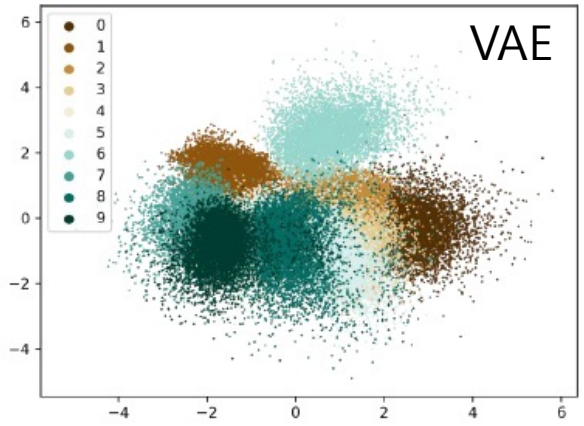
Latent vector z가 모르는 분포가 아니라, 어떤 정해진 확률분포(일반적으로 Gaussian)를 따른다고 가정 후에 encoding

-> mean/std. dev값을 넣어주어 계산

-> z의 미분 계산이 불가능 하여 노이즈 샘플링

Latent distribution





- VAE 는 latent vector 구분이 가능하며 , control 할 때 유리(가우시안을 따른다고 가정하기 때문)
- VAE 는 representation 을 sampling 가능하여 , train set 에 없는 데이터 판별 가능(아웃라이어 판별 가능)

$$\log p(x) \geq \underbrace{E_{z\sim q(z|x)}[\log p(x|z)]}_{\text{Reconstruction Term}} - \underbrace{D_{KL}\big(q(z|x)||p(z)\big)}_{\text{Regularization term}} = ELBO$$

Reconstruction Term	Regularization term
Image에 대해 latent vector를 구한 뒤 원래 이미지가 나올 수 있도록 학습	q(z|x)가 표준 정규 분포 p(z)와 유사하도록 조정

Multimodal Pre-Training: DALL-E

Vector Quantized Variational AutoEncoder(VQ-VQE)

- VAE는 latent vector가 continuous 한 반면,
VQ-VQE는 **latent vector**가 **discrete**
- 별도의 prior 분포를 가정하지 않고
모든 latent code에 대해 **uniform한 prior를 가정**.

$$L = \underbrace{\|\mathbf{x} - D(\mathbf{e}_k)\|_2^2}_{\text{reconstruction loss}} + \underbrace{\|\text{sg}[E(\mathbf{x})] - \mathbf{e}_k\|_2^2}_{\text{VQ loss}} + \underbrace{\beta\|E(\mathbf{x}) - \text{sg}[\mathbf{e}_k]\|_2^2}_{\text{commitment loss}}$$

Auto encoder와 동일한 역할

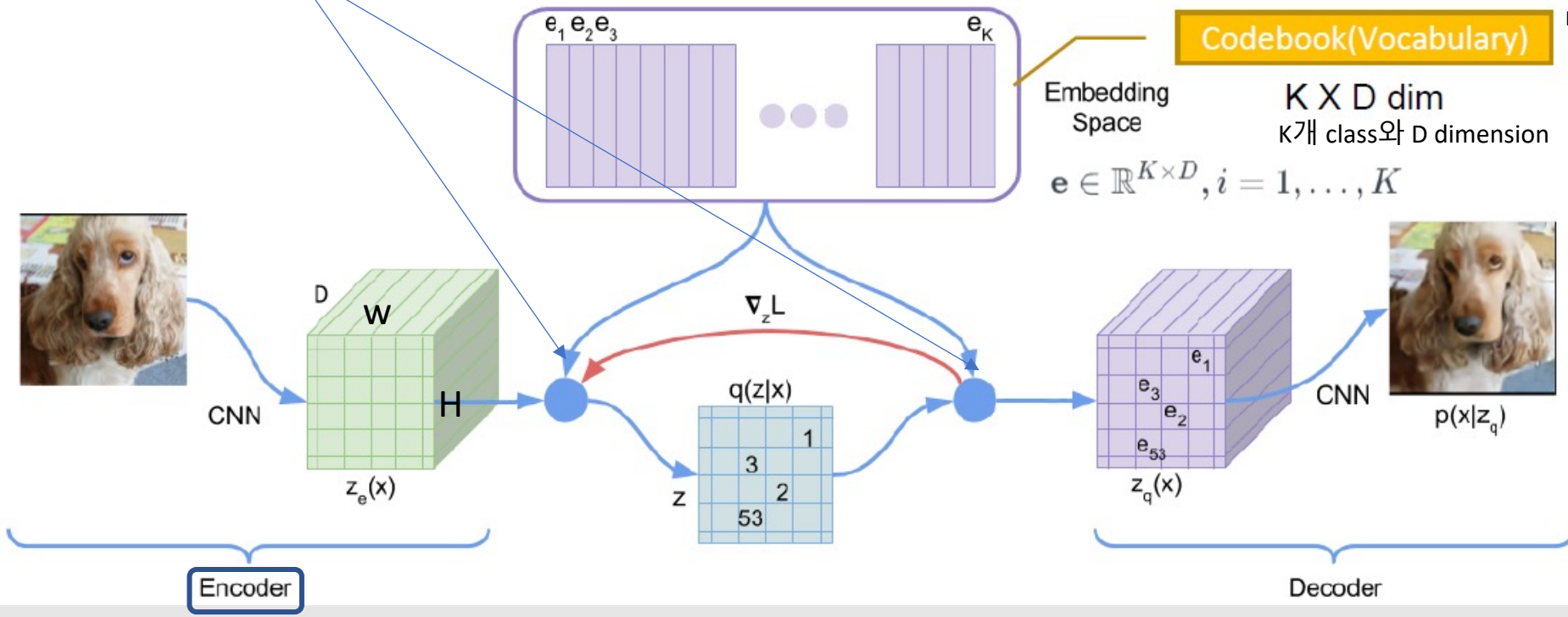
Codebook vector

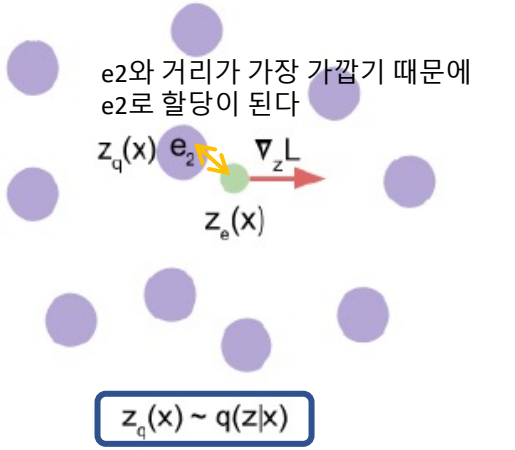
Encoding parameter 조정

sg(·): stop gradient operator
⇒ Zero gradient를 가짐
⇒ 고정 되어 있다

$$\mathbf{z}_q(\mathbf{x}) = \text{Codebook}(E(\mathbf{x})) = \mathbf{e}_k \text{ where } k = \arg\min_i \|E(\mathbf{x}) - \mathbf{e}_i\|_2$$

Image encoding vector와 가장 가까운 e를 넣어준다



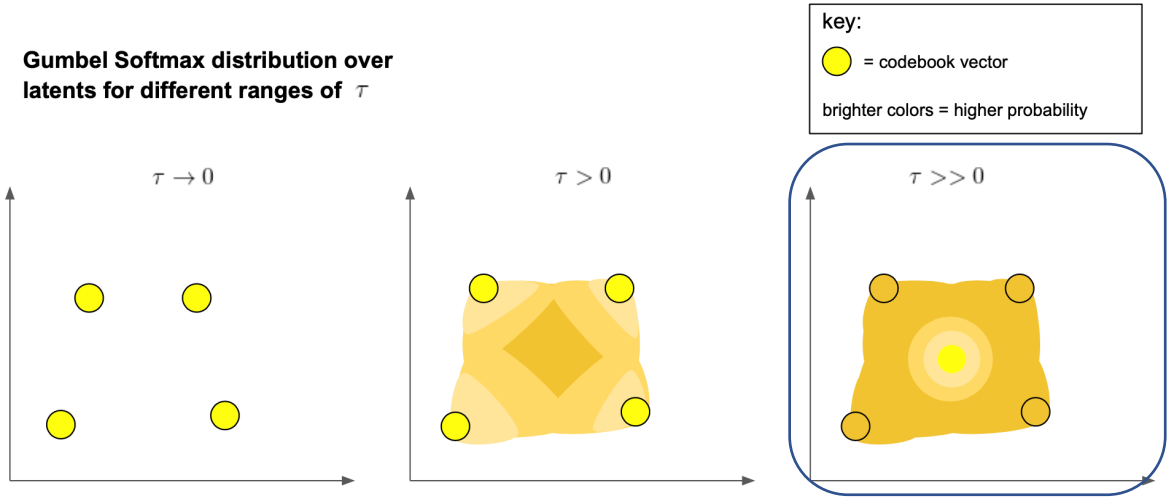


Visualisation of the embedding space

Multimodal Pre-Training: DALL-E

discrete VAE(dVAE)

- VQ VAE 와 유사하지만 , 하나의 codebook vector(discrete) 만 선택하지 않고 posterior **probability distribution 을 stochastic 하게 표현**
- **Backpropagation 을 위해** Gumbel softmax Trick 을 사용해 argmax 를 **softmax 로 근사** (argmax 사용 시 gradient 계산 불가)



Distribution의 형태로 사용

$g_1, g_2, \ldots, g_k$ 가 Standard Gumble 분포를 따르고 i.i.d. 일 때 클래스 확률이 $q(e_i|x)$ 인 카테고리 분포에서 샘플 z

$$z = codebook\left[\arg\max_i [g_i + \log(q(e_i|x))]\right]$$

$$y_i = \frac{e^{\frac{g_i + log(q(e_i|x))}{\tau}}}{\sum_{j=1}^{k} e^{\frac{g_j + log(q(e_j|x))}{\tau}}}$$

$\tau \to 0$

Temperature가 0 에 가까울수록 distribution 이 hard

학습할 때는 softmax 를 사용해 gradient 를 계산

Multimodal Pre-Training: DALL-E

Two Stage training

- Stage 1: dVAE를 training
  - 256 X 256 RGB image를 32 X 32 image tokens로 압축
  - 각 image token은 8192 values 중 1개
  - Transformer context size를 192배 (8 X 8 X 3) 압축
  - Image latent의 sequence를 예측 후 dVAE로 decoding
  - KL weight $\beta = 6.6$ 일 때 좋은 결과

256-> 32, RGB 3dim

아예 그림이 망가지지는 것은 아님(dVAE결과물)

Learning the Visual Codebook

$$\log p_{\theta,\psi}(\mathbf{x},\mathbf{y}) \geq E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})\right] - \beta D_{KL}\left(q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x})\,||\,p_\psi(\mathbf{y},\mathbf{z})\right)$$

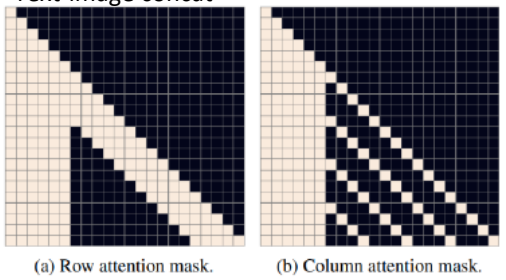
- Transformer를 고정한 상태로 dVAE encoder q_ϕ 와 dVAE decoder p_θ 를 학습

(token dim K=8192 codebook vector)

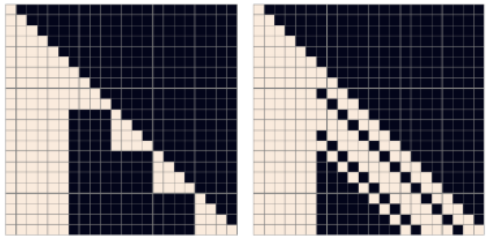
p_ψ 를 uniform distribution으로 놓음

- Stage 2: Autoregressive transformer를 train
  - 256 BPE-encoded text tokens와 32 X 32 image tokens를 이어서 joint distribution으로 모델

Text-image concat



(a) Row attention mask. (b) Column attention mask.



(c) Column attention mask with transposed image states. (d) Convolutional attention mask.

Learning the Prior

$$\log p_{\theta,\psi}(\mathbf{x},\mathbf{y}) \geq E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})\right] - \beta D_{KL}\left(q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x})\,||\,p_\psi(\mathbf{y},\mathbf{z})\right)$$

- dVAE encoder q_ϕ와 dVAE decoder p_θ를 고정한 상태로 transformer(prior distribution) p_ψ 를 학습
- 120억 개의 parameter sparse transformer 사용
- 256개의 BPE-encoded text token(vocab size = 16384)와 32 X 32 image token(vocab size=8192) 사용
- dVAE encoder로 argmax sampling
- Text와 Image token이 single strea으로 autoregressively model - 모든 text token에 대해 attention

Multimodal Pre-Training: CLIP

Contrastive Language-Image Pre-training

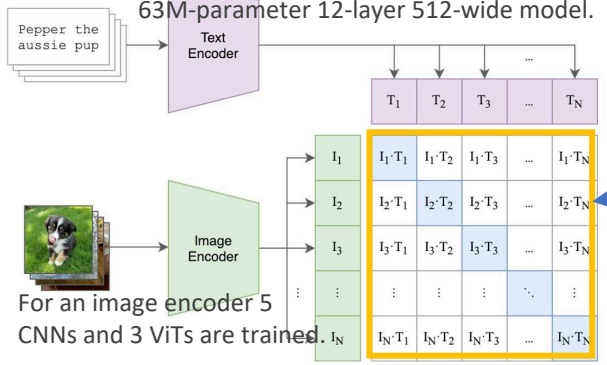
The main idea of CLIP is learning from **natural language supervision**. CLIP learns similarity between image representation and text representation(Joint representation). CLIP has great capability on zero-shot transfer and representation learning in computer vision.

- **Few-Shot Learning**
  - Problem of making predictions based on a limited number of samples.
  - Compared to fully supervised classification, a model is asked to classify test images whose classes are **not in the training set**.
  - Support set is a small set of labeled images to give class information to query images. ***n*-shot** method means that every class in the support set has *n* samples. EX) zero-shot = no support set

Limitations in Computer Vision

- **Zero-Shot Setting**
  - Many approaches use pre-trained models on a large-scale dataset as fine-tuning them on the specific tasks.
  - Without contextual information, however, a vision model cannot classify images of unseen labels in zero-shot setting.
- **Large-scale Dataset**
  - ImageNet-22K and JFT-300M are well-known large datasets with labeled images.
  - Images with qualified natural language titles or descriptions are still insufficient for general vision tasks.

(1) Contrastive pre-training



The text encoder is a Transformer with 63M-parameter 12-layer 512-wide model.

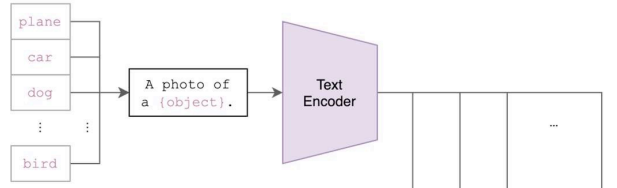
For an image encoder 5 CNNs and 3 ViTs are trained.

An image encoder (CNN or ViT) extracts image features while a text encoder extracts text representations on image-text pairs.

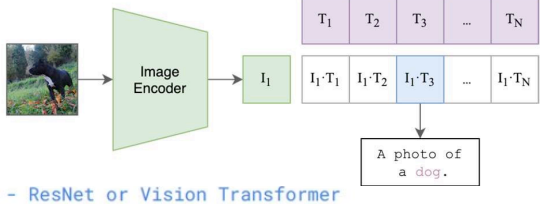
Given *N* pairs, CLIP computes cosine similarity of N^2 possible pairings.

Both encoders are jointly trained using a **contrastive objective**, which maximize cosine similarity of *N* real pairs and minimize that of N^2 incorrect pairs.

(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



At test time, the text encoder synthesizes a linear classifier from descriptions of the target classes.

Using image representations, CLIP predicts classes of the given images with the linear classifier, even in zero-shot setting.

{label} ⟶ "A photo of a {label}."
Prompt engineering
⟶ "A photo of a small {label}".
Ensembling

Compared to using contextless class, **prompt engineering** and ensembling boost zero-shot classification performance.

```
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
```

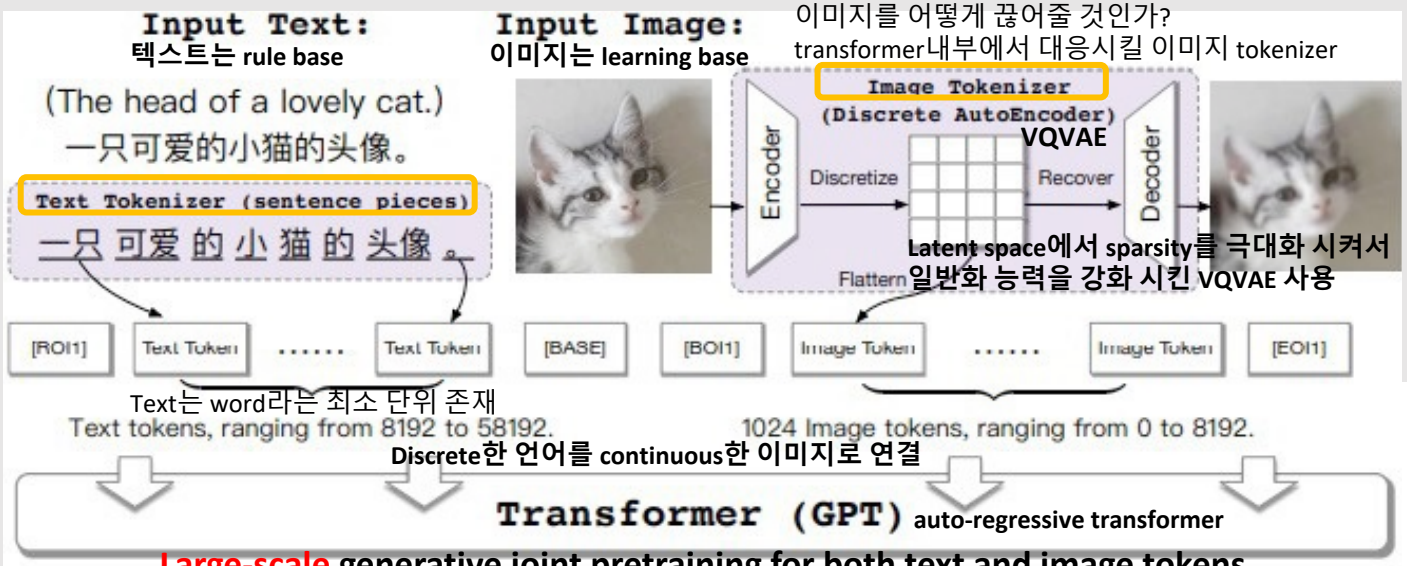
```
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter
```

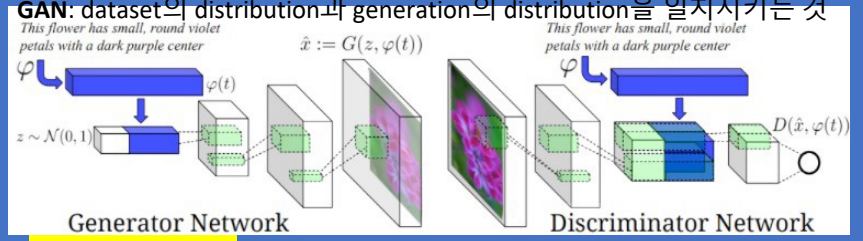
Multimodal Pre-Training: CogView

Combining VQVAE and Transformers for text-to-image generation:
4B parameter Transformer with VQ-VAE tokenizer



Large-scale generative joint pretraining for both text and image tokens.

