부스트캠프 AI Tech 2기

boostcamp ai tech

BERT, MT-DNN

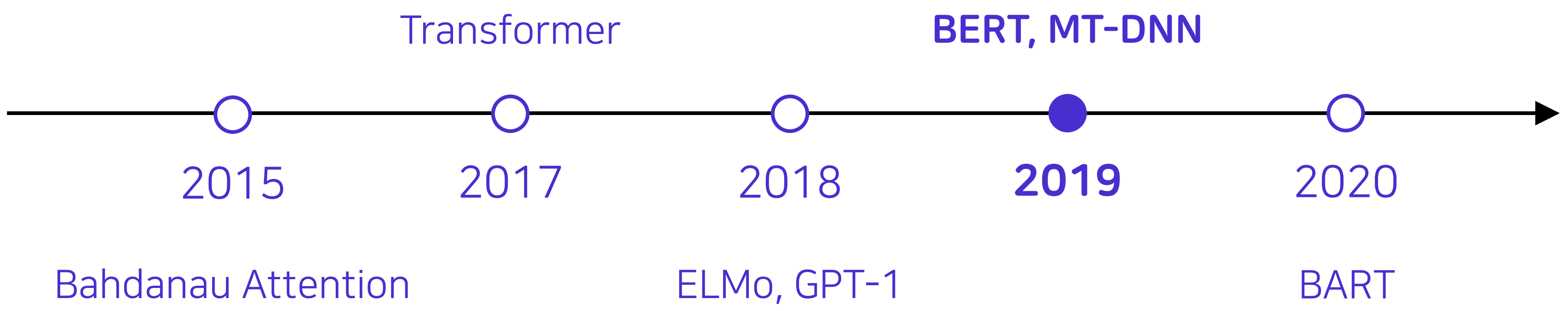
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL 2019
Multi-Task Deep Neural Networks for Natural Language Understanding, ACL 2019

Email : jinmang2@gmail.com

GitHub : github.com/jinmang2

Huggingface Hub: huggingface.co/jinmang2

Boostcamp AI Tech 2 NLP 논문 모임에 오신 여러분 환영합니다!