GAN: dataset의 distribution과 generation의 distribution을 일치시키는 것



GAN-based models can perform reasonable synthesis in simple and domain-specific dataset. BUT, the results on complex and domain-general scenes are far from satisfactory.
=> dataset의 complexity가 올라가면 학습이 너무 어려워진다

1. Disentangle shape, color, gesture and other features from pixels
2. Understand the input text
3. Align objects and features with corresponding words
4. Learn complex distributions to generate the overlapping and composite of different objects and features

달리의 경우 같은 화질의 이미지 혹은 카툰 이미지를 사용했지만, General + 다양한 화질의 이미지 학습 시키고 싶은 cogview
⇒ **Param size + dataset size 둘 다 늘림** => 크면 클수록 모델이 불안정함 = **Text-to-image pretraining is very unstable**
= gradient가 계속 바뀜(큰 값으로 요동침 -> 즉, 미분값의 변동성이 큼) => **여러 학습 테크닉 추가**

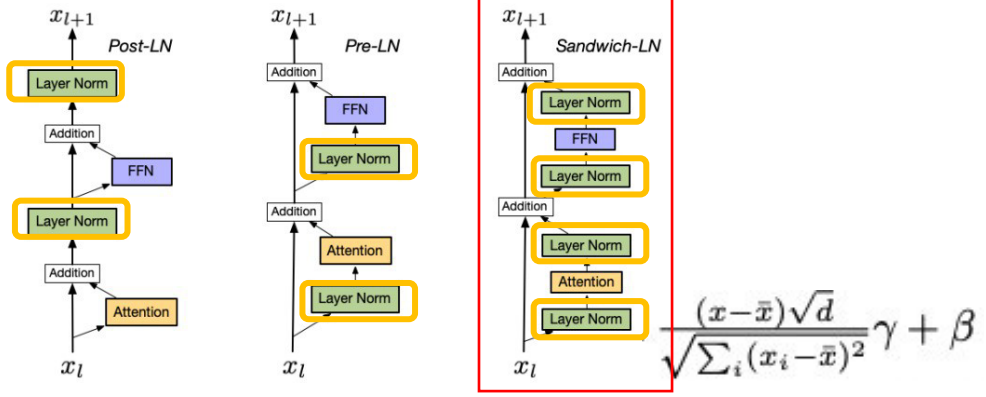
1. Precision Bottleneck Relaxation (PB-Relax)

$$\text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right) = \text{softmax}\left(\left(\frac{Q^T}{\alpha\sqrt{d}}K - \max\left(\frac{Q^T}{\alpha\sqrt{d}}K\right)\right) \times \alpha\right)$$

Similar to Layer norm(LayerNorm(x) = LayerNorm(x/max(x)))

결과값이 바뀌지 않는 선에서 나눠버리자! => gradient를 줄임
Layer norm은 결과값이 바뀌지 않는 선에서
(max 값이 바뀌지 않는 선에서) 최대값으로 나눈다.
Gradient가 튀는 순간이 attention 전에 발생하는 것을 확인
=> attention 상에서 값이 바뀌어도 결과값은 바뀌지 않도록 하는 테크닉을 사용

2. Sandwich LayerNorm (Sandwich-LN)



$$\frac{(x-\bar{x})\sqrt{d}}{\sqrt{\sum_i (x_i-\bar{x})^2}}\gamma + \beta$$

Multimodal Pre-Training: CogView

Finetuning strategies for various downstream tasks
: Super-resolution, self-reranking, style transfer, fashion design

1. Super-resolution

image가 주어지고 문장이 주어지면 관련된 부분의 해상도가 증가하는 Task

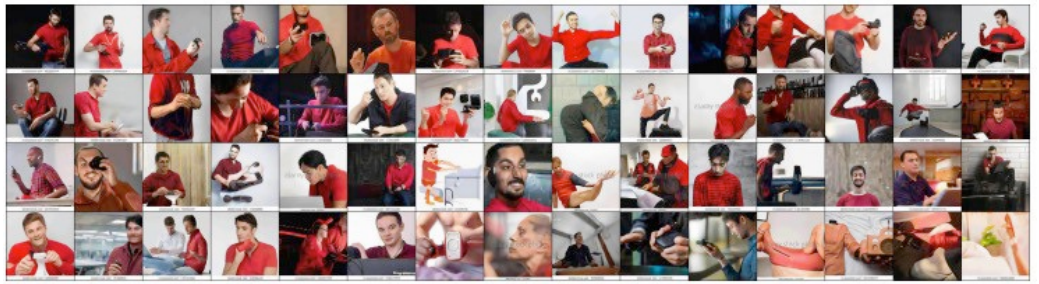
"A tiger is playing football."

달리는 동일 resolution 이미지로만 학습한 반면, cogView는 일부러 다양한 resolution과 많은 domain image에서 image token을 representation하였다.
⇒ 같은 도메인의 resolution 변화 또한 representation space내에 표현이 되어있음
⇒ 이부분을 강조하여 fine tuning

2. Image Captioning and Self-reranking

$$\text{CapLoss}(x,t) = \frac{1}{|t|}\sum_{i=0}^{|t|} -\log p(t_i|x,t_{0:i-1})$$

Exchanging the order of text and image tokens in the input sequence.
➔ Self-reranking



60 generated images for "A man in red shirt is playing video games"
Displayed in the order of CapLoss

3. Style transfer Learning

스타일 별(4개)로 1000개 정도밖에
학습하지 않았는데도 좋은 성능을 보여줌.

"A {object} of {style} style"

4. Industrial Fashion Design

1. Train a domain-specific VQGAN instead of general VQVAE.
: 대신 pretrained cogView는 사용

2. Decrease the number of parameters and increase the length of sequences for a higher resolution.

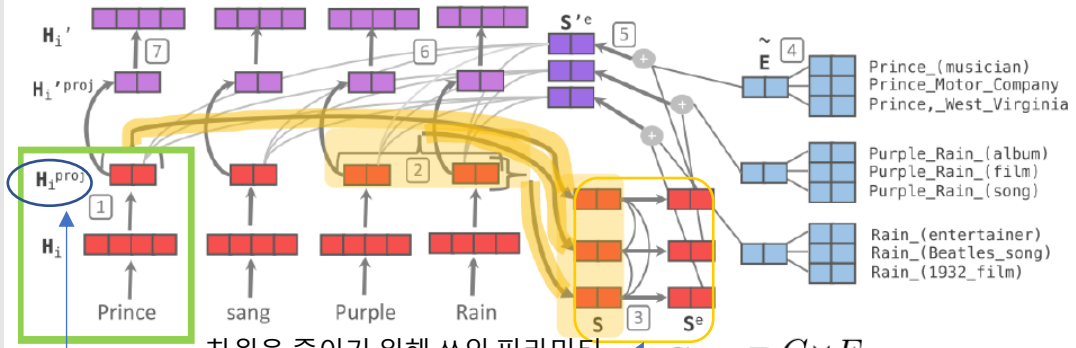
Knowledge-Enhanced Pre-Training: **KnowBERT**

Knowledge Attention and Recontextualization (KAR) mechanism

별도로 존재하는 도메인 지식이 반영된 entity embeddings 를 BERT wordpiece tokenizer layer 에 통합시키는 방법을 제안, Bert objective에 관련 loss를 추가

[1] Mention-span(entity) representations

Entity 는 '**Prince**', '**Purple Rain**', '**Rain**' 총 3 개



$$\mathbf{H}_i^{\text{proj}} = \mathbf{H}_i \mathbf{W}_1^{\text{proj}} + \mathbf{b}_1^{\text{proj}}$$

차원을 줄이기 위해 쓰인 파라미터

$$\mathbf{S} \in \mathbb{R}^{C \times E}$$

Entity embedding의 차원에 맞추어 projection(512-> E)

Self-attentive span pooling (Lee et al., 2017) 방식을 이용하여 mention span(=entity) 에 대한 모든 WordPieces 를 pooling
=> entity-entity co-occurrence 반영 효과

[2, 3] Entity linker, Knowledge enhanced entity-span representations

$$\mathbf{s}_m'^e = \mathbf{s}_m^e + \tilde{\mathbf{e}}_m,$$

$$\psi_{mk} = \text{MLP}(p_{mk}, \mathbf{s}_m^e \cdot \mathbf{e}_{mk}),$$

Two-layer MLP (100 hidden dimensions) 를 이용하여 **score 계산**

- mention-span vector s_m^e
- embedding from Knowledge Base e_{mk}
- prior probability p_{mk}
  - 등장 빈도수에 기반한 확률

$$\tilde{\psi}_{mk} = \begin{cases} \dfrac{\exp(\psi_{mk})}{\sum_{\psi_{mk} \geq \delta} \exp(\psi_{mk})}, & \psi_{mk} \geq \delta \\ 0, & \psi_{mk} < \delta. \end{cases}$$

score가 threshold 아래이면 0 으로 취급 , threshold 를 넘기면 normalize 하여 사용

$$\tilde{\mathbf{e}}_m = \sum_k \tilde{\psi}_{mk} \mathbf{e}_{mk}.$$

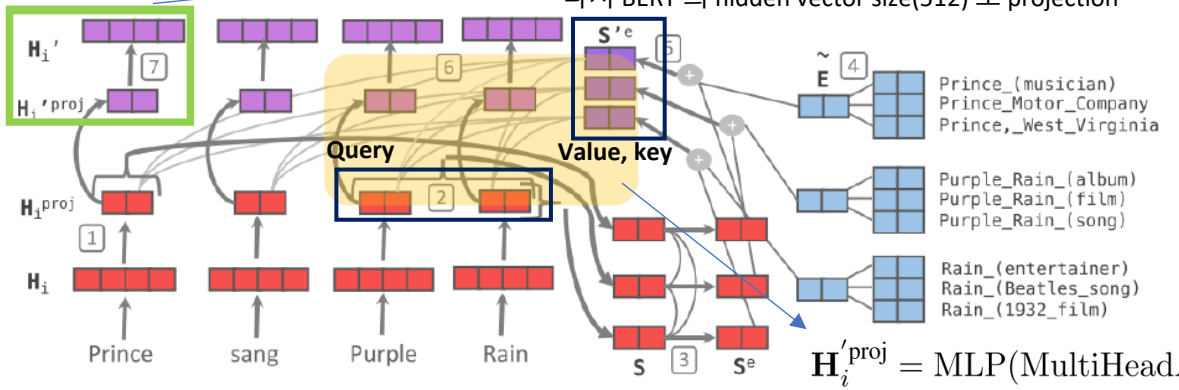
$$\mathbf{S}^e = \text{TransformerBlock}(\mathbf{S}).$$

inverse matrix

[4] Recontextualization

$$\mathbf{H}'_i = \mathbf{H}'^{\text{proj}}_i \mathbf{W}_2^{\text{proj}} + \mathbf{b}_2^{\text{proj}} + \mathbf{H}_i$$

다시 BERT 의 hidden vector size(512) 로 projection



$$\mathbf{H}_i'^{\text{proj}} = \text{MLP}(\text{MultiHeadAttn}(\mathbf{H}_i^{\text{proj}}, \mathbf{S}'^e, \mathbf{S}'^e))$$

query key value

Algorithm 1: KnowBert training method

Input: Pretrained BERT and J KBs
Output: KnowBert
for $j = 1 \ldots J$ **do**
  Compute entity embeddings for KB_j
  if EL supervision available **then**
    Freeze all network parameters except those in (Eq. 1–3)
    Train to convergence using (Eq. 4) or (Eq. 5)
  end
  Initialize $\mathbf{W}_2^{\text{proj}}$ as $(\mathbf{W}_1^{\text{proj}})^{-1}$
  Unfreeze all parameters except entity embeddings
  Minimize $\mathcal{L}_{\text{KnowBert}} = \mathcal{L}_{\text{BERT}} + \sum_{i=1}^{j} \mathcal{L}_{\text{EL}_i}$
end

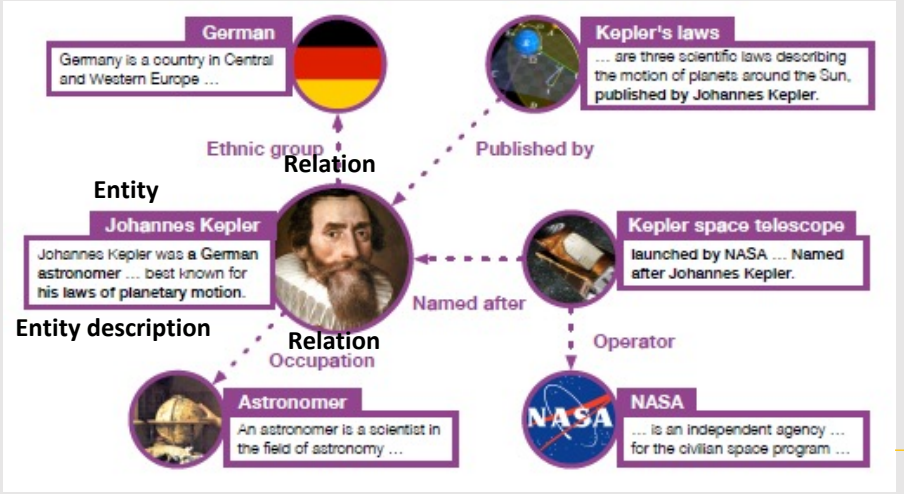
Entity linking **objectives**:
Log likelihood 식과 max margin 식 (g 는 정답 인덱스를 의미)

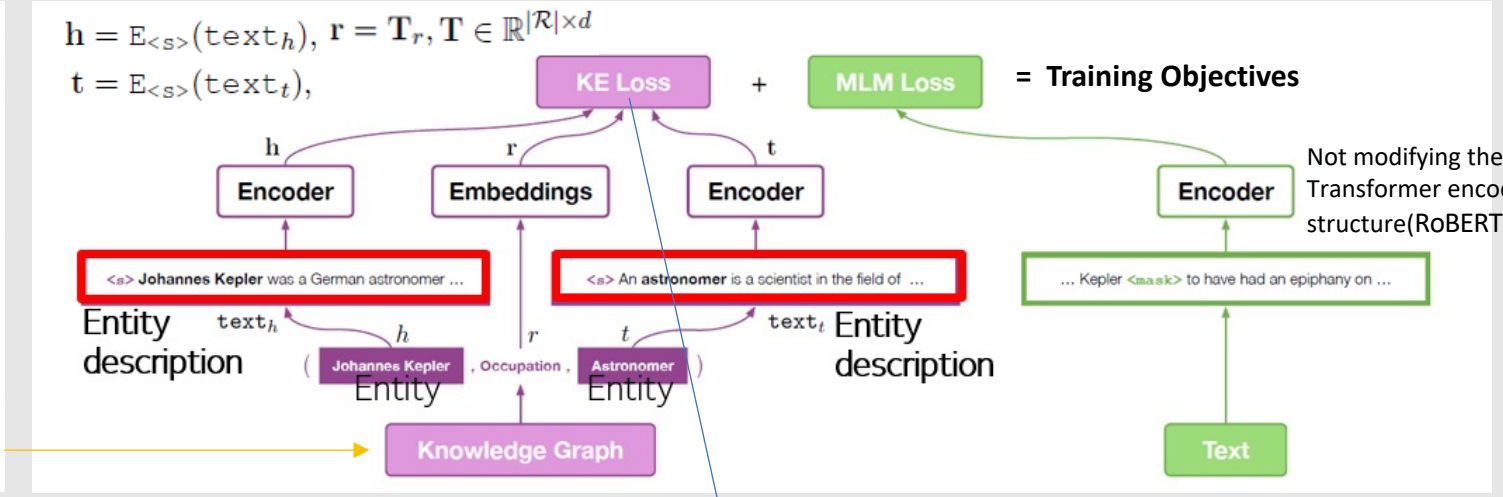
$$\mathcal{L}_{\text{EL}} = -\sum_m \log \left(\frac{\exp(\psi_{mg})}{\sum_k \exp(\psi_{mk})} \right)$$

$$\mathcal{L}_{\text{EL}} = \max(0, \gamma - \psi_{mg}) + \sum_{e_{mk} \neq e_{mg}} \max(0, \gamma + \psi_{mk}),$$

Knowledge-Enhanced Pre-Training: **KEPLER**

A Unified Model for **K**nowledge **E**mbedding and **P**re-trained **L**anguag**E** **R**epresentation

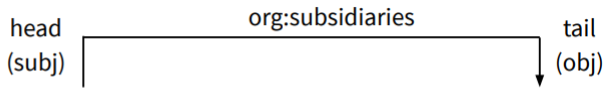




Downstream Tasks

related to Relational Knowledge

TACRED



[...] included Aerolineas's domestic subsidiary, Austral.

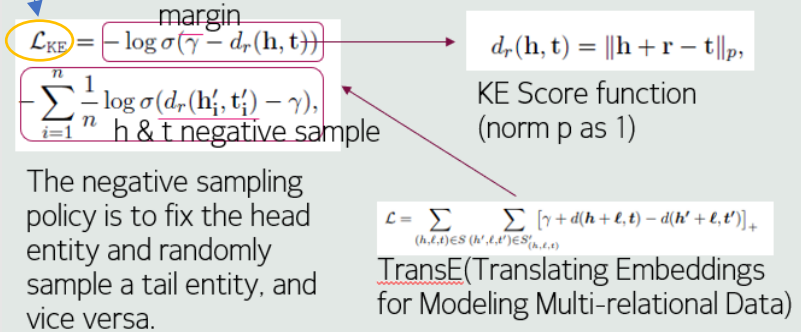
Sentence with Target Entity	Entity Types
During the Inca Empire, {the Inti Raymi} was the most important of four ceremonies celebrated in Cusco.	event, festival, **ritual, custom, ceremony, party, celebration**
{They} have been asked to appear in court to face the charge.	person, **accused, suspect, defendant**
Ban praised Rwanda's commitment to the UN and its role in {peacemaking operations}.	event, **plan, mission, action**

Open Entity

FewRel (Few-Shot Relation Classification Dataset)

Supporting Set	
(A) capital_of	(1) *London* is the capital of *the U.K.* (2) *Washington* is the capital of *the U.S.A.*
(B) member_of	(1) *Newton* served as the president of *the Royal Society*. (2) *Leibniz* was a member of *the Prussian Academy of Sciences*.
(C) birth_name	(1) *Samuel Langhorne Clemens*, better known by his pen name *Mark Twain*, was an American writer. (2) *Alexei Maximovich Peshkov*, primarily known as *Maxim Gorky*, was a Russian and Soviet writer.
Test Instance	
(A) or (B) or (C)	*Euler* was elected a foreign member of *the Royal Swedish Academy of Sciences*.

Table 1: An example for a 3 way 2 shot scenario. Different colors indicate different entities, blue for head entity, and red for tail entity.



Knowledge-Enhanced Pre-Training: **KGLM**

Knowledge Graph Language Model

Language model with mechanisms for **selecting and copying information from an external knowledge graph** (KG), using LSTM

KGLM computes $P(x_t, \mathcal{E}_t \mid x_{<t}, \mathcal{E}_{<t})$ **Marginalizing out the local knowledge graph** to compute the probability of the tokens(inference),

Entity(t) Entity(<t)

$$P(x) = \Sigma_{\mathcal{E}}\, p(x,\ \mathcal{E})$$

Using a single-layer softmax over **hidden state of LSTM** to predict one of {new, related, Ø }

Input Sequence

Super Mario Land is a game published by Nintendo .

t_t

Decide the **type of token**

1) Related entity (in the local KG)

Relation to Existing Entity

parent from local entities p_t

platform game
side-scrolling game
Super Mario Land
...

e_t

Super Mario Land r_t Nintendo

SELF PUBLISHER GENRE PLATFORM PUB.DATE

Game Boy
platform game
1989

x_t

standard vocabulary

a
the
company
dog
...

mimicking the **TransE**
: 지식 정보 그래프 정보를 embedding 공간으로 옮겨서 해석 가능하게 만드는 방법
=> head entity 와 relation 벡터 합이 tail entity가 유사하게 만드는 것

$$p(e_t) = \text{softmax}(\mathbf{v}_e \cdot (\mathbf{h}_{t,p} + \mathbf{h}_{t,r}))$$

aliases of e_t

Kabushiki
Koppai
Nintendo
...

2) New entity (not in the **local** KG)

Mention of a New Entity

pick from all entities e_t

AAA Inc.
...
Sony Inc.
...
Zzyzx, CA

Distribution over standard vocabulary and aliases of e_t

3) Not an entity

Not an Entity Mention

$e_t = \emptyset$

Distribution over standard vocabulary

Initially, the graph is empty

Building a local KG by KGLM

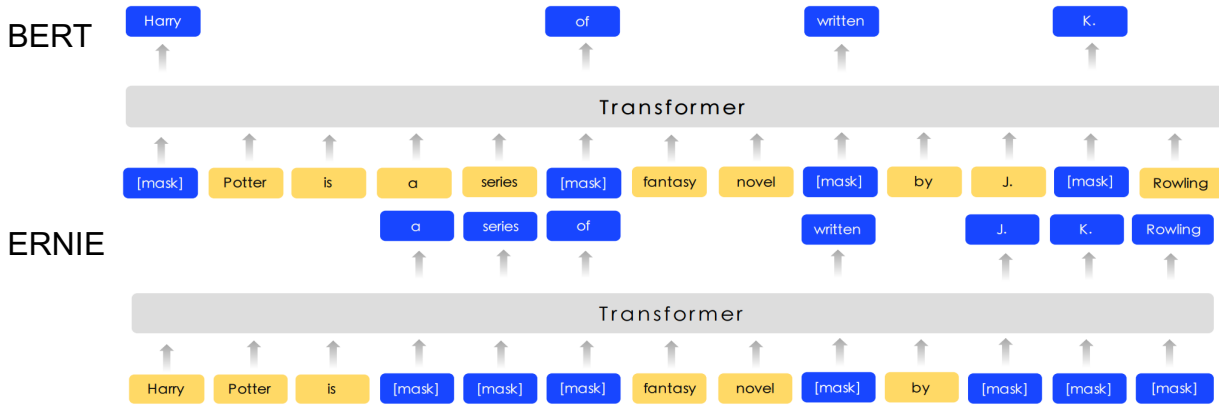
ERNIE: Enhanced Representation through Knowledge Integration, 2019

https://arxiv.org/abs/1904.09223

Motive	The vast majority of Language representation pre-training studies model the representations by predicting the missing word only through the contexts. These works **do not consider the prior knowledge** in the sentence. It is intuitive that if the model learns more about prior knowledge, the model can obtain more reliable language representation.	***Keywords***	ERNIE 1.0, Language representation model, Knowledge masking strategy, Basic-Level Masking, Phrase-Level Masking, Entity-Level Masking, Dialogue embedding

Introduction

- ERNIE (enhanced representation through knowledge integration) by using knowledge masking strategies. In addition to basic masking strategy, we use two kinds of **knowledge strategies: phrase-level strategy and entity-level strategy**.
- Instead of adding the knowledge embedding directly, ERNIE implicitly learned the information about knowledge and longer semantic dependency
- ERNIE is pre-trained on heterogeneous Chinese data, and then applied to 5 Chinese NLP tasks.



Model

- Using a multi-layer Transformer as basic encoder like previous pre-training model such as GPT, BERT and XLM
- For Chinese corpus, we add spaces around every character in the CJK Unicode range and use the WordPiece to tokenize
- The first token of every sequence is the special classification embedding([CLS]).
- A multi-stage **knowledge masking strategy** to integrate phrase and entity level knowledge into the Language representation

(1) **Basic-Level Masking**: the basic language unit is word, and for Chinese, the basic language unit is Chinese Character

(2) **Phrase-Level Masking**: Phrase is a small group of words or characters together acting as a conceptual unit.

(3) **Entity-Level Masking**: Name entities contain persons, locations, organizations, products, etc. First analyzing the named entities in a sentence, and then masking and predicting all slots in the entities

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]

ERNIE: Enhanced Representation through Knowledge Integration, 2019

https://arxiv.org/abs/1904.09223

Model

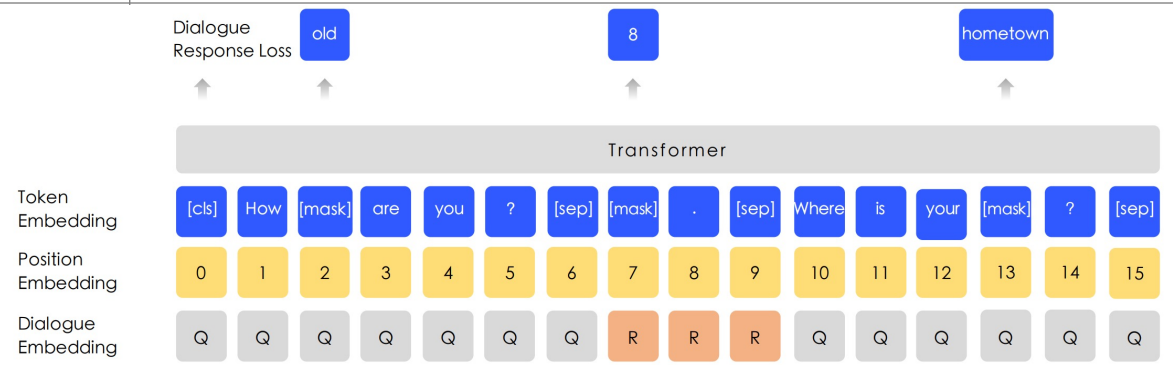
- ERNIE was chosen to have the same model size as BERT-base for comparison purposes.
- 12 encoder layers, 768 hidden units and 12 attention heads

Dataset

- Mixed corpus: Chinese Wikepedia (21M), Baidu Baike(51M), Baidu news(47M), Baidu Tieba(54M)
- Baidu Baike contains encyclopedia articles written in formal languages,
- Baidu news provides the latest information about movie names, actor names, football team names, etc.
- Baidu Tieba (for DLM) is an open discussion forum like Reddits

Model (DLM)

- **DLM (Dialogue Language Model)**
  . introducing **dialogue embedding** to identify the roles in the dialogue
  . ERNIE's dialogue embedding plays the same roles as token type embedding in BERT, except that ERNIE can also represent multi-turn conversations (e.g., QRQ, QRR, QQR, Q for "Query", R "Response").
  . Like MLM in BERT, masks are applied to enforce the model to predict missing words conditioned on both query and response.
  . The model architecture of DLM task is compatible with that of the MLM task, thus it is pre-trained alternatively with the MLM task.



Results

- The test results on 5 Chinese NLP tasks can be seen that ERNIE outperforms BERT on all tasks, creating new state-of-the-art results on these Chinese NLP tasks.

Task	Metrics	Bert		ERNIE	
		dev	test	dev	test
XNLI	accuracy	78.1	77.2	79.9 (+1.8)	78.4 (+1.2)
LCQMC	accuracy	88.8	87.0	89.7 (+0.9)	87.4 (+0.4)
MSRA-NER	F1	94.0	92.6	95.0 (+1.0)	93.8 (+1.2)
ChnSentiCorp	accuracy	94.6	94.3	95.2 (+0.6)	95.4 (+1.1)
nlpcc-dbqa	mrr	94.7	94.6	95.0 (+0.3)	95.1 (+0.5)
	F1	80.7	80.8	82.3 (+1.6)	82.7 (+1.9)

- Ablation Studies: Adding phrase-level mask to the baseline word-level mask can improve the performance of the mode, 0.7%/1.0% of improvement in develop/test accuracy is achieved on this DLM task

pre-train dataset size	mask strategy	dev Accuracy	test Accuracy
10% of all	word-level(chinese character)	77.7%	76.8%
10% of all	word-level&phrase-level	78.3%	77.3%
10% of all	word-level&phrase-leve&entity-level	78.7%	77.6%
all	word-level&phrase-level&entity-level	79.9 %	78.4%

corpus proportion(10% of all training data)	dev Accuracy	test Accuracy
Baike(100%)	76.5%	75.9%
Baike(84%) / news(16%)	77.0%	75.8%
Baike(71.2%)/ news(13%)/ forum Dialogue(15.7%)	77.7%	76.8%

Code

- From the paper: https://github.com/PaddlePaddle/ERNIE

ERNIE 2.0: A Continual Pre-training Framework for Language Understanding, 2019

https://arxiv.org/abs/1907.12412

Motive	Current pre-training procedures usually focus on training the model with several simple tasks to grasp the co-occurrence of words or sentences. However, besides co-occurring information, there exists other valuable lexical, syntactic and semantic information in training corpora.	***Keywords***	ERNIE 2.0, Continual pre-training framework, Continual multi-task learning, Continual learning,

Introduction

- a continual pre-training framework named ERNIE 2.0 which incrementally builds pre-training tasks and then learn pre-trained models on these constructed tasks via **continual multi-task learning**.
- ERNIE 2.0 model is to capture lexical, syntactic and semantic aspects of information in the training data.

Model

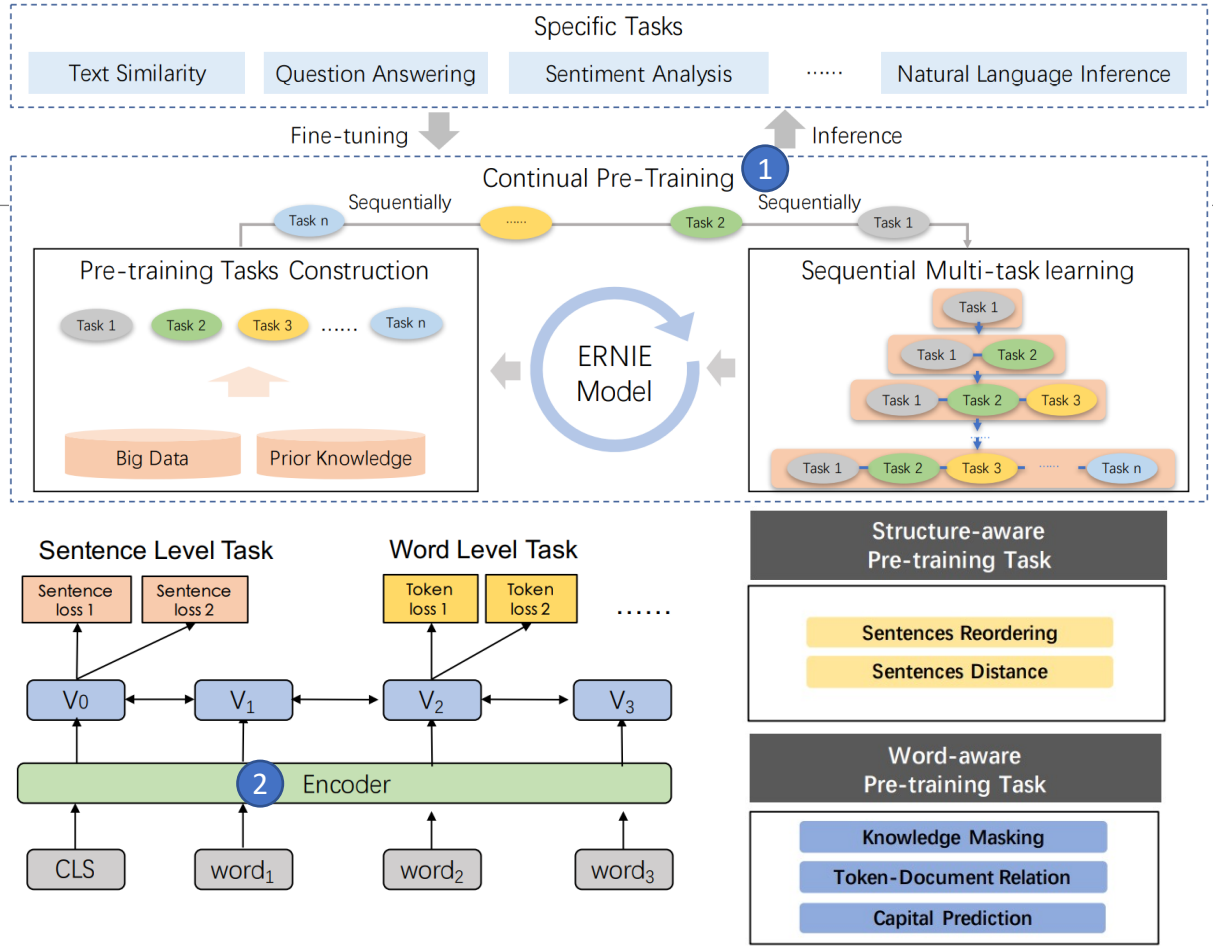
- ERNIE 2.0: instead of training with a small number of pre-training objectives, it could constantly introduce a large variety of pre-training tasks to help the model efficiently learn the lexical, syntactic and semantic representations
- Continual Pre-training

  (1) Pre-training Tasks Construction: Constructing different kinds of tasks at each time that the learned parameters encodes the previously learned knowledge, Allocating each task N training iterations

  (2) the architecture of continual multi-task learning in each stage contains a series of **shared text encoding layers** to encode contextual information, which can be customized by using recurrent neural networks or a deep Transformer consisting of stacked self-attention layers

  (3) Loss functions: Token-level loss, Sentence-level loss
- Fine-tuning for Application Tasks

  : the pre-trained model can be adapted to different language understanding tasks, such as question answering, natural language inference



ERNIE 2.0: A Continual Pre-training Framework for Language Understanding, 2019

https://arxiv.org/abs/1907.12412

Model

- Transformer Encoder: Using a multi-layer Transformer as the basic encoder like other pre-training models
- Task Embedding: The model feeds task embedding to represent the characteristic of different tasks with an id ranging from 0 to N.
- Using the same model settings of transformer as BERT
  Base: 12 layers, 12 self-attention heads, 768-dimensional of hidden size
  Large: 24 layers, 16 self-attention heads, 1024-dimensional of hidden size

Dataset

- Pre-training

Corpus \ Task	Token-Level Loss			Sentence-Level Loss			
	Knowledge Masking	Capital Prediction	Token-Document Relation	Sentence Reordering	Sentence Distance	Discourse Relation	IR Relevanc
Encyclopedia	✓	✓	✓	✓	✓	×	×
BookCorpus	✓	✓	✓	✓	✓	×	×
News	✓	✓	✓	✓	✓	×	×
Dialog	✓	✓	✓	✓	✓	×	×
IR Relevance Data	×	×	×	×	×	×	✓
Discourse Relation Data	×	×	×	×	×	✓	×

Corpus Type	English(#tokens)	Chinese(#tokens)
Encyclopedia	2021M	7378M
BookCorpus	805M	-
News	-	1478M
Dialog	4908M	522M
IR Relevance Data	-	4500M
Discourse Relation Data	171M	1110M

- Fine-tuning: GLUE dataset for English (Left), Chinese task (Right)

Task	*BASE*		
	Epoch	Learning Rate	Batch Size
CoLA	3	3e-5	64
SST-2	4	2e-5	256
MRPC	4	3e-5	32
STS-B	3	5e-5	128
QQP	3	3e-5	256
MNLI	3	3e-5	512
QNLI	4	2e-5	256
RTE	4	2e-5	4
WNLI	4	2e-5	8

Task	*BASE*		
	Epoch	Learning Rate	Batch Size
CMRC 2018	2	3e-5	64
DRCD	2	5e-5	64
DuReader	2	5e-5	64
MSRA-NER	6	5e-5	16
XNLI	3	1e-4	512
ChnSentiCorp	10	5e-5	24
LCQMC	3	2e-5	32
BQ Corpus	3	3e-5	64
NLPCC-DBQA	3	2e-5	64

Results

- The results on GLUE benchmark, where the results on dev set are the median of five runs and the results on test set are scored by the GLUE evaluation server
- ERNIE 2.0$_{BASE}$ outperforms BERT$_{BASE}$ on all of the 10 tasks and obtains a score of 80.6.
- ERNIE 2.0$_{LARGE}$ consistently outperforms BERT$_{LARGE}$ and XLNet$_{LARGE}$ on most of the tasks except MNLI-m.

Task(Metrics)	*BASE model*		*LARGE model*				
	Test		Dev			Test	
	BERT	ERNIE 2.0	BERT	XLNet	ERNIE 2.0	BERT	ERNIE 2.0
CoLA (Matthew Corr.)	52.1	**55.2**	60.6	63.6	**65.4**	60.5	**63.5**
SST-2 (Accuracy)	93.5	**95.0**	93.2	95.6	**96.0**	94.9	**95.6**
MRPC (Accurary/F1)	84.8/88.9	**86.1/89.9**	88.0/-	89.2/-	**89.7/-**	85.4/89.3	**87.4/90.2**
STS-B (Pearson Corr./Spearman Corr.)	87.1/85.8	**87.6/86.5**	90.0/-	91.8/-	**92.3/-**	87.6/86.5	**91.2/90.6**
QQP (Accuracy/F1)	89.2/71.2	**89.8/73.2**	91.3/-	91.8/-	**92.5/-**	89.3/72.1	**90.1/73.8**
MNLI-m/mm (Accuracy)	84.6/83.4	**86.1/85.5**	86.6/-	**89.8/-**	89.1/-	86.7/85.9	**88.7/88.8**
QNLI (Accuracy)	90.5	**92.9**	92.3	93.9	**94.3**	92.7	**94.6**
RTE (Accuracy)	66.4	**74.8**	70.4	83.8	**85.2**	70.1	**80.2**
WNLI (Accuracy)	**65.1**	**65.1**	-	-	-	65.1	**67.8**
AX(Matthew Corr.)	34.2	**37.4**	-	-	-	39.6	**48.0**
Score	78.3	**80.6**	-	-	-	80.5	**83.6**

Code

- From the paper: https://github.com/PaddlePaddle/ERNIE

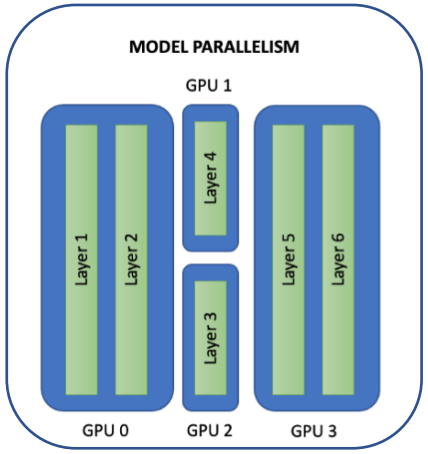
03 Improving Computational Efficiency

Improving computational efficiency

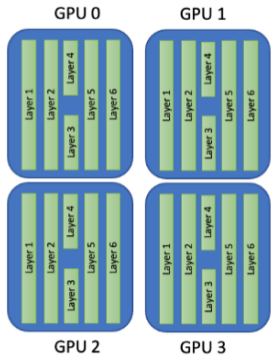
- System-level optimization
  - Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism
- Efficient pre-training
  - Train No Evil: Selective Masking for Task-Guided Pre-Training
  - Accelerating Training of Transformer-Based Language Models with Progressive Layer Dropping
- Model compression
  - ALBERT: A Lite BERT for Self-supervised Learning of Language Representations
  - Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning
  - Knowledge Distillation: A Survey
  - DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter
  - MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers

System-Level Optimization: Megatron-LM

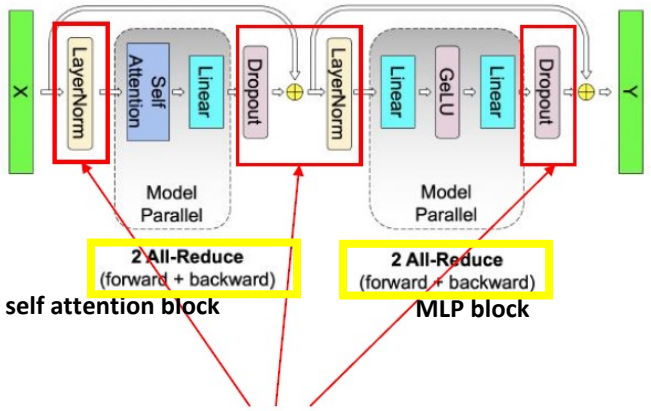
Training Multi Billion Parameter **Language Models** Using **Model Parallelism**



Parameter sharing

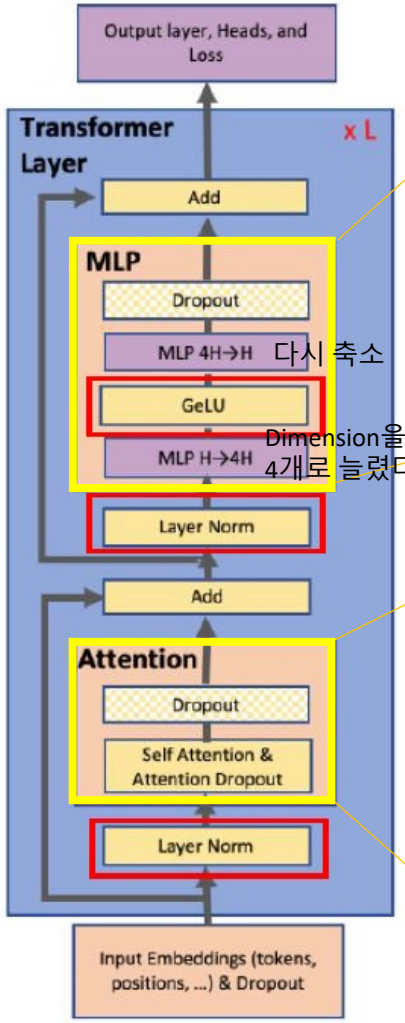


layer를 넘어가면서 데이터 통신이 필요
=> 모델을 어떻게 자르는 것에 따라 통신 overhead가 많을 수도 있음
초거대 모델 ex. GPT3(param수가 많아지면 성능이 좋아진다) 추세
=> 그러나 gpu 메모리 부족!(cpu처럼 대응이 불가능)



Duplicate the computation to reduce communication

Self attention block 과 MLP block을 통해 model parallelism 구현

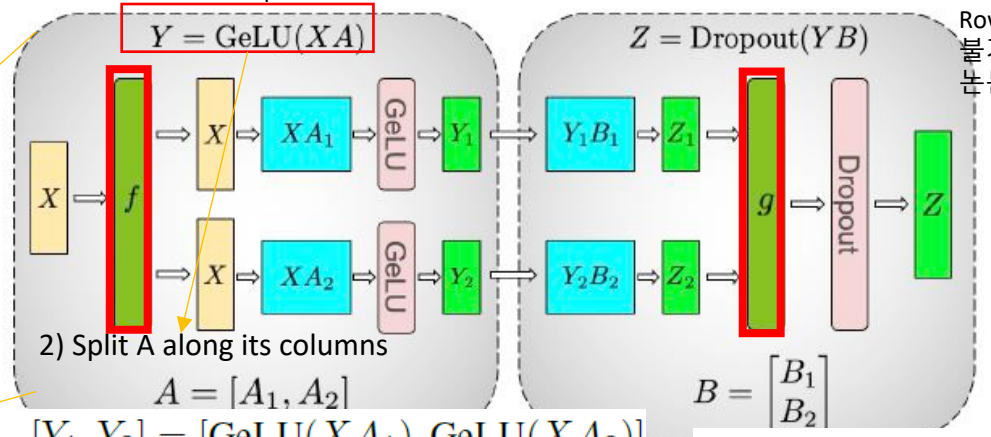


1) Split A along its rows

$$X = [X_1, X_2],\ A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \qquad Y = \mathrm{GeLU}(X_1A_1 + X_2A_2)$$

$$\mathrm{GeLU}(X_1A_1 + X_2A_2) \neq \mathrm{GeLU}(X_1A_1) + \mathrm{GeLU}(X_2A_2)$$

Matrix 곱셈이 parallelize



Row 단위로 하면 양립 불가이기 때문에 논문에선 concat (2)

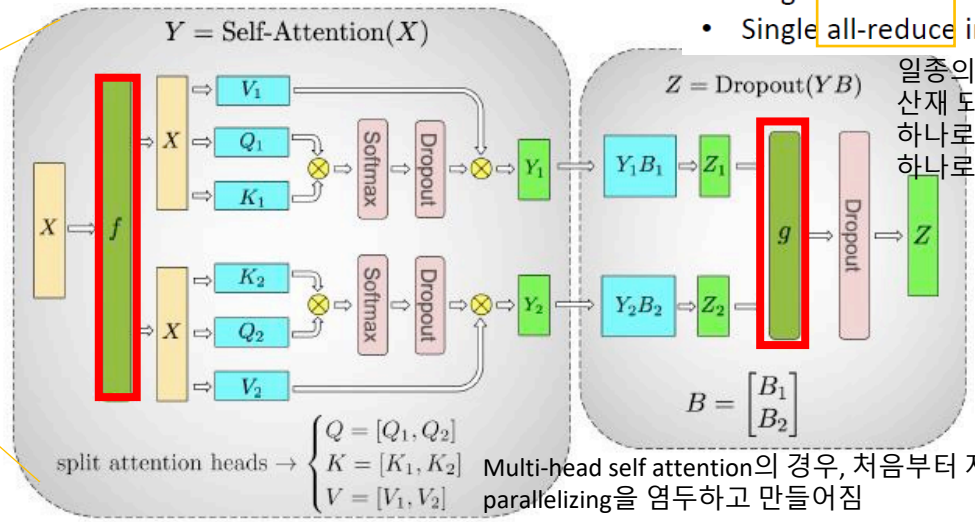
2) Split A along its columns

$$A = [A_1, A_2] \qquad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

$$[Y_1, Y_2] = [\mathrm{GeLU}(XA_1), \mathrm{GeLU}(XA_2)]$$

- Single all-reduce in forward pass (g)
- Single all-reduce in backward pass (f)

일종의 통신인데, device로 산재 되어 있는 데이터를 하나로 모아서 평균 내거나, 하나로 더하여 넘긴다



Multi-head self attention의 경우, 처음부터 자체가 parallelizing을 염두하고 만들어짐

Efficient Pre-Training: **Progressive layer dropout**

Training the **transformer-based language models** is expensive (in terms of training time/memory/cost). Ex. Batch 사이즈를 키우면 memory cost가 크다
Progressive **layer dropping** (PLD) is proposed to **reduce the training time**.