00. BERT vs ELMo, GPT-1

High-Level View

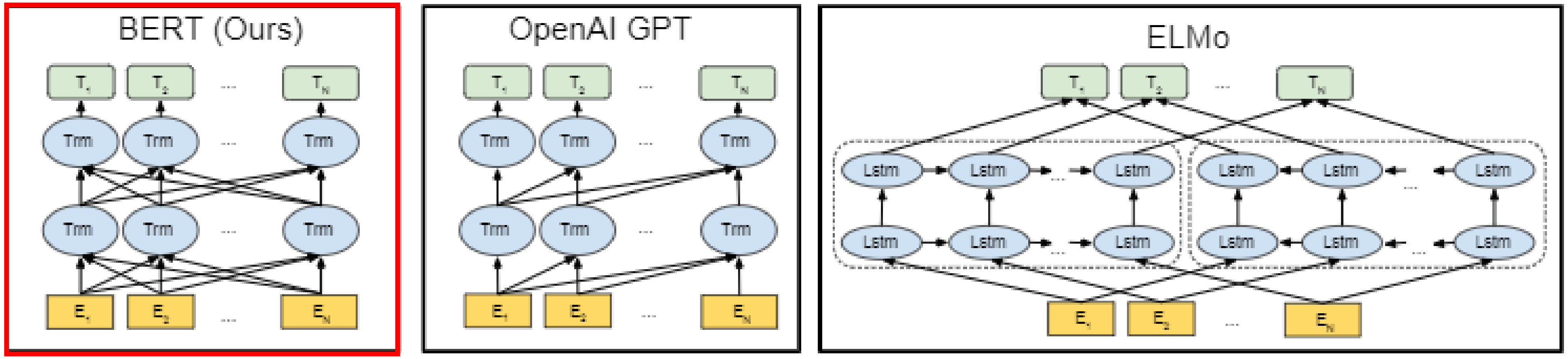
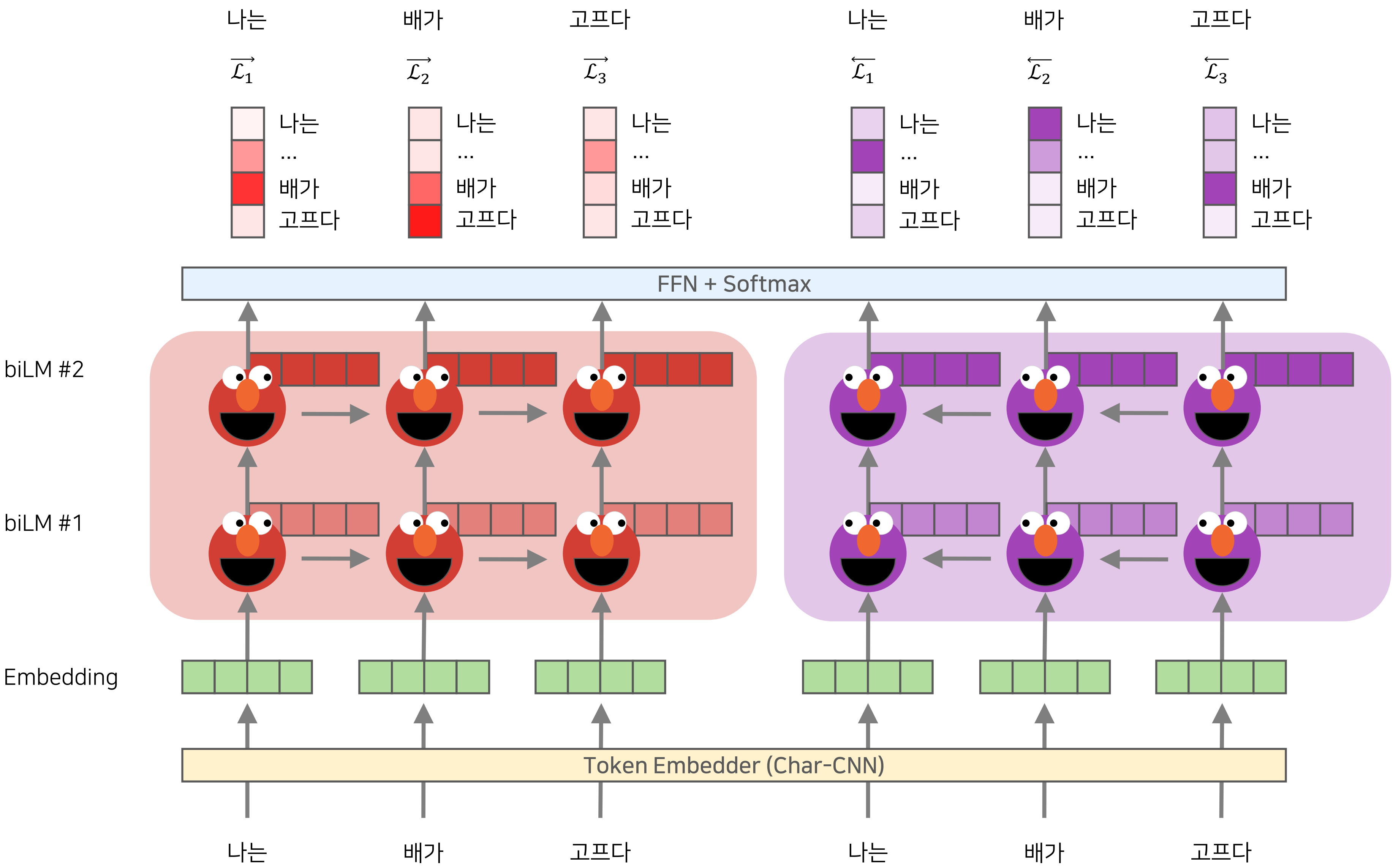