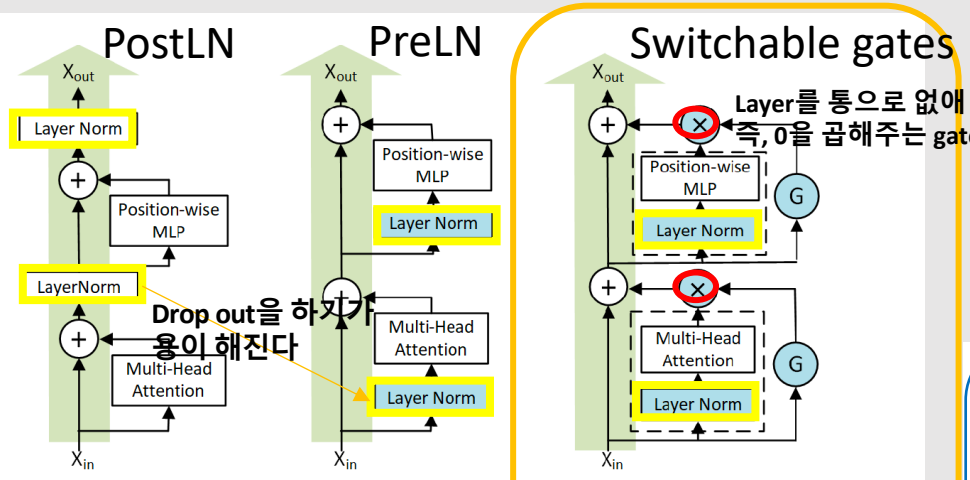
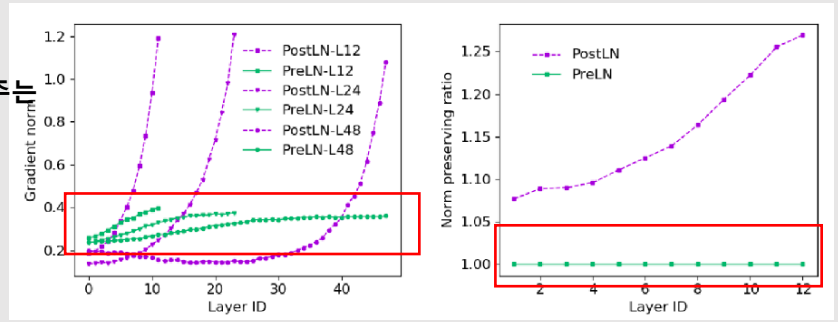
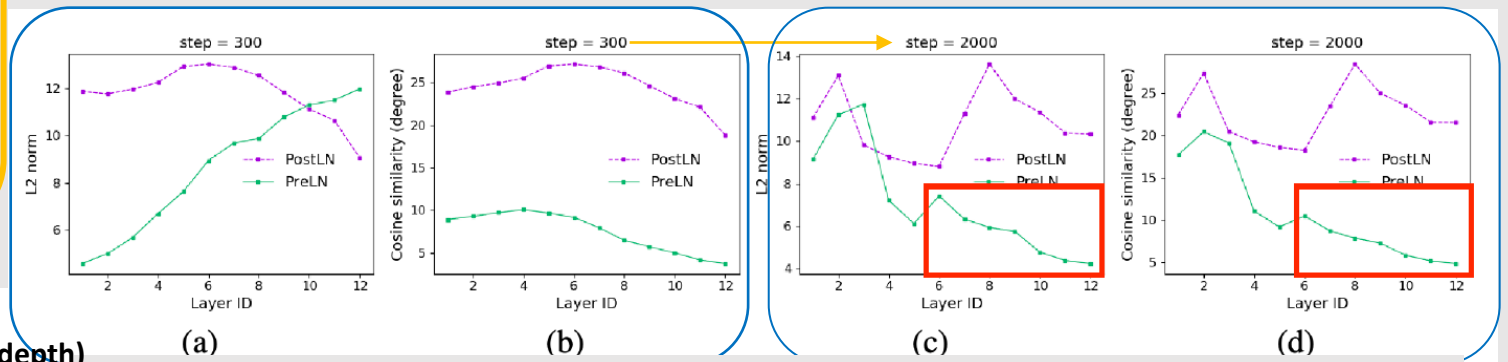


Figure 5: Transformer variants, showing a single layer block.



PreLN preserves norm of representation & gradient, more stable/robust for the random layer dropping.
전체적으로 layer가 균일하게 weight 증가
=> 훈련 용이



Input and output embeddings for BERT(Input과 output을 차이가 어떻게 되는지를 표시)
PreLN의 경우, step수 가 많아지면 Layer가 뒤로 가면 갈 수록 input을 바꾸는 정도가 적어진다(뒤로 갈수록 변화량이 크지 않다) => 뒤쪽 layer dropout을 많이 해도 앞쪽 layer는 덜 damaging 받는다.

Algorithm 1 **Progressive_Layer_Dropping**

Input: $keep\_prob\ \bar{\theta}$
InitBERT($switchable\_transformer\_block$)
$\gamma \leftarrow \frac{100}{T}$
for t ← 1 to T **do**
  $\theta_t \leftarrow (1-\bar{\theta})exp(-\gamma \cdot t) + \bar{\theta}$
  step ← $\frac{1-\theta_t}{L}$
  $p \leftarrow 1$
  for l ← 1 to L **do**
    action ∼ Bernoulli(p)
    if action == 0 **then**
      $x_{i+1} \leftarrow x_i$
    else
      $x_i^{'} \leftarrow x_i + f_{ATTN}(f_{LN}(x_i)) \times \frac{1}{p}$
      $x_{i+1} \leftarrow x_i^{'} + f_{FFN}(f_{LN}(x_i^{'})) \times \frac{1}{p}$
    $x_i \leftarrow x_{i+1}$
    p ← p - step
  Y ← $output\_layer(x_L)$
  loss ← $loss\_fn(\bar{Y}, Y)$
  backward(loss)

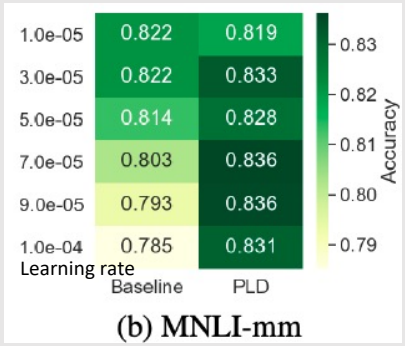
Schedule dropout (w.r.t. time and depth)

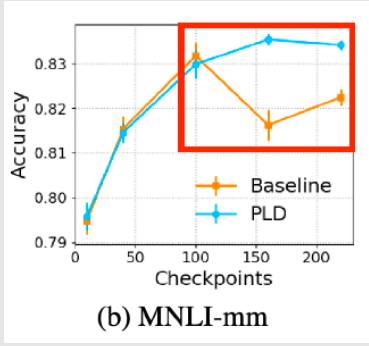
Reduce expected ratio of retained layers as training progresses.

step수가 많아질수록 뒤쪽 layer dropout rate를 높이는 방향(progressive scheduling)
Reduce the retaining rate for upper layers.

나중에 0.5로 수렴
When $L = 12$ and $\bar{\theta} = 0.5$, the effective number of layers is $\bar{L} \approx 9$.
약 25% 연산량
절약(forward-backward)

PLD is **robust to high learning rate**, faster convergence (2.8x speedup)
Ratio of retained layers 0.5~0.9 is appropriate. PLD also benefits downstream task (GLUE) performance & training speed. PLD takes a more **robust optimization path toward the optimum**.

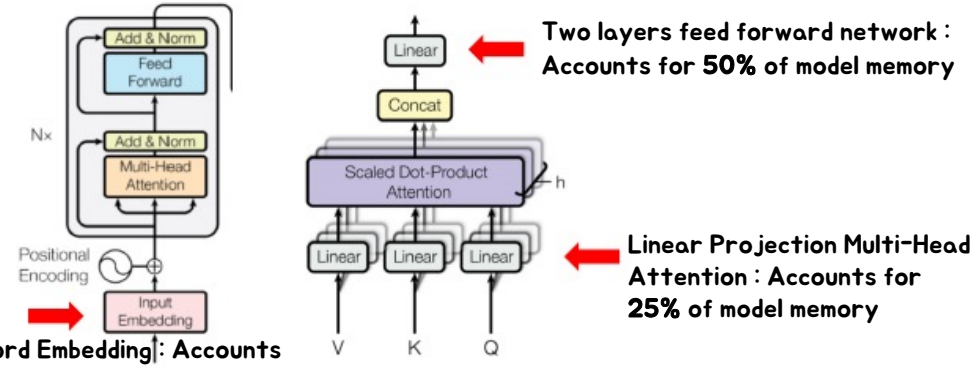




Model Compression: **Compressing BERT**

Explore weight pruning for BERT and ask: how does compression during pre-training affect transfer learning?

Pruning BERT

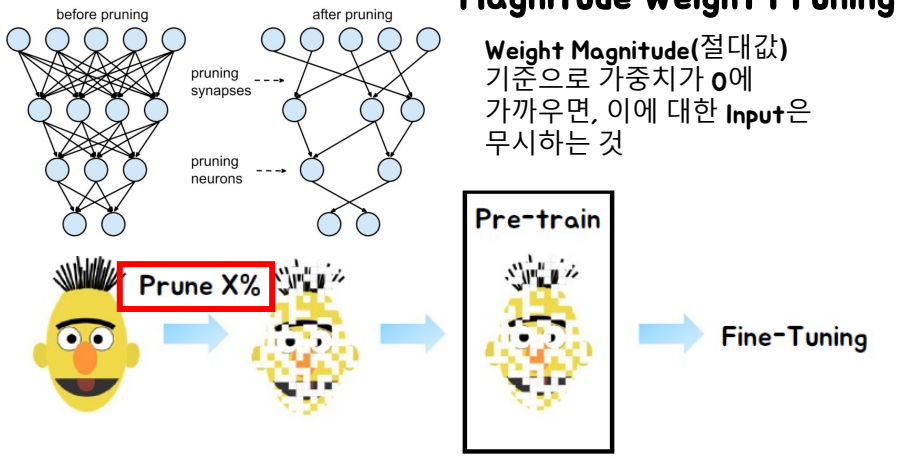


Two layers feed forward network : Accounts for **50%** of model memory

Linear Projection Multi-Head Attention : Accounts for **25%** of model memory

Word Embedding : Accounts for **20%** memory of model memory

Magnitude Weight Pruning



Weight Magnitude(절대값) 기준으로 가중치가 0에 가까우면, 이에 대한 Input은 무시하는 것

We conclude that BERT can be **pruned once during pre-training** rather than separately for each task without affecting performance.

Low levels of pruning (30-40%) do not affect pre-training loss or transfer to downstream tasks at all. (30퍼센트까지는 downstream task accuracy에 영향을 주지 않음을 알 수 있음. Pre-trained loss도 30퍼센트까지는 영향을 주지 않음.)

Medium levels of pruning increase the pre-training loss and prevent useful pre-training information from being transferred to downstream tasks.

Why does pruning affect transfer Learning?

- 가설1 : Regularization;
  - Weight pruning prevents certain weights from changing during fine-tuning, preventing adaption
- 가설2 : Information Deletion
  - Weight pruning deletes valuable information stored in the weights by setting them to 0
  - This information could not be recovered by training on the downstream data alone

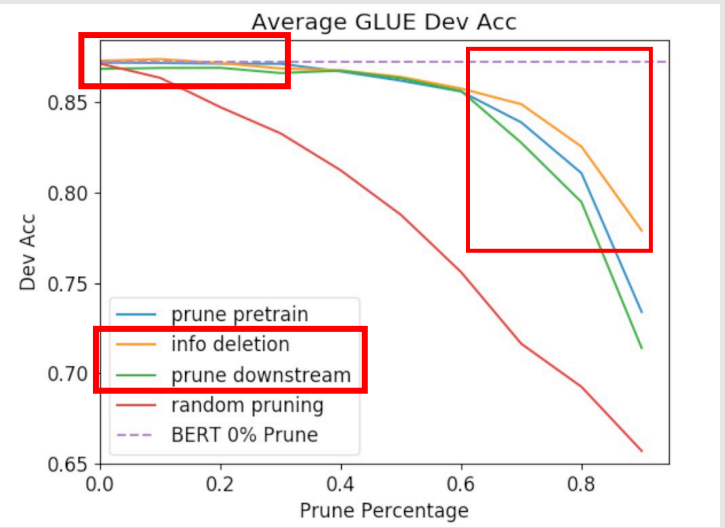
Re-initialize pruned weights to 0 then fine-tune. This will eliminate the regularizing effects of hypothesis.

- Until 60% of sparsity, any decrease in performance can be attributed to information deletion.
- Past the point(60%), an additional penalties encourage for over regularizing the model

Is it better to prune before/after fine-tuning?

Fine tuning does not improve credibility when using magnitude weight Pruning. Even if it continues fine tuning for much longer, it will not change which magnitude weight pruning selects for pruning.

Pruning before fine-tuning is fine



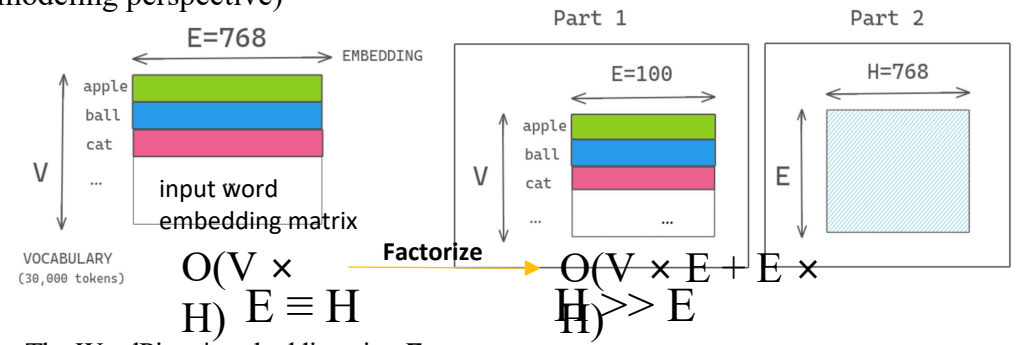
Model Compression: **ALBERT**

A LITE **BERT** FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

ALBERT(Lan et al., 2019) is another important variant of BERT, which provides several interesting observations on **reducing parameters**. As a sacrifice to its space efficiency, **ALBERT has a slower fine-tuning and inference speed**.

1. **Factorized** embedding parameterization

The power of BERT-like representations comes from the use of context to provide the signal for learning such *context-dependent* representations. (from a modeling perspective)



The WordPiece* embedding size ***E*** *(context-independent)*
Hidden-layer embeddings ***H*** *(context-dependent)*

1) Projection of one-hot encoded vector V into the low dimension space E
2) Projection of E into the hidden layer H

※ subword tokenization methods

***WordPiece:**
tokenization based on likelihood

어휘 사전
a, b, c, e, l, m, n, o, s, t, u

$$\frac{p(st)}{p(s)p(t)}$$

Likelihood 기준 vocabulary를 만든다

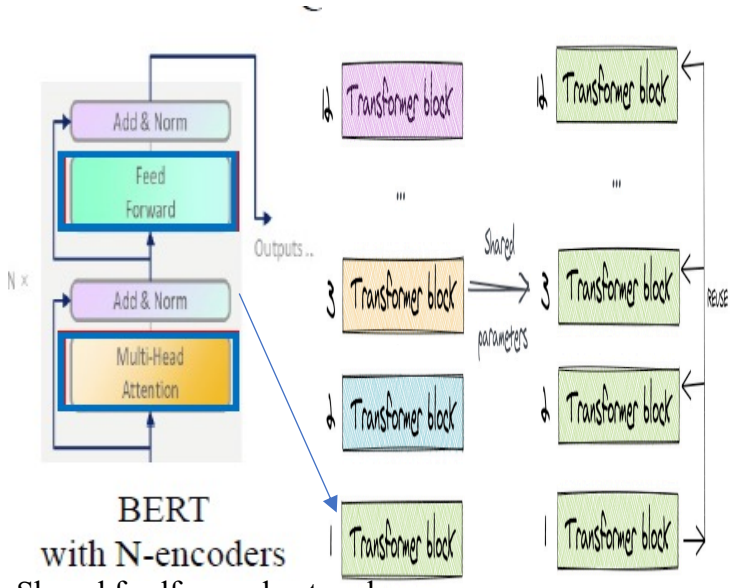
Byte Pair Encoding (BPE): tokenization based on frequency, Byte-Level Byte Pair Encoding (BBPE): tokenization based on frequency

vocabulary
a, b, c, e, l, m, n, o, s, t, u, st, me

Input text = stem
Tokenization procedure:
[stem] -> [st, ##em] -> [st, ##e, ##m]

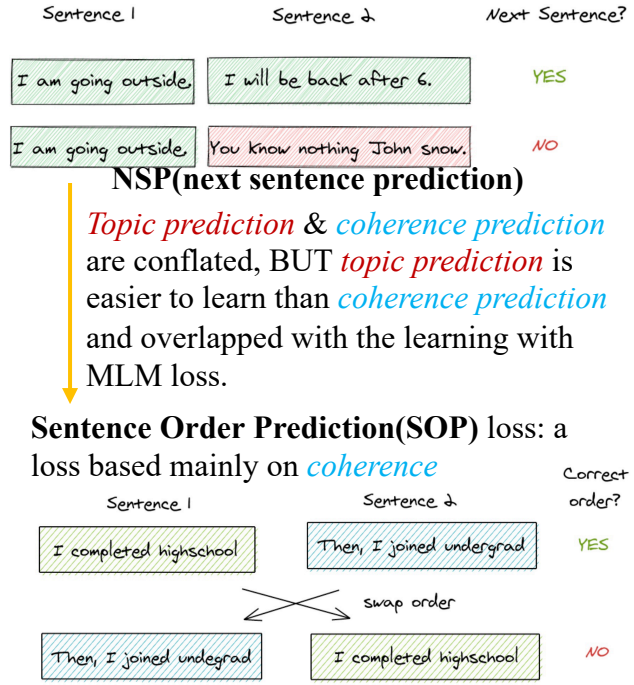
Tokens = [st, ##e, ##m]

2. Cross-layer **parameter sharing**



Shared feedforward network & Shared attention

3. Inter-sentence coherence loss



NSP(next sentence prediction)

Topic prediction & *coherence prediction* are conflated, BUT *topic prediction* is easier to learn than *coherence prediction* and overlapped with the learning with MLM loss.

Sentence Order Prediction(SOP) loss: a loss based mainly on *coherence*

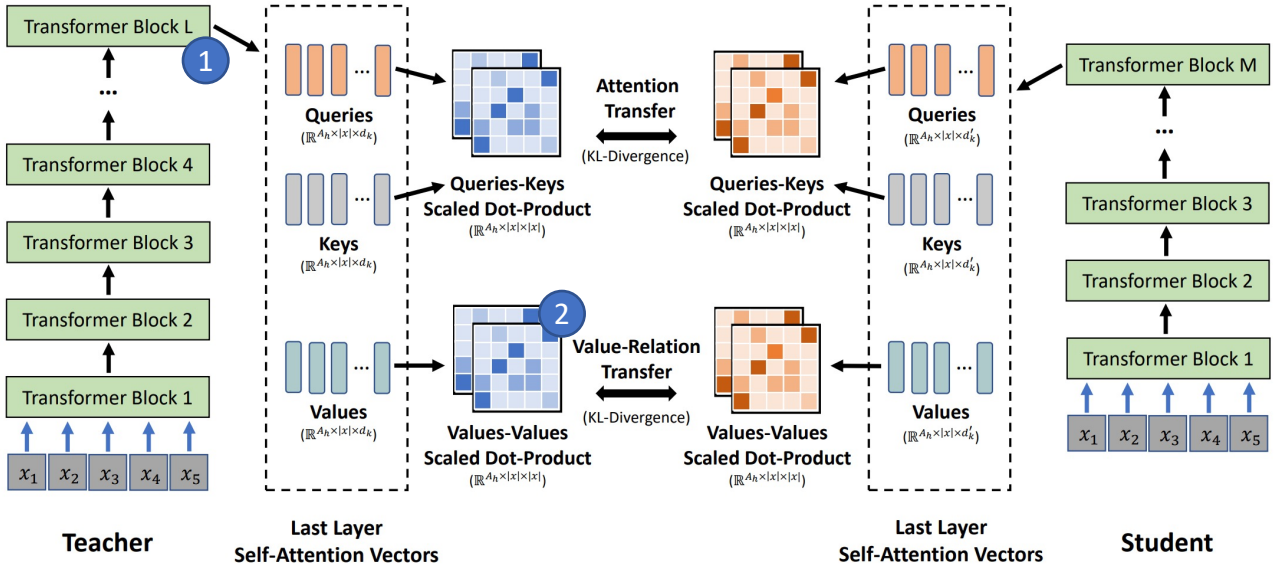
Training Efficiency

Data-throughput for BERT-large is about 3.17x higher compared to ALBERT-xxlarge.

Models	Steps	Time	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
BERT-large	400k	34h	93.5/87.4	86.9/84.3	87.8	94.6	77.3	87.2
ALBERT-xxlarge	125k	32h	**94.0/88.1**	**88.3/85.3**	87.8	**95.4**	**82.5**	**88.7**

MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers, 2020

https://arxiv.org/abs/2002.10957

Motive	Pre-trained language models usually consist of hundreds of millions of parameters which brings challenges for fine-tuning and online serving in real-life applications due to latency and capacity constraints.	***Keywords***	MiniLM, Knowledge distillation, Deep self-attention distillation, Teacher assistant, Self-Attention Distribution Transfer, Self-Attention Value-Relation Transfer
Introduction	• MiniLM is the deep self-attention distillation framework for task-agnostic Transformer based LM distillation • The key idea is to deeply **mimic the self-attention modules**: distilling the self-attention module of the **last Transformer layer** of the teacher model, alleviating the difficulties in layer mapping between the teacher and student models		• Using the **scaled dot-product between values** in the self-attention module as the new deep self-attention knowledge: converting representations of different dimensions into relation matrices with the same dimensions • **Teacher assistant** helps the distillation of large pre-trained Transformer based model
Model	• Deep self-attention distillation (1) training the student by deeply mimicking the self-attention module of the teacher's last layer (2) transferring the relation between values to achieve a deeper mimicry in addition to performing attention distributions (i.e., the scaled dot-product of queries and keys) transfer in the self-attention module (3) Using a **teacher assistant** when the size gap between the teacher model and student model is large : First distilling the teacher into a teacher assistant with L-layer Transformer, The assistant model is then used as the teacher to guide the training of the final student • MiniLM avoids restrictions on the number of student layers and the effort of finding the best layer mapping. Distilling relation between self-attention values allows the hidden size of students to be more flexible and avoids introducing linear matrices to transform student representations		

04 Interpretation and Theoretical Analysis

Interpretation and theoretical analysis

- Knowledge of PTMs: Linguistic knowledge & world knowledge
  - A Structural Probe for Finding Syntax in Word Representations
  - Linguistic Knowledge and Transferability of Contextual Representations
  - What Does BERT Learn about the Structure of Language?
  - Open Sesame: Getting Inside BERT's Linguistic Knowledge
  - Evaluating Commonsense in Pre-Trained Language Models

- Robustness of PTMs
  - Trick Me If You Can: Human-in-the-loop Generation of Adversarial Examples for Question Answering
  - Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment
- Structural sparsity of PTMs
  - What Does BERT Look At? An Analysis of BERT's Attention
  - Revealing the Dark Secrets of BERT
- Theoretical analysis of PTMs
  - Why Does Unsupervised Pre-training Help Deep Learning?
  - A Theoretical Analysis of Contrastive Unsupervised Representation Learning

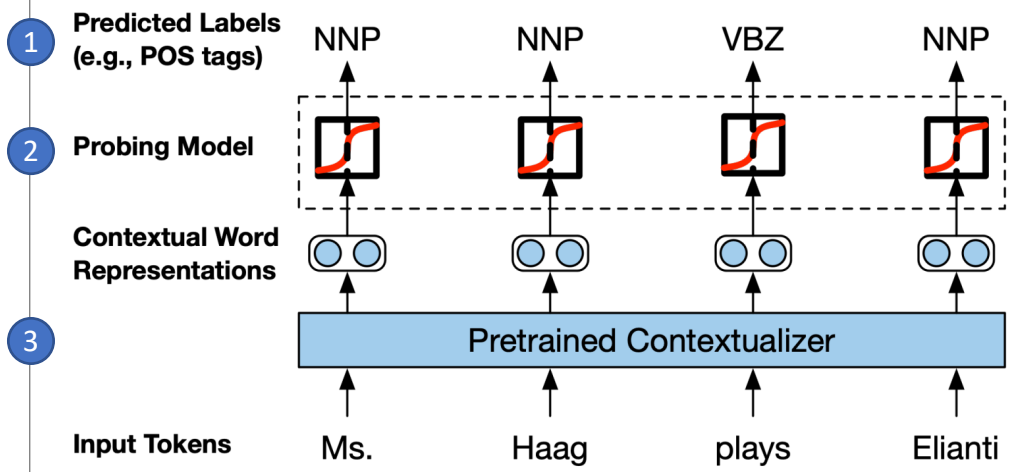
A Structural Probe for Finding Syntax in Word Representations, 2019

https://aclanthology.org/N19-1419/

Motive	Recent work has improved our ability to detect linguistic knowledge in word representations. Current methods for detecting syntactic knowledge do not test whether syntax trees are represented in their entirety. Open questions remain as to **whether deep contextual models encode entire parse trees in their word representations**.	***Keywords***	Linguistic knowledge, Syntactic knowledge, Dependency parse trees,
Introduction	• A structural probe is a simple model which tests whether **syntax trees** are consistently embedded in a linear transformation of a neural network's word representation space • A simple structural probe for **finding syntax in word representations** • **Tree structure** is embedded if the transformed space has the property that squared L2 distance between two words' vectors corresponds to the number of edges between the words in the parse tree	***Properties***	• Distances are guaranteed non-negative and symmetric • The model not only encodes which word is governed by which other word, but each word's proximity to every other word in the syntax tree
Model (Structural Probe)	• Goal: testing whether a neural network embeds each sentence's dependency parse tree in its contextual word representations – a structural hypothesis • Definition: to embed a graph is to learn a vector representation of each node such that geometry in the vector space - distances and norms - approximates geometry in the graph, it is **finding the distance on the original space that best fits the tree metrics** • Every inner product corresponds to a **distance metric**. squared distances is defined as $d_B(\mathbf{h}_i^\ell, \mathbf{h}_j^\ell)^2 = (B(\mathbf{h}_i^\ell - \mathbf{h}_j^\ell))^T (B(\mathbf{h}_i^\ell - \mathbf{h}_j^\ell))$ • The parameters of our probe are exactly the matrix B, approximating through gradient descent which we train to recreate the tree distance between all pairs of words in all sentences in the training set of a parsed corpus $\min_B \sum_\ell \frac{1}{\lvert s^\ell \rvert^2} \sum_{i,j} \lvert d_{T^\ell}(w_i^\ell, w_j^\ell) - d_B(\mathbf{h}_i^\ell, \mathbf{h}_j^\ell)^2 \rvert$	***Model (Tree depth Structural Probe)***	• The **parse tree depth**: defined as **the number of edges** in the parse tree between a word and the root of the tree. • To see if there exists a squared norm on the word representation space that encodes this tree norm. • Replacing the vector distance function with the squared vector norm $\lVert \mathbf{h}_i \rVert_A = (B\mathbf{h}_i)^T (B\mathbf{h}_i)$ • Training B to recreate the parse tree depth of a word
		Evaluation	• Whether representations from ELMo and BERT (BASE and LARGE), models pre-trained on language modeling-like objectives, embed parse trees • ELMo and BERT-large layers are dimensionality 1024; BERT-base layers are dimensionality 768

Linguistic Knowledge and Transferability of Contextual Representations, 2019

https://arxiv.org/abs/1903.08855

Motive	**Contextual word representations (CWR)** derived from large-scale neural language models are successful across a diverse set of NLP tasks, suggesting that they encode useful and transferable features of language. However, their linguistic knowledge and transferability are not yet well understood	***Keywords***	Pretrained word representations, Contextual word representations, Probing model,
Introduction	• Studying CWRs with a diverse set of 17 probing tasks designed to assess a wide array of phenomena, such as coreference, knowledge of semantic relations, and entity information, among others. • Using probing models to analyze the linguistic information within CWRs and Generating features for words from pretrained contextualizers and train a model to make predictions from those features		1. What features of language do these vectors capture, and what do they miss? 2. How and why does transferability vary across representation layers in contextualizers? 3. How does the choice of pretraining task affect the vectors' learned linguistic knowledge and transferability?
Model	① **Probing Tasks** • Token Labeling part-of-speech tagging (POS), CCG supertagging (CCG, syntactic roles), syntactic constituency ancestor tagging, semantic tagging task (ST), Preposition supersense disambiguation (PS), event factuality (EF) • Segmentatoin Syntactic chunking (Chunk), Named entity recognition (NER), Grammatical error detection (GED), Conjunct identification (Conj) • Pairwise Relations Arc prediction, arc classification, coreference arc prediction ② **Probing Model** : a linear model as our probing model; limiting its capacity enables us to focus on what information can be easily extracted from CWRs ③ **Contextualizers**: ELMo, OpenAI transformer, BERT		

What Does BERT Learn about the Structure of Language?, 2019

https://aclanthology.org/P19-1356/

3. Summaries of Papers

Motive

BERT is a recent language representation model that has surprisingly performed well in diverse language understanding benchmarks. This remarkable result suggests that BERT could "learn" structural information about language. Can we **unveil the representations learned by BERT to proto-linguistics structures**?

Keywords

BERT,
Linguistic structures,
BERT by Layers,
Span representations,

Introduction

- Investigating the **linguistic structure implicitly learned** by BERT's representation, Performing a series of experiments to probe the **nature of the representations learned by different layers** of BERT
- The lower layers capture phrase-level information, which gets diluted in the upper layers
- BERT captures a rich hierarchy of linguistic information, with surface features in lower layers, syntactic features in middle layers and semantic features in higher layers

- BERT requires deeper layers for handling harder cases involving long-distance dependencies
- Using the recently introduced Tensor Product Decomposition Network (TPDN) (McCoy et al., 2019) to explore different hypotheses about the compositional nature of BERT's representation and find that BERT implicitly captures classical, tree-like structures

Model & Results

① **Phrasal Syntax**

- Based on the *bert-base-uncased variant*, which consists of 12 layers, each having a hidden size of 768 and 12 attention heads (110M param)
- Extracting **span representations** from each layer of BERT
- Computing the span representation at each layer by concatenating the **first and last hidden vector**, along with their element-wise product and difference
- BERT mostly captures phrase-level information in the lower layers and that this information gets gradually diluted in higher layers

(a) Layer 1 (b) Layer 2 (c) Layer 11 (d) Layer 12

Dataset

- (**Phrasal Syntax**) Randomly picking 3000 labeled chunks and 500 spans not labeled as chunks from the CoNLL 2000 chunking dataset

- Performing a k-means clustering on span representations with k = 10, the number of distinct chunk types, Evaluating the resulting clusters using the Normalized Mutual Information (NMI) metric
- The lower layers encode phrasal information better than higher layers

layer	1	2	3	4	5	6	7	8	9	10	11	12
NMI	**0.38**	0.37	0.35	0.3	0.24	0.2	0.19	0.16	0.17	0.18	0.16	0.19

Evaluating Commonsense in Pre-Trained Language Models, 2020

https://arxiv.org/abs/1911.11931

3. Summaries of Papers

Motive

relatively little work has been done investigating commonsense knowledge contained in contextualized representations, which is crucial for human question answering and reading comprehension.

Commonsense knowledge spans "a huge portion of human experience, encompassing knowledge about the spatial, physical, social, temporal, and psychological aspects of typical everyday life."

Keywords

Commonsense knowledge,
Contextualized representations,
Conjunction Acceptability,
CAT dataset,

Introduction

- The models include off-the-shelf embeddings from GPT, GPT2, BERT, XLNet and RoBERTa
- The benchmarks include Conjunction Acceptability, Sense Making, Winograd Schema Challenge, SWAG (multiple choices), HellaSwag (agumented version of SWAG) , Sense Making with Reasoning, and Argument Reasoning Comprehension

Model

- **Commonsense ability**
  (1) a model with commonsense ability should have basic knowledge about the world, for example, water always goes down
  (2) a model should have the ability to reason over commonsense knowledge, such as water always goes down because there is gravity on the earth
- **Reframing all the tasks** into sentence-scoring tasks by substitution or concatenation
- Score

$$Score(S) = \frac{\sum_{k=1}^{n} log(P_\theta(w_k|context_k)}{n}$$

Original:
Paul tried to call George on the phone, but he wasn't successful.
Who is he?
Candidate: A. Paul (correct) B. George

Reframed:
A. Paul tried to call George on the phone, but Paul wasn't successful. (Positive sample)
B. Paul tried to call George on the phone, but George wasn't successful. (Negative sample)

Code

- From the paper: https://github.com/XuhuiZhou/CATS

Results

- Bi-directional context can be more useful for learning commonsense. Intuitively, the models with bi-directional context can make more sentence-level inference.
- A larger training dataset intuitively allows a model to have access to more commonsense knowledge, thus performs better
- There is a decrease of performances as inference step increases. SWAG and HellaSwag fall out the trend.
- The commonsense knowledge contained in the pre-trained models may remain in a surface level without deep semantic comprehension

	CA	WSC	SM	SMR	SWAG	HellaSwag	ARCT1	ARCT2	Average
RANDOM	0.500	0.500	0.500	0.333	0.250	0.250	0.500	0.500	0.416
GPT	0.831	0.555	0.735	0.318	0.603	0.261	0.471	0.524	0.537
GPT2-base	0.787	0.512	0.706	0.355	0.522	0.300	0.466	0.509	0.520
GPT2-medium	0.885	0.569	0.746	0.385	0.597	0.338	0.462	0.527	0.564
BERT-base	0.891	0.576	0.697	0.418	0.631	0.364	0.468	0.503	0.569
BERT-large	0.934	0.618	0.694	0.444	0.683	0.395	0.477	0.517	0.595
XLNet-base	0.809	0.544	0.662	0.374	0.551	0.381	0.516	0.526	0.545
XLNet-large	0.891	0.636	0.583	0.394	0.692	0.435	0.563	0.570	0.596
RoBERTa-base	0.956	0.625	0.750	0.423	0.691	0.414	0.500	0.537	0.612
RoBERTa-large	0.962	0.693	0.792	0.512	0.761	0.500	0.543	0.599	0.670
HUMAN	0.993	0.920	0.991	0.975	0.880	0.945	0.909	0.909	0.945

Robustness of PTMs: **TEXTFOOLER**

IS BERT REALLY ROBUST? A STRONG BASELINE FOR NATURAL LANGAUGE ATTACK ON TEXT CLASSIFICIATION AND ENTAILMENT

자연어 공격 시스템 필요한 3 가지 속성
–사람의 시각에서 결과 (Label)는 동일 (원본 문장 Label = 적대적 문장 Label) But 모델 결과는 반대
–의미적 유사성 원본 문장 의미 = 적대적 문장 의미
–언어적 유창성 (문법 오류 x)

$$F(X_{adv}) \neq F(X), \text{and } Sim(X_{adv}, X) \geq \epsilon,$$

- N개의 문장, label
–X = {X1, X2,...,XN } , Y = {Y1, Y2,...,YN }
–F : X(input) → Y(output). **Pre-trained model**
–$Xadv$: 적대적 예시문장(adversarial example)
–Sim: 두 문장의 유사도를 측정하는 함수
–ϵ: 최소 유사도 허용수치

Step1: Word Importance Ranking

1: Initialization: $X_{adv} \leftarrow X$
2: **for** each word w_i in X **do**
3: Compute the importance score I_{w_i} via Eq.2
4: **end for**
5:
6: Create a set W of all words $w_i \in X$ sorted by the descending order of their importance score I_{w_i}.

중요 단어 순서 정렬

$$I_{w_i} = \begin{cases} F_Y(X) - F_Y(X_{\setminus w_i}), & \text{if } F(X) = F(X_{\setminus w_i}) = Y \\ (F_Y(X) - F_Y(X_{\setminus w_i})) + (F_{\bar{Y}}(X_{\setminus w_i}) - F_{\bar{Y}}(X)), & \\ \text{if } F(X) = Y, F(X_{\setminus w_i}) = \bar{Y}, \text{ and } Y \neq \bar{Y}. & \end{cases}$$

–$X\setminus wi$: X에서 wi단어를 제거한 문장
–원본 문장(X)라벨과 단어 제거 문장($X\setminus wi$) 라벨이 같을 경우: 두 문장의 confidence 값 차이
–원본 문장(X)라벨과 단어 제거 문장($X\setminus wi$) 라벨이 다를 경우:
원본Class(Y)에 대한 두 문장의confidence 값 차이+바뀐Class(Y)에 대한 두 문장의 confidence 값 차이

Step2: Word Transformer

7: Filter out the stop words in W.

필터링 Stop words(The, when, none 등), 문법적 오류

8: **for** each word w_j in W **do**
9: Initiate the set of candidates CANDIDATES by extracting the top N synonyms using CosSim(Emb_{w_j}, Emb_{word}) for each word in $Vocab$.

Synonym Extraction
CANDIDATES: wj(현재단어)와 교체가능한 단어집합
= wj와 **코사인 유사도**가 δ(델타)보다 높은 top **N개**의 단어들 집합
– N & δ (N: 50, δ: 0.7 균형적)
: N이 높을수록, δ가 낮을수록 다양한 동의어 후보, but 의미적 유사성 하락

Robustness of PTMs: **TEXTFOOLER**

IS BERT REALLY ROBUST? A STRONG BASELINE FOR NATURAL LANGAUGE ATTACK ON TEXT CLASSIFICIATION AND ENTAILMENT

TextFooler를 통한 NLP 모델공격 문장 생성 예시

	Movie Review (Positive (POS) ↔ Negative (NEG))
Original (Label: NEG)	The characters, cast in impossibly ***contrived situations***, are ***totally*** estranged from reality.
Attack (Label: POS)	The characters, cast in impossibly ***engineered circumstances***, are ***fully*** estranged from reality.
Original (Label: POS)	It cuts to the ***knot*** of what it actually means to face your ***scares***, and to ride the ***overwhelming*** **metaphorical wave** that life wherever it takes you.
Attack (Label: NEG)	It cuts to the ***core*** of what it actually means to face your ***fears***, and to ride the ***big*** **metaphorical wave** that life wherever it takes you.
	SNLI (Entailment (ENT), Neutral (NEU), Contradiction (CON))
Premise	Two small boys in blue soccer uniforms use a wooden set of steps to wash their hands.
Original (Label: CON)	The boys are in band ***uniforms***.
Adversary (Label: ENT)	The boys are in band ***garment***.
Premise	A child with wet hair is holding a butterfly decorated beach ball.
Original (Label: NEU)	The ***child*** is at the ***beach***.
Adversary (Label: ENT)	The ***youngster*** is at the ***shore***.

다른 기존 알고리즘 대비 더 작은 단어를 교체했음에도 더 높은 성공률을 기록하였다

Dataset	Model	Success Rate	% Perturbed Words
IMDB	(Li et al. 2018)	86.7	6.9
	(Alzantot et al. 2018)	97.0	14.7
	Ours	**99.7**	**5.1**
SNLI	(Alzantot et al. 2018)	70.0	23.0
	Ours	**95.8**	**18.0**
Yelp	(Kuleshov et al. 2018)	74.8	-
	Ours	**97.8**	**10.6**

Table 5: Comparison of our attack system against other published systems. The target model for IMDB and Yelp is LSTM and SNLI is InferSent.

두가지 Task로 실험
-Text Classification
-Text Entailment

	WordCNN					WordLSTM					BERT				
	MR	**IMDB**	**Yelp**	**AG**	**Fake**	**MR**	**IMDB**	**Yelp**	**AG**	**Fake**	**MR**	**IMDB**	**Yelp**	**AG**	**Fake**
Original Accuracy	78.0	89.2	93.8	91.5	96.7	80.7	89.8	96.0	91.3	94.0	86.0	90.9	97.0	94.2	97.8
After-Attack Accuracy	2.8	0.0	1.1	1.5	15.9	3.1	0.3	2.1	3.8	16.4	11.5	13.6	6.6	12.5	19.3
% Perturbed Words	14.3	3.5	8.3	15.2	11.0	14.9	5.1	10.6	18.6	10.1	16.7	6.1	13.9	22.0	11.7
Semantic Similarity	0.68	0.89	0.82	0.76	0.82	0.67	0.87	0.79	0.63	0.80	0.65	0.86	0.74	0.57	0.76
Query Number	123	524	487	228	3367	126	666	629	273	3343	166	1134	827	357	4403
Average Text Length	20	215	152	43	885	20	215	152	43	885	20	215	152	43	885

	InferSent		ESIM		BERT	
	SNLI	**MultiNLI (m/mm)**	**SNLI**	**MultiNLI (m/mm)**	**SNLI**	**MultiNLI (m/mm)**
Original Accuracy	84.3	70.9/69.6	86.5	77.6/75.8	89.4	85.1/82.1
After-Attack Accuracy	3.5	6.7/6.9	5.1	7.7/7.3	4.0	9.6/8.3
% Perturbed Words	18.0	13.8/14.6	18.1	14.5/14.6	18.5	15.2/14.6
Semantic Similarity	0.50	0.61/0.59	0.47	0.59/0.59	0.45	0.57/0.58
Query Number	57	70/83	58	72/87	60	78/86
Average Text Length	8	11/12	8	11/12	8	11/12

Adv. 를 추가하여 training시킬 경우, 더 높은 성능을 보인다

	MR		SNLI	
	Af. Acc.	**Pert.**	**Af. Acc.**	**Pert.**
Original	11.5	16.7	4.0	18.5
+ Adv. Training	**18.7**	**21.0**	**8.3**	**20.1**

Table 12: Comparison of the after-attack accuracy ("Af. Acc.") and percentage of perturbed words ("Pert.") of original training ("Original") and adversarial training ("+ Adv. Train") of BERT model on MR and SNLI dataset.

Structural Sparsity of PTMs

What Does BERT Look At? An Analysis of **BERT's Attention**

Recent/Previous works

- **outputs** of language models on chosen input sentences
- **internal vector representations** of the model

$$\alpha_{ij} = \frac{\exp\left(q_i^T k_j\right)}{\sum_{l=1}^{n} \exp\left(q_i^T k_l\right)} \qquad o_i = \sum_{j=1}^{n} \alpha_{ij} v_j$$

The output o(i th token) of the attention head is a weighted sum of the value vectors.