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

Bidirectional Encoder Representation from Transformer

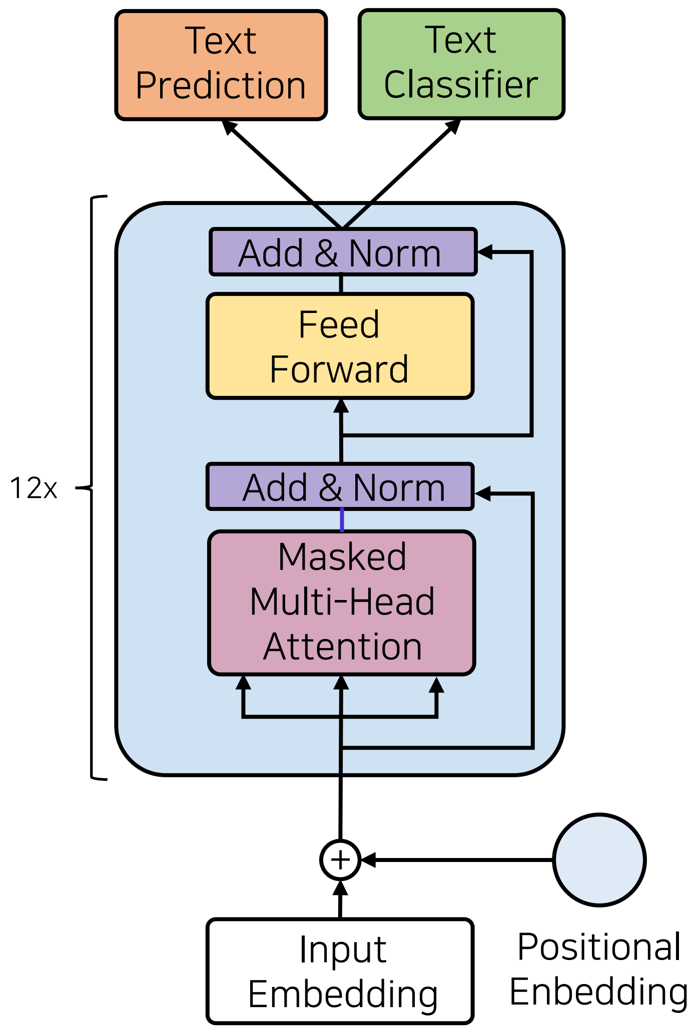
01. Review: ELMo

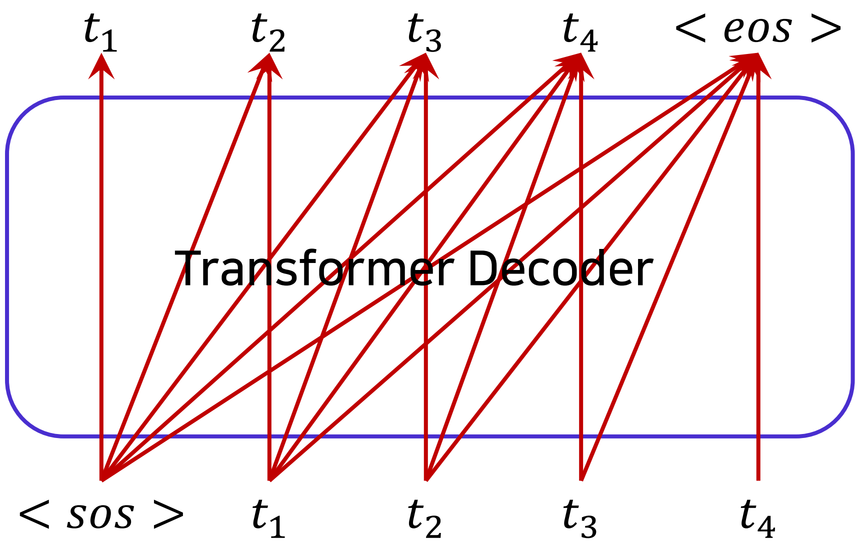
High-Level View



01. Review: GPT-1

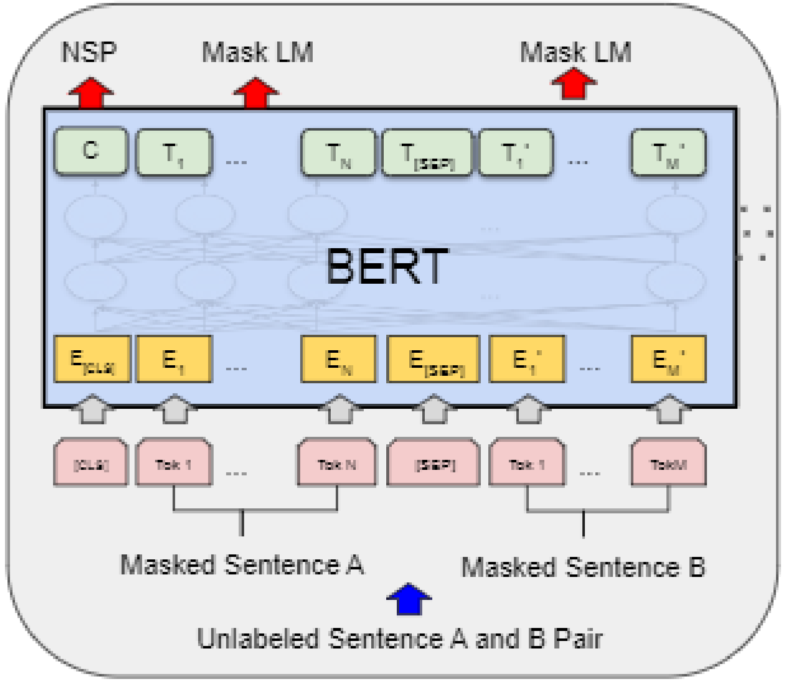
High-Level View