This study

- explore how **BERT's attention heads behave.** (attention map)
- probe each attention head for **linguistic phenomena**.
- evaluate the ability of the heads to **classify various syntactic relations**.
- propose an attention-based probing classifier

3. Surface-Level Patterns in Attention

Setup

- attention maps from bert base over 1000 random Wikipedia segments
- Each segment consists of 128 tokens corresponding to 2 consecutive paragraphs
- [CLS]<paragraph-1>[SEP]<paragraph-2>[SEP]

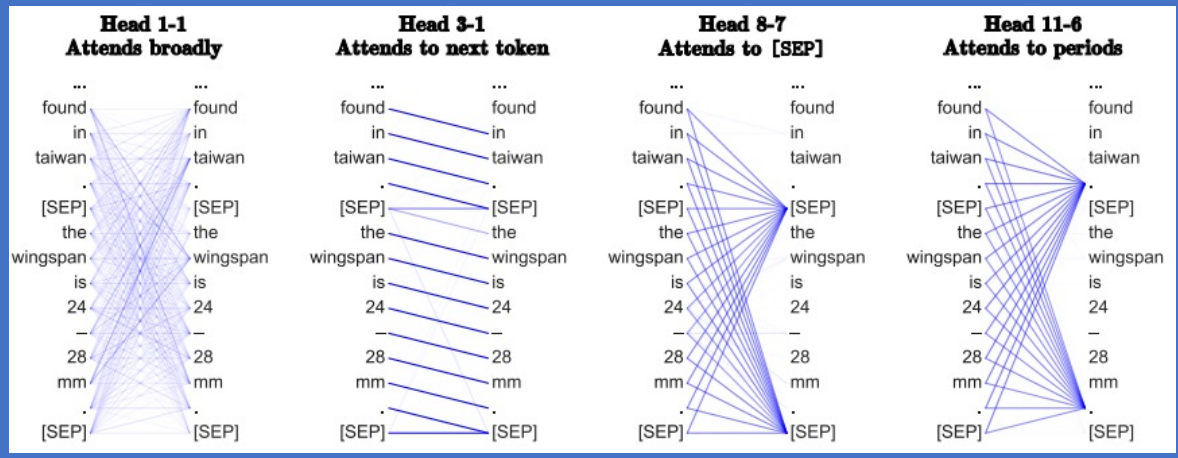
3.1 **Relative Position**

BERT's attention head attend to the current, previous, or next token

- Most heads put little attention on the **current** token
- Four attention heads(in layers 2,4,7,8) put > 50% on the **previous** token
- Four attention heads(in layers 1,2,3,6) put > 50% on the **next** token

3.2 Attending to **Separator Tokens** ([SEP] [CLS] [. or ,])

- Over half of Layers 6-10 focuses on [SEP]
  ([SEP] exists twice on average in 128 tokens segment : around 1/64)
- [SEP] aggregates segment-level information ? No
  => expect attention heads processing [SEP] to attend broadly over the whole segment to build up these representations. they instead almost entirely attend to themselves and the other [SEP] token.
- Attending to [SEP] does not change BERT's output
  - Gradient based measures of feature importance is low
- [SEP] is supposed to be used as a "no-op" for attention heads



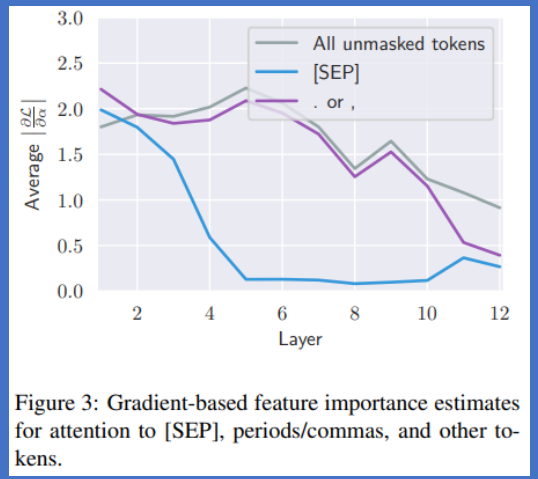
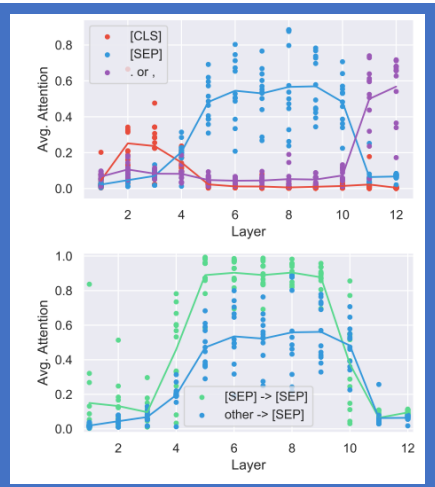


Figure 3: Gradient-based feature importance estimates for attention to [SEP], periods/commas, and other tokens.

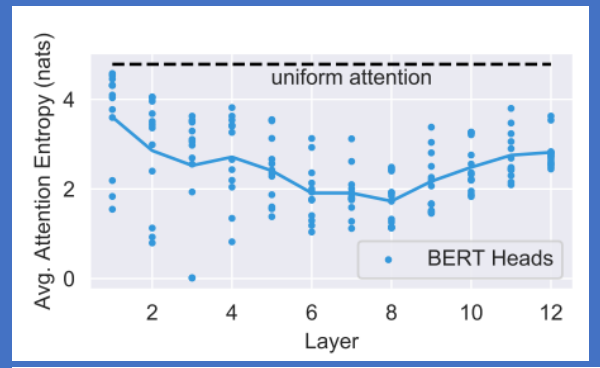


Structural Sparsity of PTMs

What Does BERT Look At? An Analysis of **BERT's Attention**

3.3 **Focused vs Broad Attention**

- whether attention heads focus on a few words or broadly over words
  - Broad = high entropy, focused = low entropy
  - Entropy가 높다 = 정보량이 낮다
- In lower layers, some heads have very **broad** attention
  - High entropy attention heads spend at most 10% on any single word
  - The output of these heads is a bag-of-vectors representation of the sentence.
- Measured entropies for all attention heads from only the [CLS] token
- The last layer has a high entropy from [CLS]
- Make sense [CLS] token is used as input for the "next sentence prediction"



4. Probing Individual Attention Heads

4.1 Method

- BERT uses byte-pair tokenization (words are split up into multiple tokens)
- Convert token-token attention maps to **word-word attention maps**
- Sum up the attention weights over its tokens

4.2 **Dependency Syntax**

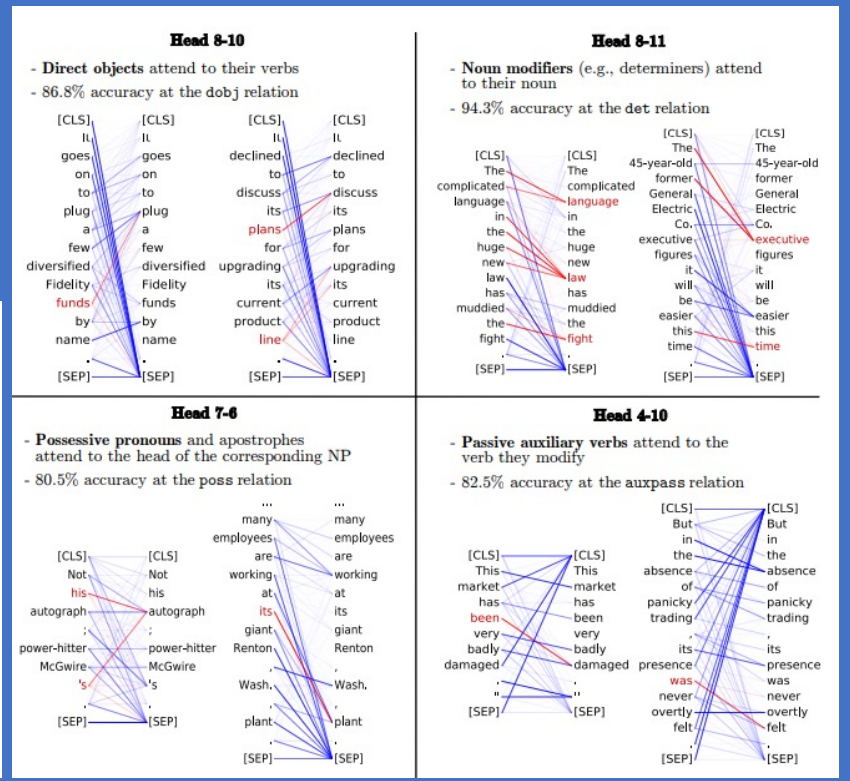
Setup

- On the WSJ Penn Tree-bank
- Evaluate **both** "directions" of prediction
  - Head word attending to the dependent
  - Dependent word attending to the head

Results

- No single attention head for syntax "overall"
- Some attention heads specialize to specific dependency relations
  : we would not say individual attention heads capture dependency structure as a whole (future work)
- The dependent attends to the head word rather than the other way

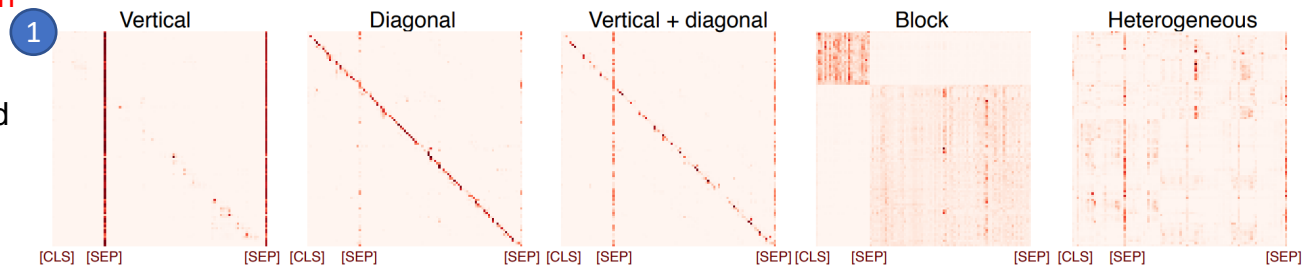
Relation	Head	Accuracy	Baseline
All	7-6	34.5	26.3 (1)
prep	7-4	66.7	61.8 (-1)
pobj	9-6	**76.3**	34.6 (-2)
det	8-11	**94.3**	51.7 (1)
nn	4-10	70.4	70.2 (1)
nsubj	8-2	58.5	45.5 (1)
amod	4-10	75.6	68.3 (1)
dobj	8-10	**86.8**	40.0 (-2)
advmod	7-6	48.8	40.2 (1)
aux	4-10	81.1	71.5 (1)
poss	7-6	**80.5**	47.7 (1)
auxpass	4-10	**82.5**	40.5 (1)
ccomp	8-1	**48.8**	12.4 (-2)
mark	8-2	**50.7**	14.5 (2)
prt	6-7	**99.1**	91.4 (-1)



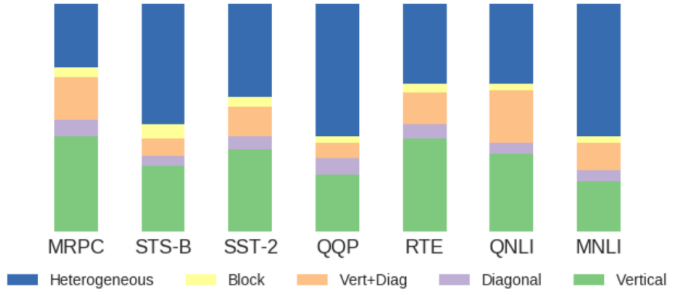
Revealing the Dark Secrets of BERT, 2019

https://arxiv.org/abs/1908.08593

Motive	BERT-based architectures currently give state-of-the-art performance on many NLP tasks. However, **the exact mechanisms** that contribute to the BERT's outstanding performance **still remain unclear**. We address this problem through selecting a set of linguistic features of interest and conducting a series of experiments	***Keywords***	BERT, Self-attention weights, Self-attention maps, Self-attention patterns,
Introduction	• Aiming to provide insights about how well these features are captured by BERT. • We propose a methodology and offer the first detailed analysis of BERT's capacity to capture different kinds of linguistic information by encoding it in its **self-attention weights**		• We present evidence of BERT's over-parametrization and suggest a counterintuitive yet frustratingly simple way of improving its performance,
Model & Results	• Using bert-base, 12-layer, 768-hidden, 12-heads, 110M paramameters • **Q1. What are the common attention patterns**, how do they change during fine-tuning, and how does that impact the performance on a given task? ① **BERT's self-attention patterns: five frequently occurring patterns** (Manual inspection of self-attention maps for both basic pre-trained and fine-tuned BERT models) 1) Vertical: attention to special BERT tokens [CLS], [SEP] 2) Diagonal: attention to the previous/following tokens 3) Vertical + Diagonal: a mix of the previous two types 4) Block: intra-sentence attention with two distinct sentences 5) Heterogeneous: cannot be characterized • Manually annotated around 400 sample **self-attention maps** as belonging to one of the five classes • Training a **convolutional neural network** with 8 convolutional layers and ReLU activation functions **to classify** input maps into one of these classes. • This model achieved the F1 score of 0.86 on the annotated dataset. We used this classifier **to estimate the proportion** of different self-attention patterns for the target GLUE tasks using up to 1000	***Dataset***	• The subset of GLUE tasks for fine-tuning: MRPC, STS-B, SST-2, QQP, RTE, QNLI, MNLI



• The first three types are most likely associated with language model pre-training, while the last two potentially encode semantic and syntactic information



• (Left) Estimated percentages of the identified self-attention classes for each of the selected GLUE task

Revealing the Dark Secrets of BERT, 2019

https://arxiv.org/abs/1908.08593

3. Summaries of Papers

Model & Results

- Q2. **What linguistic knowledge** is encoded in self-attention weights of the fine-tuned models and what portion of it comes from the pre-trained BERT?

② **Relation-specific heads in BERT**: Focusing on whether BERT captures **FrameNet's relations** between frame-evoking lexical units (predicates) and **core frame elements** whether the links between them produce higher attention weights

- With 473 annotated sentences, for each of these sentences, we obtain pre-trained BERT's attention weights for each of the 144 heads.
- Certain types of linguistic relations may be captured by self-attention patterns in specialized BERT heads

③ **Change in self-attention patterns after fine-tuning**: To study how attention per head changes on average for each of the target GLUE tasks, we calculate **cosine similarity** between pre-trained and fine-tuned BERT's flattened arrays of attention weights

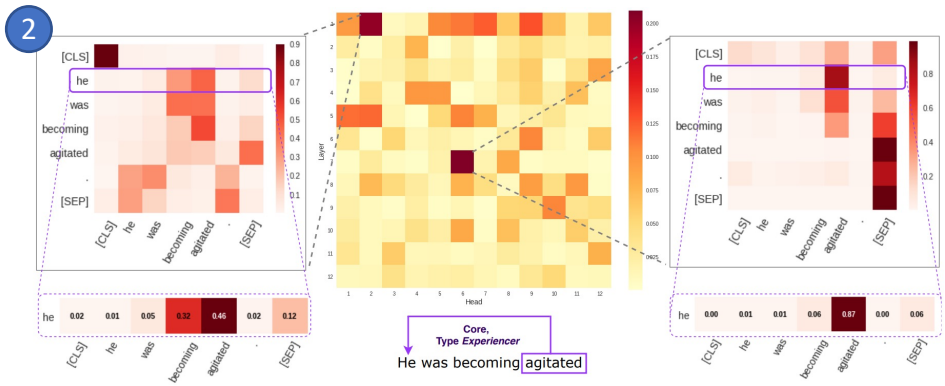
- **The last two layers encode task-specific features** that are attributed to the gain of scores, while earlier layers capture more fundamental and low-level information used in fine-tuned models

④ **Attention to linguistic features**: Investigating whether fine-tuning BERT for a given task creates self-attention patterns which emphasize specific linguistic features

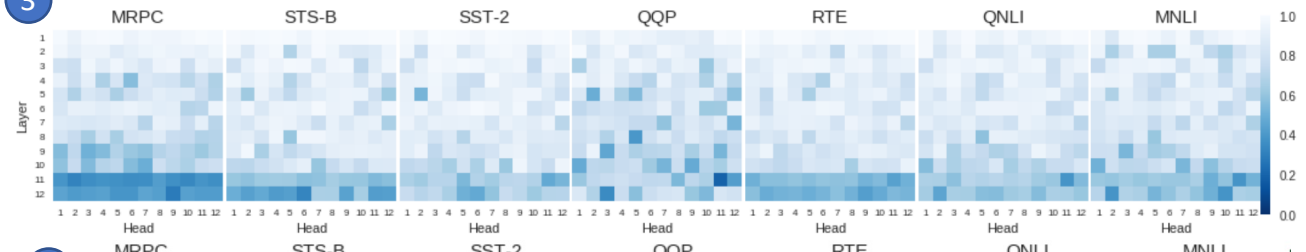
- Investigating attention to nouns, verbs, pronouns, subjects, objects, and negation words , and special BERT tokens across the tasks
- The striped attention maps generally come from BERT pre-training tasks rather than from task-specific linguistic reasoning.

- The heatmap of averaged attention scores over all collected examples suggests that 2 out of 144 heads tend to attend to the parts of the sentence that FrameNet annotators identified as core elements

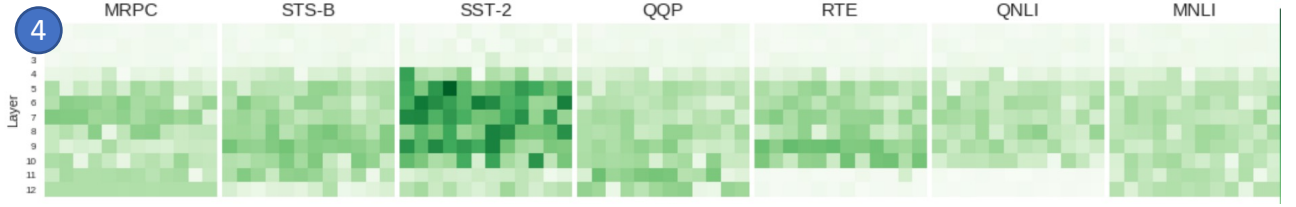
2



3



4



- Q3. How different are the **self-attention patterns** of different heads, and how important are they for a given task?

⑤ **Token-to-token attention**: Investigating the attention patterns between tokens in the same

- Potential head candidates that coincided with diagonally structured attention maps

⑥ **Disabling self-attention heads**: Unexpectedly, disabling some heads leads not to a drop in accuracy, as one would expect, but to an increase in performance. This effect is different across tasks and datasets

Code

- From the paper: https://github.com/text-machine-lab/dark-secrets-of-BERT

05 Beyond fine-tuning: Prompt-based Learning

Pre-train, Fine-tune vs. Pre-train, Prompt, Predict

- In "pre-train, prompt, and predict paradigm, instead of adapting pre-trained LMs to downstream tasks via objective engineering, downstream tasks are reformulated to look more like those solved during the original LM training with the help of a textual prompt.
- For example, when recognizing the emotion of a social media post, "I missed the bus today.", we may continue with a prompt "I felt so ", and ask the LM to fill the blank with an emotion-bearing word.
- By selecting the appropriate prompts we can manipulate the model behavior so that the pre-trained LM itself can be used to predict the desired output, sometimes even without any additional task-specific training
- The advantage of this method is that, given a suite of appropriate prompts, a single LM trained in an entirely unsupervised fashion can be used to solve a great number of tasks
- However, as with most conceptually enticing prospects, there is a catch – this method introduces the necessity for prompt engineering, finding the most appropriate prompt to allow a LM to solve the task at hand.

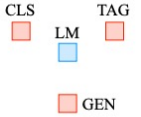
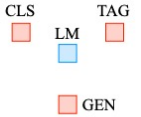
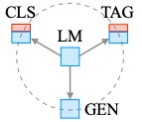
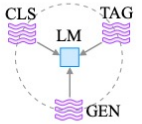
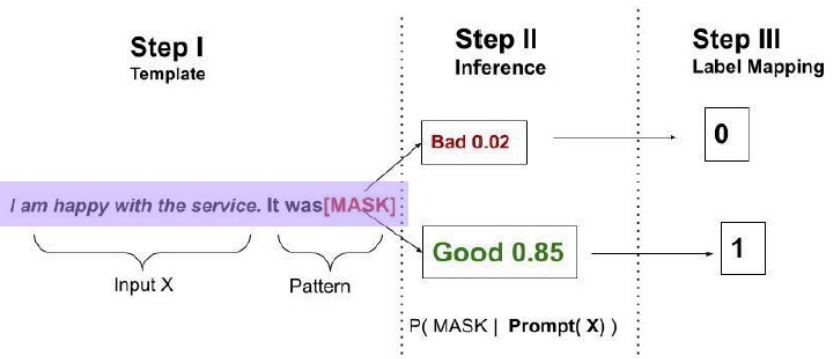
Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Table 1: Four paradigms in NLP. The "**engineering**" column represents the type of engineering to be done to build strong systems. The "**task relation**" column, shows the relationship between language models (LM) and other NLP tasks (CLS: classification, TAG: sequence tagging, GEN: text generation). : fully unsupervised training. : fully supervised training. : Supervised training combined with unsupervised training. indicates a textual prompt. Dashed lines suggest that different tasks can be connected by sharing parameters of pre-trained models. "LM→Task" represents *adapting LMs (objectives) to downstream tasks* while "Task→LM" denotes *adapting downstream tasks (formulations) to LMs*.

Prompt-based Learning: Pre-train, Prompt, and Predict

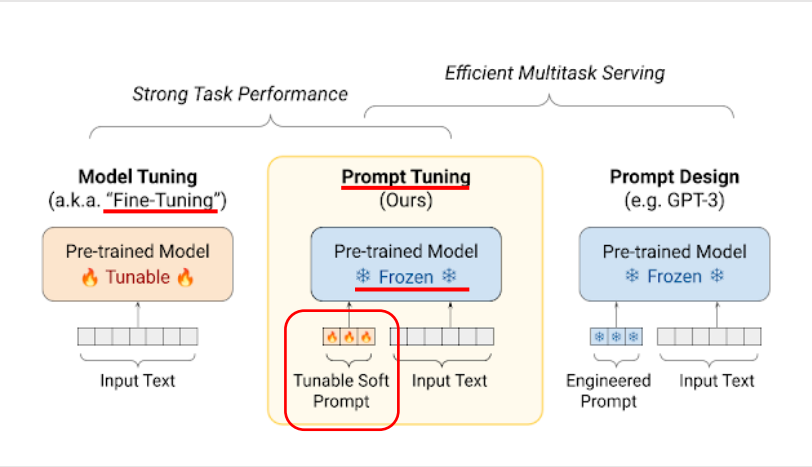
What are Prompts?

Prompts are the text given to the language model to be completed.



NLP Task	Input X	Template	X'	Label Mapping
Sentiment Classification	I liked the film	X, **it is [MASK]**	I liked the film, it is [MASK]	Good: 1 Great:1 Bad: 0 Terrible:0
Aspect-based Sentiment	The pizza is really good, but service is very poor	X, **What about service [MASK]**	The pizza is really good, but service is very poor, What about service [MASK]	Good: 1 Great: 1 Bad: 0 Terrible: 0
Textual Entailment	H: Cats are smaller elephants P: Elephants are bigger than cats	H ? **[MASK]**, P	Cats are smaller elephants? [MASK] Elephants are bigger than cats	Yes: 1 No: 0

Prompting in NLP, by Savas Yıldırım, 2022, Medium



Prompt-based Learning: Pre-train, Prompt, and Predict

Name	Notation	Example	Description
Input	$\boldsymbol{x}$	I love this movie.	One or multiple texts
Output	$\boldsymbol{y}$	++ (very positive)	Output label or text
Prompting Function	$f_{\text{prompt}}(\boldsymbol{x})$	`[X]` Overall, it was a `[Z]` movie.	A function that converts the input into a specific form by inserting the input $\boldsymbol{x}$ and adding a slot `[Z]` where answer $\boldsymbol{z}$ may be filled later.
Prompt	$\boldsymbol{x}'$	I love this movie. Overall, it was a `[Z]` movie.	A text where `[X]` is instantiated by input $\boldsymbol{x}$ but answer slot `[Z]` is not.
Filled Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z})$	I love this movie. Overall, it was a bad movie.	A prompt where slot `[Z]` is filled with any answer.
Answered Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}^*)$	I love this movie. Overall, it was a good movie.	A prompt where slot `[Z]` is filled with a true answer.
Answer	$\boldsymbol{z}$	"good", "fantastic", "boring"	A token, phrase, or sentence that fills `[Z]`

Basic prompting predicts the highest-scoring $\hat{y}$ three steps.

Steps	Description
Prompt Addition	**A prompting function is applied** to modify the input text x into a prompt x' (1) Apply a template, which is a textual string that has two slots: an input slot [X] for input x, an answer slot [Z] for an intermediate generated answer text z (2) Fill slot [X] with the input text x
Answer Search	Define a function that fills in the location [Z] in prompt with the potential answer z (filled prompt) **Search for the highest-scoring text $\hat{z}$** that maximizes the score of the LM $\hat{z} = \underset{z \in Z}{\text{search}}\, P(f_{\text{fill}}(x', z); \theta$
Answer Mapping	Go from the highest-scoring answer $\hat{z}$ to the highest-scoring output $\hat{y}$

Examples of Prompt

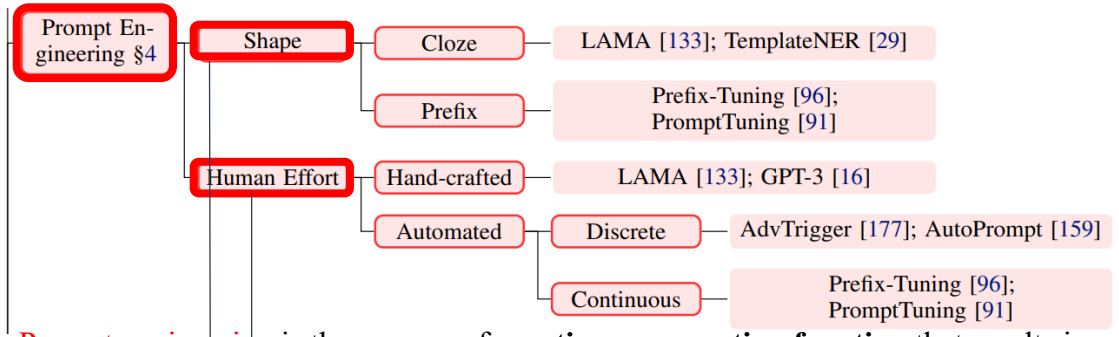
predict

Type	Task	Input (`[X]`)	Template	Answer (`[Z]`)
Text CLS	Sentiment	I love this movie.	Prompt `[X]` The movie is `[Z]`.	great fantastic ...
	Topics	He prompted the LM.	`[X]` The text is about `[Z]`.	sports science ...
	Intention	What is taxi fare to Denver?	`[X]` The question is about `[Z]`.	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	`[X]` What about service? `[Z]`.	Bad Terrible ...
Text-pair CLS	NLI	`[X1]`: An old man with ... `[X2]`: A man walks ...	`[X1]`? `[Z]`, `[X2]`	Yes No ...

Design Considerations for Prompting

Factors	Description
Pre-trained Model Choice	There are a wide variety of pre-trained LMs that could be used to calculate the probability P(x; θ)
Prompt Engineering	Choosing a proper prompt has a large effect not only on the accuracy, but also on which task the model performs in the first place
Answer Engineering	Depending on the task, we may want to design Z differently, possibly along with the mapping function
Prompt-based Training Strategies	Methods to train parameters, either of the prompt, the LM, or both

Prompt-based Learning: Pre-train, Prompt, and Predict



Prompt engineering is the process of **creating a prompting function** that results in the most effective performance on the downstream task.

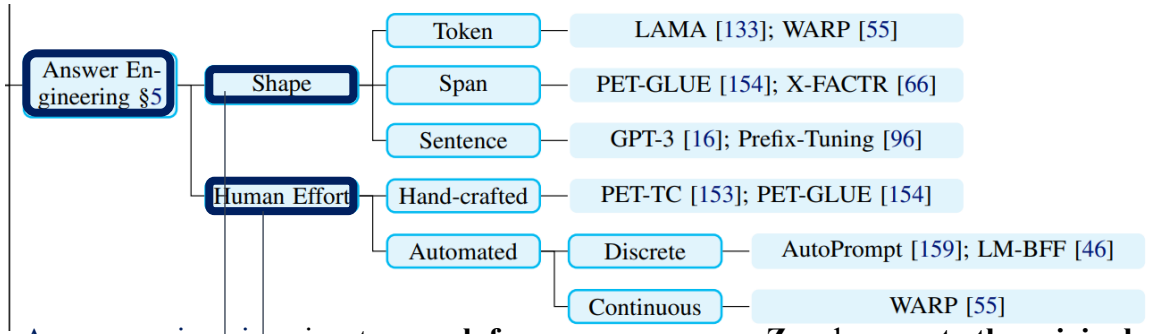
Cloze prompts: fill in the blanks of a textual string (masked LM)
Prefix prompts: continue a string prefix (auto-regressive LM)

Manual Template Engineering is to manually create intuitive templates based on human introspection.

Automated Template Learning is to automate the template design process

Discrete prompts (hard prompts): the prompt is an actual text string
Continuous prompts (soft prompts): the prompt is described directly in the embedding space of the underlying LM

Static prompting function: same prompt template for each input
Dynamic prompting function: a custom template for each input



Answer engineering aims to **search for an answer space Z** and a **map to the original output Y** that results in an effective predictive model.

The shape of an answer characterizes its granularity.
Tokens, Span, Sentence

Manual Answer Space Design is to crafted manually by an interested system or benchmark designer.
Unconstrained, Constrained (limited to relevant topics)

Automated Answer Search is to do automatic answer search

Discrete Answer Search: start with an initial answer space, and then expand this answer space to broaden its coverage, or performing relation extraction and use words as an answer
Answer Paraphrasing, Prune-then-Search, Label Decomposition

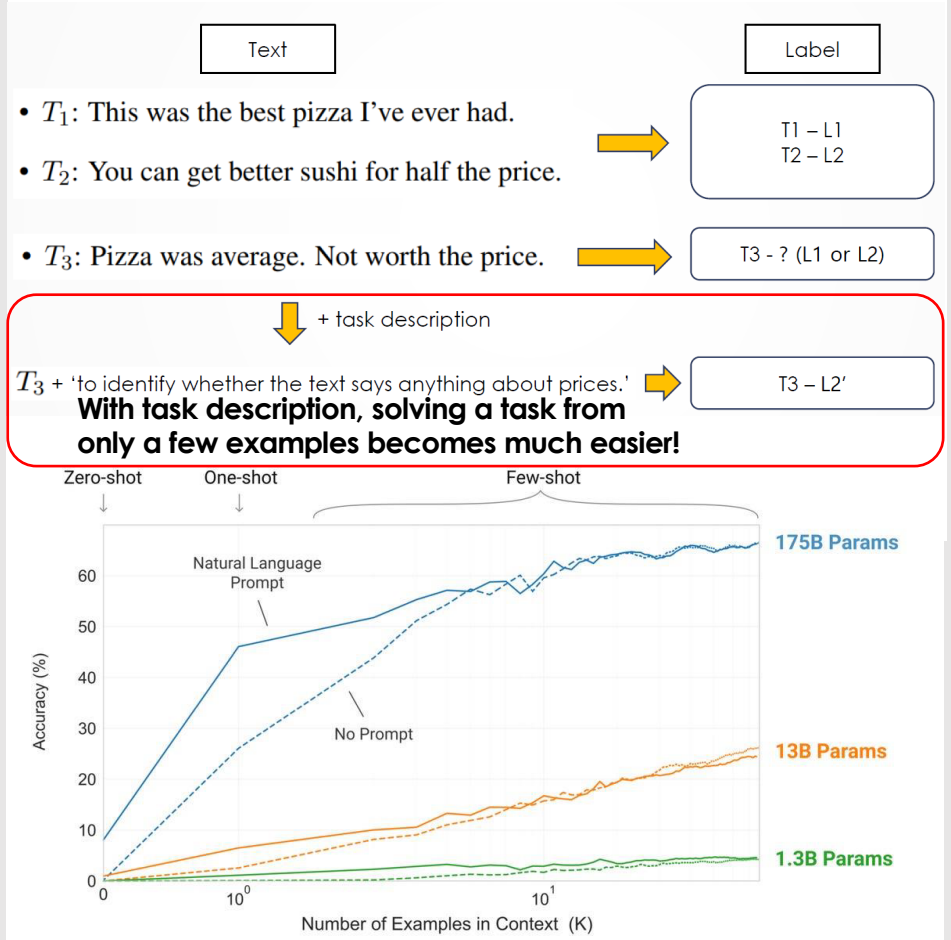
Continuous prompts: Very few works explore the possibility of using soft answer tokens which can be optimized through gradient descent.

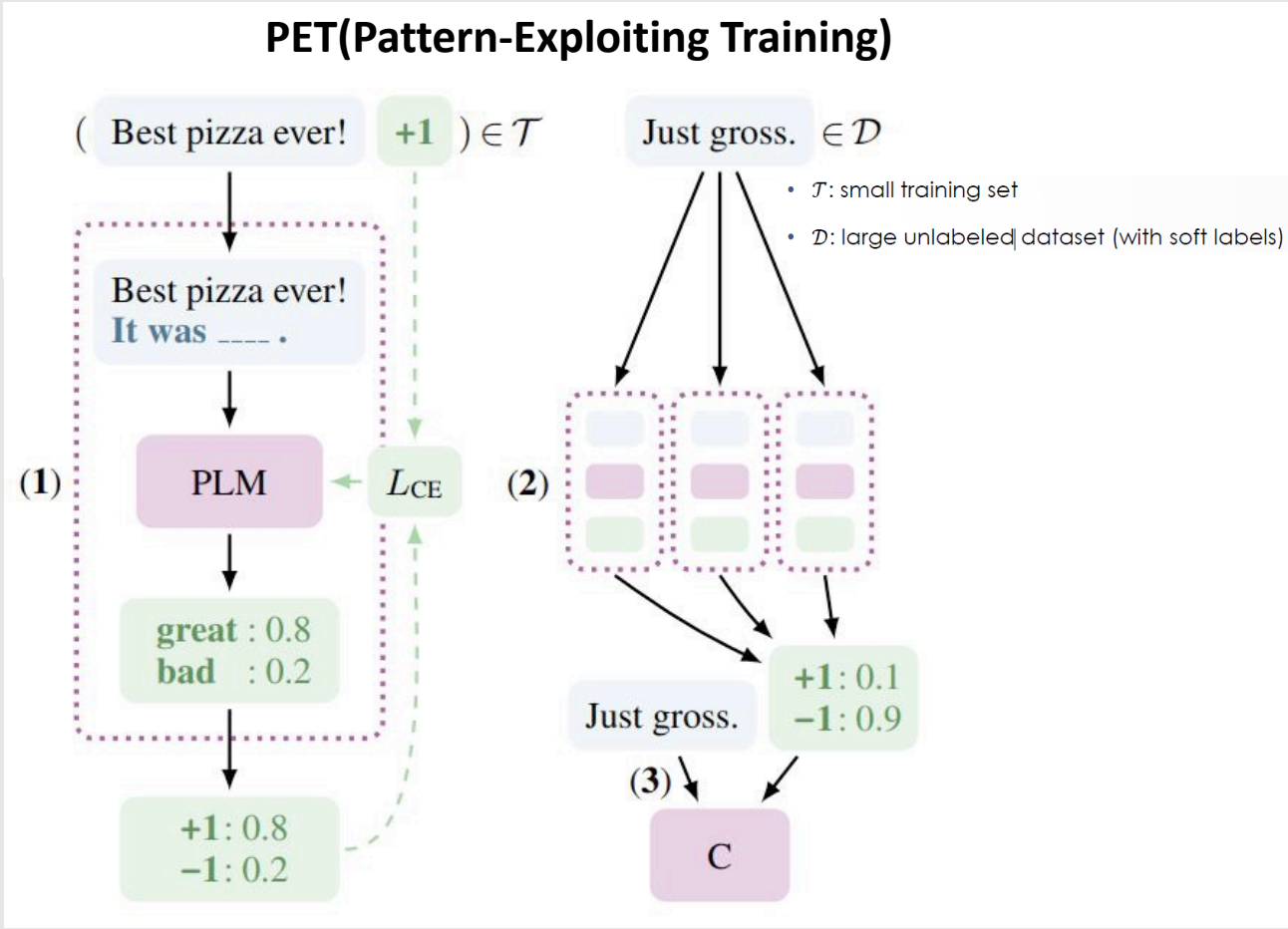
Prompt-based Learning: **PET(Pattern-Exploiting Training)**

Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference

It's Not Just Size That Matters:
Small Language Models Are Also Few Shot Learners
(두 논문 저자 동일)

Semi-supervised training procedure
Reformulate input examples into cloze-style phrases

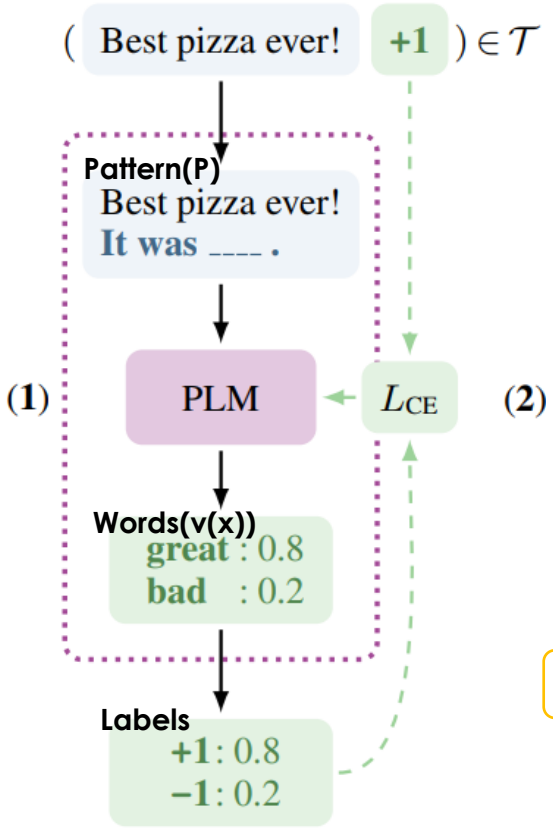


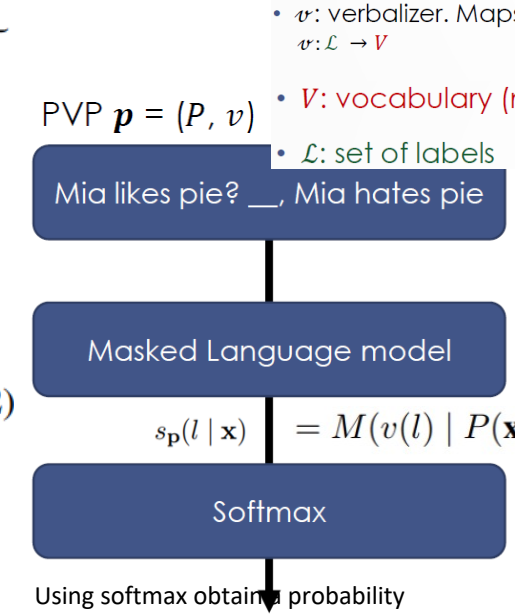


Prompt-based Learning: **PET(Pattern-Exploiting Training)**

Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference

Step1: Auxiliary LM





- P: pattern. Contains one mask token. $P(x) \in V^*$
- v: verbalizer. Maps each label to a word $v: \mathcal{L} \rightarrow V$
- V: vocabulary (mask token ___ $\in V$)
- $\mathcal{L}$: set of labels

(P, v): pattern-verbalizer pair (PVP)

Example1

x: input phrases $x = (s_1, \ldots, s_k)$ with $s_i \in V^*$

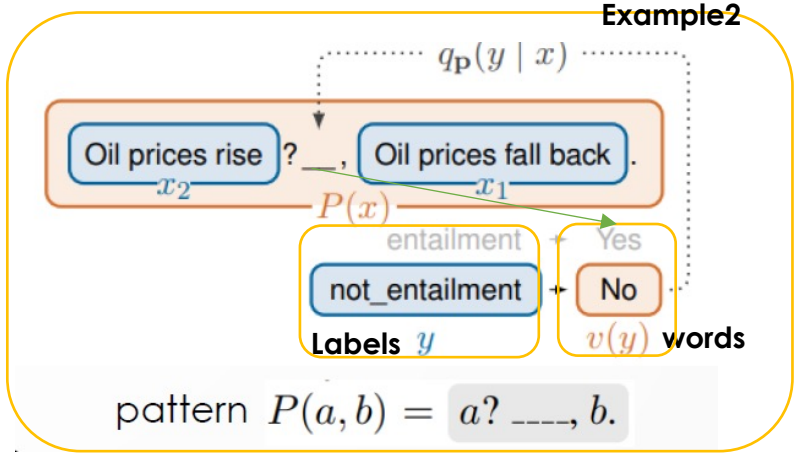
$\mathbf{x} = (\text{Mia likes pie, Mia hates pie})$

$P(\mathbf{x}) =$ Mia likes pie? ____, Mia hates pie.

Label y_0 : Yes, y_1: No

$v(y)$

Example2



- L_{CE} : Cross-entropy loss
- L_{MLM}: language modeling loss

True distribution(one-hot) of training example $(x, l) \in \mathcal{T}$

- $\mathcal{T}$: small training set

$$L = (1 - \alpha) \cdot L_{CE} + \alpha \cdot L_{MLM}$$

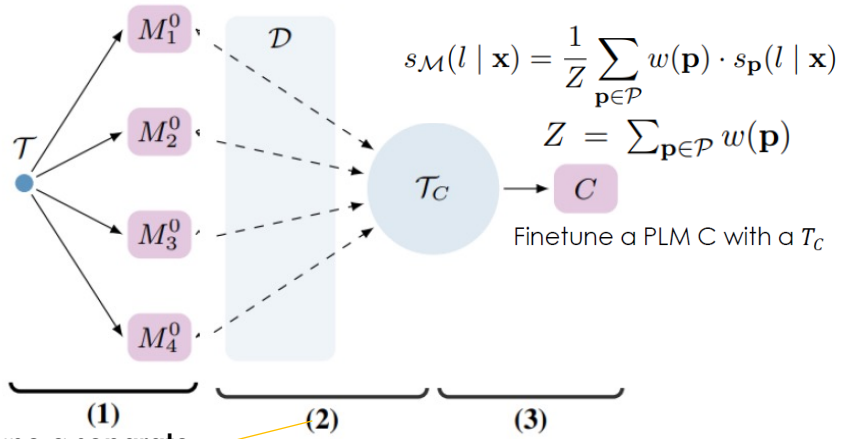
- By rules of thumb, α = 1E-04

Prompt-based Learning: **PET(Pattern-Exploiting Training)**

Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference

Step2: Combining PVPs

M_p (p: PVP) Use Ensemble $\mathcal{M} = \{M_p \mid p \in \mathcal{P}\}$.

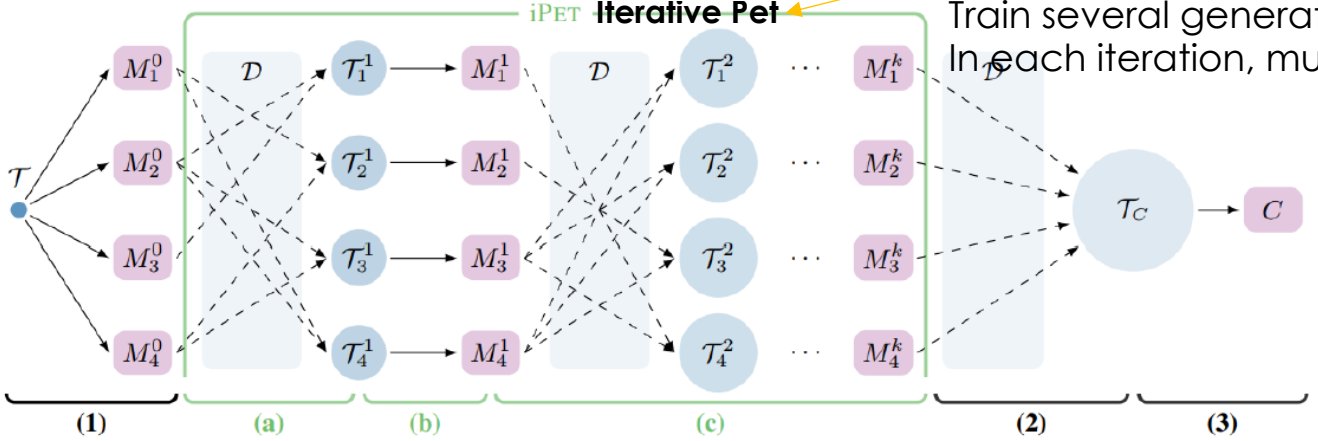


Finetune a **separate language model**

Cannot learn from each other models.

1. We obtain $\mathcal{N} \subset \mathcal{M}^{j-1} \setminus \{M_i^{j-1}\}$ by randomly choosing $\lambda \cdot (n-1)$ models from the previous generation with $\lambda \in (0, 1]$ being a hyperparameter.

iPET **Iterative Pet**



Train several generations of models on **datasets of increasing size**.
In each iteration, multiply the training set size by a constant 'd'

2. Using this subset, we create a labeled dataset

$$\mathcal{T}_{\mathcal{N}} = \{(\mathbf{x}, \arg\max_{l \in \mathcal{L}} s_{\mathcal{N}}(l \mid \mathbf{x})) \mid \mathbf{x} \in \mathcal{D}\}.$$

3. We define $\mathcal{T}_i^j = \mathcal{T} \cup \bigcup_{l \in \mathcal{L}} \mathcal{T}_{\mathcal{N}}(l)$. As can easily be verified, this dataset contains $c_j(l)$ examples for each $l \in \mathcal{L}$.

Prompt-based Learning:**P-tuning**

GPT-style models perform poorly for natural understanding (NLU) tasks

P-tuning: automatically search prompts in the **continuous space**

competitive performance to BERT => GPT understands, too

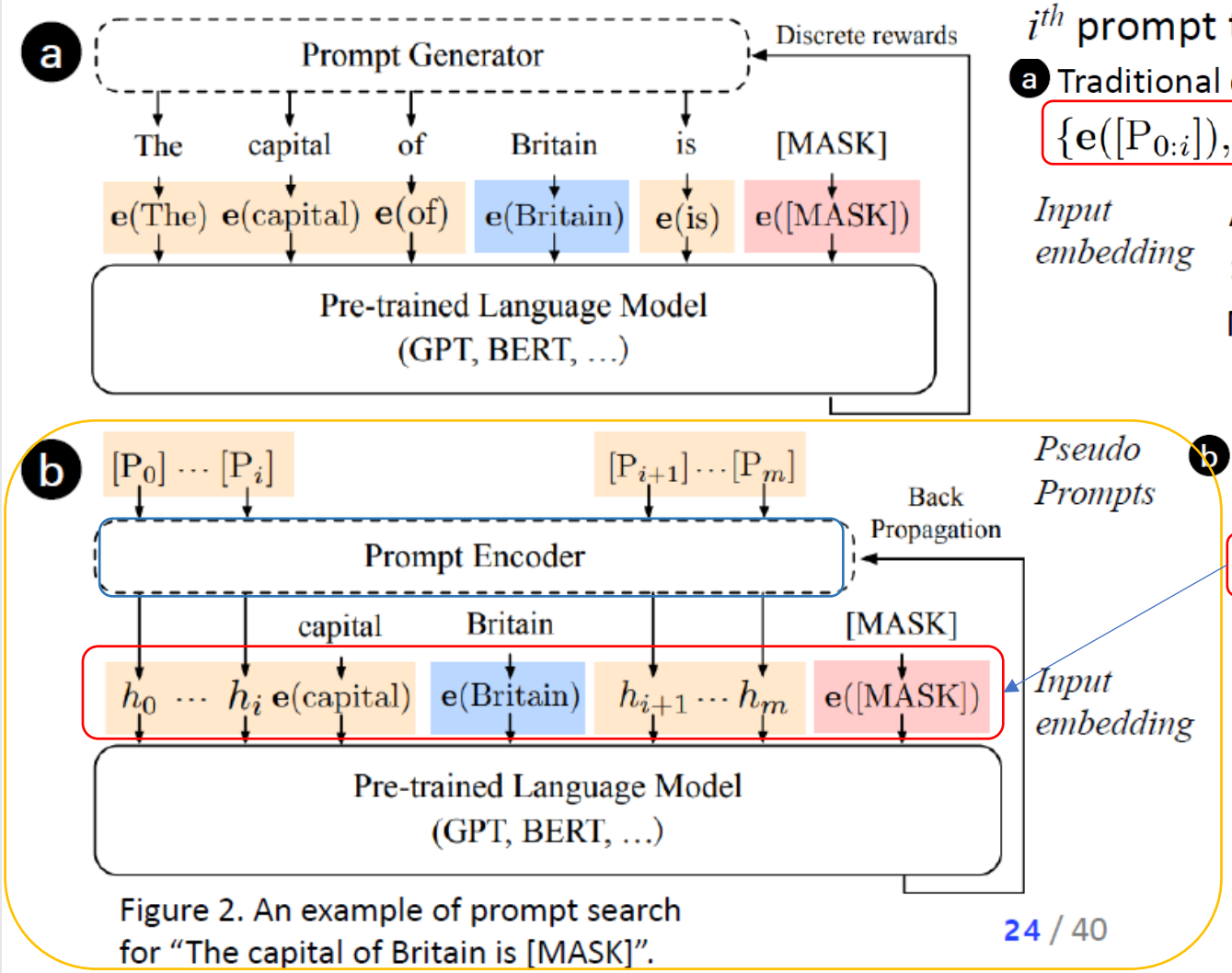


Figure 2. An example of prompt search for "The capital of Britain is [MASK]".

24 / 40

i^{th} prompt token in a Template T: $[\mathrm{P}_i]$ $T = \{[\mathrm{P}_{0:i}], \mathbf{x}, [\mathrm{P}_{i+1:m}], \mathbf{y}\}$

$[\mathrm{P}_i] \in \mathcal{V}$

The vocabulary of a language model: $\mathcal{V}$

a Traditional discrete prompts map the T into

$\{\mathbf{e}([\mathrm{P}_{0:i}]), \mathbf{e}(\mathbf{x}), \mathbf{e}([\mathrm{P}_{i+1:m}]), \mathbf{e}(\mathbf{y})\}$

Input embedding

A sequence of **discrete input tokens**

: $\mathbf{x}_{1:n} = \{x_0, x_1, ..., x_n\}$ Input embeddings: $\{\mathbf{e}(x_0), \mathbf{e}(x_1), ..., \mathbf{e}(x_n)\}$

Masked Token: $\mathbf{y}$

b P-tuning instead regards the $[\mathrm{P}_i]$ as **pseudo tokens** and map the template to

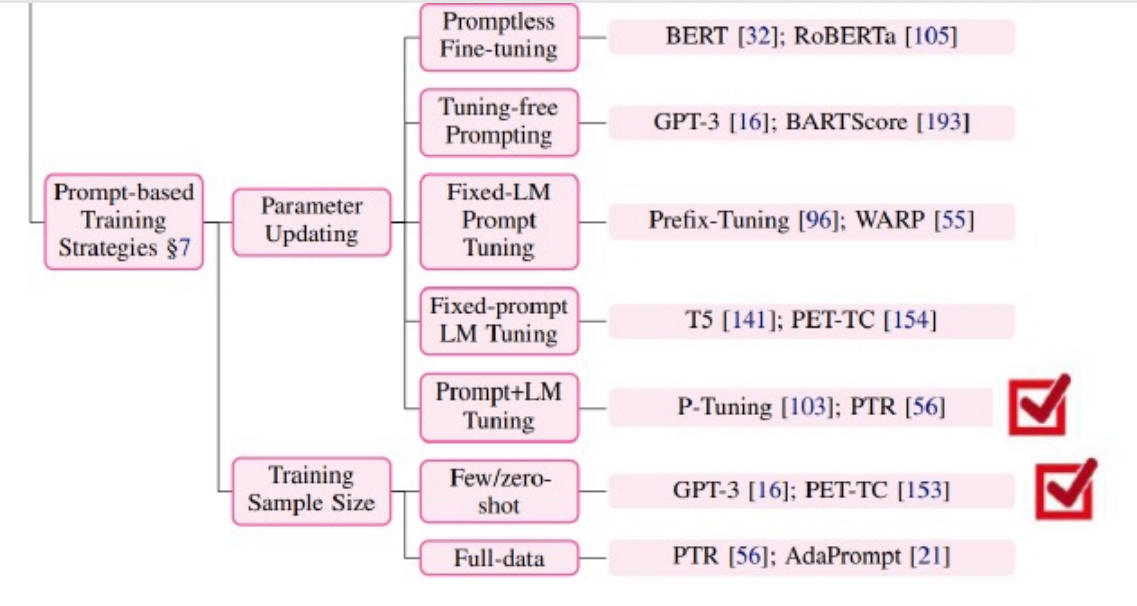
$\{h_0, ..., h_i, \mathbf{e}(\mathbf{x}), h_{i+1}, ..., h_m, \mathbf{e}(\mathbf{y})\}$

h_i are trainable embedding tensors

$$h_i = \mathrm{MLP}([\overrightarrow{h_i} : \overleftarrow{h_i}])$$
$$= \mathrm{MLP}([\mathrm{LSTM}(h_{0:i}) : \mathrm{LSTM}(h_{i:m})])$$

ReLU activated two-layer multilayer perceptron (MLP)

Prompt-based Learning:**P-tuning**



Pre-trained LM's **parameters are fixed.**
Experiments on two widely acknowledged NLU benchmarks
•**LAMA** (Language Model Analysis) knowledge probing
: P-tuning is comparable to or better than fine-tuning method.
•**SuperGLUE**
:GPT with P-tuning outperforms all the other BERT-based models on 5 out of 7 tasks. P-tuning appears to be more beneficial in low-resource settings. P-tuning outperforms both fully supervised settings and a few-shot settings.

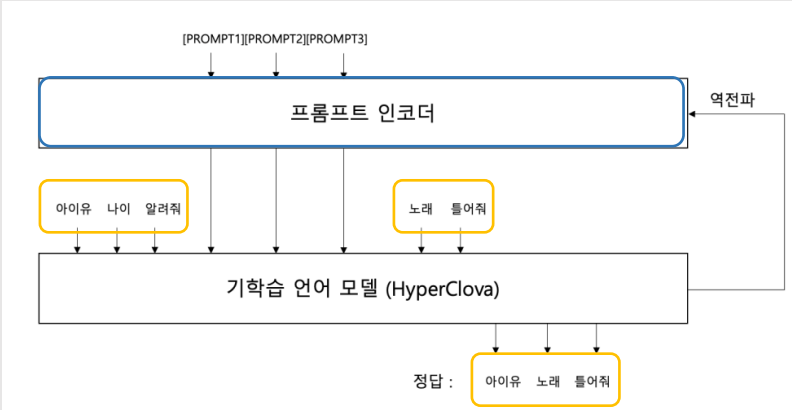


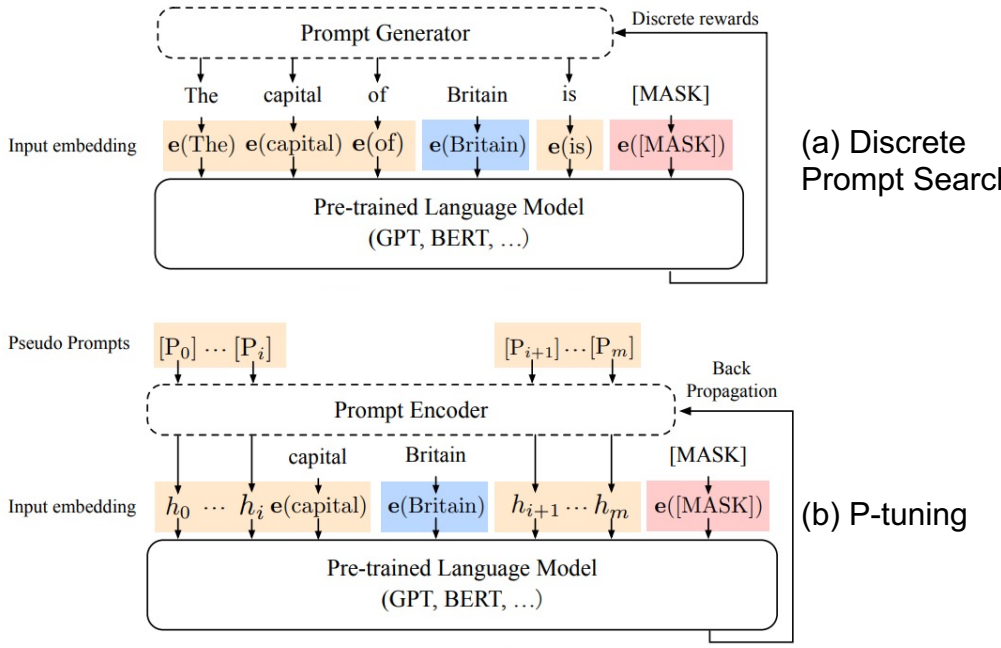
그림 3. P-tuning 학습 구조

대화형 음성 검색 시스템에 활용

GPT Understands, Too, 2021

https://arxiv.org/abs/2103.10385

자연어처리의 최신동향-3:
Transformer기반의 사전학습모델들

Motive	Researchers have observed that GPT-style models perform poorly for natural understanding tasks with fine-tuning, thus assumed that they are not suitable for language understanding in nature. The emerging GPT-3 and its particular performance on few-shot and zero-shot learning with hand-crafted prompts has swept the ML community. However, handcrafting a best-performance prompting is like finding a needle in a haystack.	***Keywords***	P-tuning, Automatically search prompts, Pseudo prompts, Anchor tokens, Prompt encoder
Introduction	• Giant models are too large to memorize the fine-tuning samples • A single-word's change in prompts can cause a drastic difference • P-tuning is a method to automatically search prompts in the continuous space to bridge the gap between GPTs and NLU applications		• P-tuning leverages **few continuous free parameters** to serve as prompts fed as the input to the pre-trained language models. We then optimize the continuous prompts using gradient descent as an alternative to discrete prompt searching.
Model	• An example of prompt search for "The capital of Britain is [MASK]". Given the context (blue zone, "Britain") and target (red zone, "[MASK]"), the orange zone refer to the prompt tokens. • In (a), the prompt generator only receives discrete rewards; on the contrary, in (b) the **pseudo prompts**, and prompt encoder can be optimized in a differentiable way. • Sometimes, adding few task-related **anchor tokens** (such as "capital" in (b)) will bring further improvement. • h_i (0 ≤ i < m) are trainable embedding tensors • Discreteness: the original word embedding e of the model has already become highly discrete after pre-training (the optimizer would easily fall into local minima) • Association: The values of prompt embeddings h_i should be dependent on each other rather than independent • In the P-tuning we propose to also model the h_i as a sequence using a **prompt encoder** consists of a very lite neural network that can solve the discreteness and association problems.		

GPT Understands, Too, 2021

https://arxiv.org/abs/2103.10385

자연어처리의 최신동향-3:
Transformer기반의 사전학습모델들

Model

- Prompt encoder: a bidirectional long-short term memory networks (LSTM), with a ReLU activated two-layer multilayer perceptron (MLP) to encourage discreteness.

$$h_i = \text{MLP}([\overrightarrow{h_i} : \overleftarrow{h_i}]) = \text{MLP}([\text{LSTM}(h_{0:i}) : \text{LSTM}(h_{i:m})])$$

- Because we want to evaluate knowledge gained from pre-training, pre-trained language models' parameters are fixed (i.e., not fine-tuned).

Dataset

LAMA (Knowledge probing),
LAMA-29k: the intersection of GPT and BERT's vocabulary
SuperGlue,

Results

- LAMA: P-tuning is comparable to or better than fine-tuning-based methods
- The surprising thing is that fine-tuning should have been more potent since it tunes all language models' parameters, while P-tuning not
- It is reasonable because, in terms of knowledge probing, many facts can only be hard-coded rather than inferenced by language models.
- The fine-tuning of parameters might result in catastrophic forgetting. On the contrary, P-tuning does not change the pre-trained models' parameters but evoke the stored knowledge by finding a better continuous prompt.

Model	MP	FT	MP+FT	P-tuning
BERT-base (109M)	31.7	51.6	52.1	52.3 (+20.6)
-AutoPrompt (Shin et al., 2020)	-	-	-	45.2
BERT-large (335M)	33.5	54.0	55.0	54.6 (+21.1)
RoBERTa-base (125M)	18.4	49.2	50.0	49.3 (+30.9)
-AutoPrompt (Shin et al., 2020)	-	-	-	40.0
RoBERTa-large (355M)	22.1	52.3	52.4	53.5 (+31.4)
GPT2-medium (345M)	20.3	41.9	38.2	46.5 (+26.2)
GPT2-xl (1.5B)	22.8	44.9	46.5	54.4 (+31.6)
MegatronLM (11B)	23.1	OOM*	OOM*	**64.2** (+41.1)

- SuperGLUE: P-tuning outperforms all the other BERT-based models on 5 out of 7 tasks. With P-tuning , GPT2 achieves comparable and even better performance as BERT-based models

Method	BoolQ (Acc.)	CB (Acc.)	CB (F1)	WiC (Acc.)	RTE (Acc.)	MultiRC (EM)	MultiRC (F1a)	WSC (Acc.)
BERT-base-cased (109M)								
Fine-tuning	72.9	85.1	73.9	71.1	68.4	16.2	66.3	63.5
MP zero-shot	59.1	41.1	19.4	49.8	54.5	0.4	0.9	62.5
MP fine-tuning	73.7	87.5	90.8	67.9	70.4	13.7	62.5	60.6
P-tuning	73.9	89.2	92.1	68.8	71.1	14.8	63.3	63.5
GPT2-base (117M)								
Fine-tune	71.2	78.6	55.8	65.5	67.8	17.4	65.8	63.0
MP zero-shot	61.3	44.6	33.3	54.1	49.5	2.2	23.8	62.5
MP fine-tuning	74.8	87.5	88.1	68.0	70.0	23.5	69.7	66.3
P-tuning	75.0 (+1.1)	91.1 (+1.9)	93.2 (+1.1)	68.3 (-2.8)	70.8 (-0.3)	23.5 (+7.3)	69.8 (+3.5)	63.5 (+0.0)

- Few-shot learning (32 train samples) on SuperGLUE dev shows that P-tuning consistently outperforms PET and PET best

Dev size	Method	BoolQ (Acc.)	CB (Acc.)	CB (F1)	WiC (Acc.)	RTE (Acc.)	MultiRC (EM)	MultiRC (F1a)
32	PET*	73.2±3.1	82.9±4.3	74.8±9.2	51.8±2.7	62.1±5.3	33.6±3.2	74.5±1.2
	PET best†	75.1	86.9	83.5	52.6	65.7	35.2	75.0
	P-tuning	77.8 (+4.6)	92.9 (+10.0)	92.3 (+17.5)	56.3 (+4.5)	76.5 (+14.4)	36.1 (+2.5)	75.0 (+0.5)

Code

- From the paper: https://github.com/THUDM/P-tuning

References

- Xu Han et al. (2021), Pre-trained Models: Past, present and Future, https://www.sciencedirect.com/science/article/pii/S2666651021000231
- Liu Pengfei et al. (2021), Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, https://arxiv.org/abs/2107.13586
- Yang Zhilin et al. (2019), XLNet:Generalized Autoregressive Pretraining for Language Understanding, https://arxiv.org/abs/1906.08237
- Du Zhenqxiao et al. (2022) GLM: General Language Model Pretraining with Autoregressive Blank Infilling, https://arxiv.org/abs/2103.10360
- Song Kaitao et al. (2019), MASS:Masked Sequence to Sequence Pre-training for Language Generation, https://arxiv.org/abs/1905.02450
- Lewis Mike et al. (2020), BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, https://aclanthology.org/2020.acl-main.703/
- Raffel Colin et al. (2020), Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, https://arxiv.org/abs/1910.10683
- Joshi Mandar et al. (2020), SpanBERT: Improving Pre-training by Representing and Predicting Spans, https://arxiv.org/abs/1907.10529
- Huang Hoyang et al. (2019), Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks, https://arxiv.org/abs/1909.00964
- Lu Jiasen et al. (2019), ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks, https://proceedings.neurips.cc/paper/2019/file/c74d97b01eae257e44aa9d5bade97baf-Paper.pdf
- Li Gen et al. (2019), Unicoder-VL: A Universal Encoder for Vision and Language for Vision and Language by Cross-modal Pre-training, https://arxiv.org/abs/1908.06066
- Ramesh Aditya et al. (2021), Zero-Shot Text-to-Image Generation, https://arxiv.org/abs/2102.12092
- Wang Xiaozhi et al. (2020), KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation, https://arxiv.org/abs/1911.06136
- Sun Yu et al. (2019), ERNIE: Enhanced Reprentation through Knowledge Integration, https://arxiv.org/abs/1904.09223
- Logan Robert et al. (2019), Barack's Wife Hillary: Using Knowledge Graphs for Fact-Aware Language Modeling, https://aclanthology.org/P19-1598/
- Micikevicius Paulius et al. (2018), Mixed Precision Training, https://arxiv.org/abs/1710.03740

References

- Shoeybi Mohammad et al. (2020), Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, https://arxiv.org/abs/1909.08053
- Zhang Minjia and Yuxiong He. (2020), Accelerating Training of Transformer-Based Language Models with Progressive Layer Dropping, https://arxiv.org/abs/2010.13369
- Sanh Victor et al. (2020), DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, https://arxiv.org/abs/2002.08307
- Wang Wenhui et al. (2020), MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers, https://proceedings.neurips.cc/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- Hewitt John and Christopher Manning. (2019), A Structural Probe for Finding Syntax in Word Representations, https://aclanthology.org/N19-1419/
- Liu Nelson et al. (2019), Linguistic Knowledge and Transferability of Contextual Representations, https://aclanthology.org/N19-1112/
- Jawahar Ganesh et al. (2019), What Does BERT Learn about the Structure of Language?, https://aclanthology.org/P19-1356/
- Clark Kevin et al. (2019), What Does BERT Look At? An Analysis of BERT's Attention, https://arxiv.org/abs/1906.04341
- Liu Xiao et al. (2021), GPT Understands, Too, https://arxiv.org/abs/2103.10385
- Schick Timo and Hinrich Scutze. (2021), Exploring Cloze Questions for Few Shot Text Classification and Natural Language Inference, https://aclanthology.org/2021.eacl-main.20.pdf
- Application of NLP and Computational Linguistics class at SNU(2022) Student Presentation Materials.