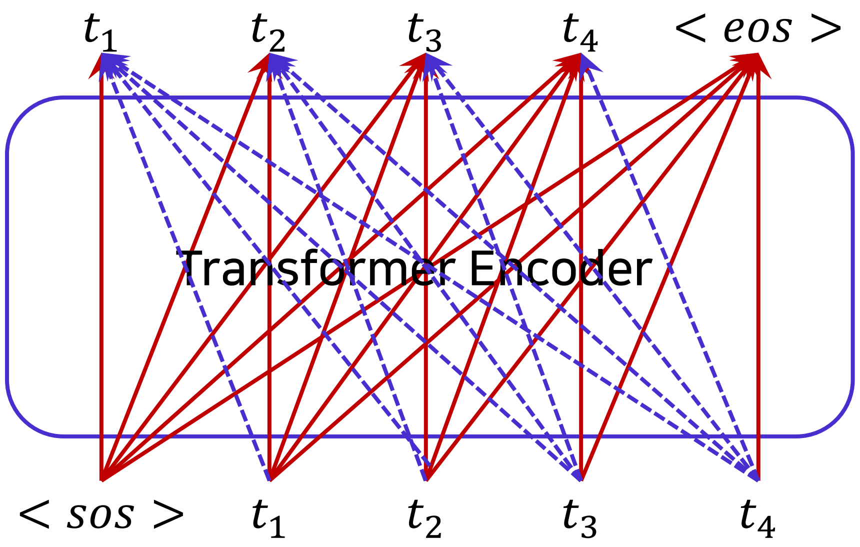


02. BERT: Bidirectional Encoder Representations from Transformer

High-Level View



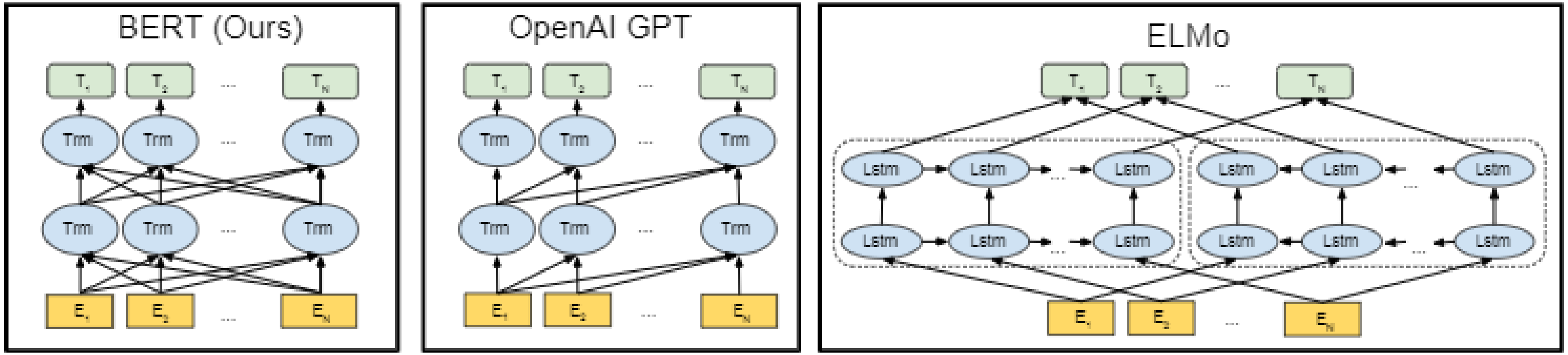
Pre-training



Forward LM과 Backward LM을 동시에 학습하는 것이 가능한가? Cheating인데...

02. BERT: Bidirectional Encoder Representations from Transformer

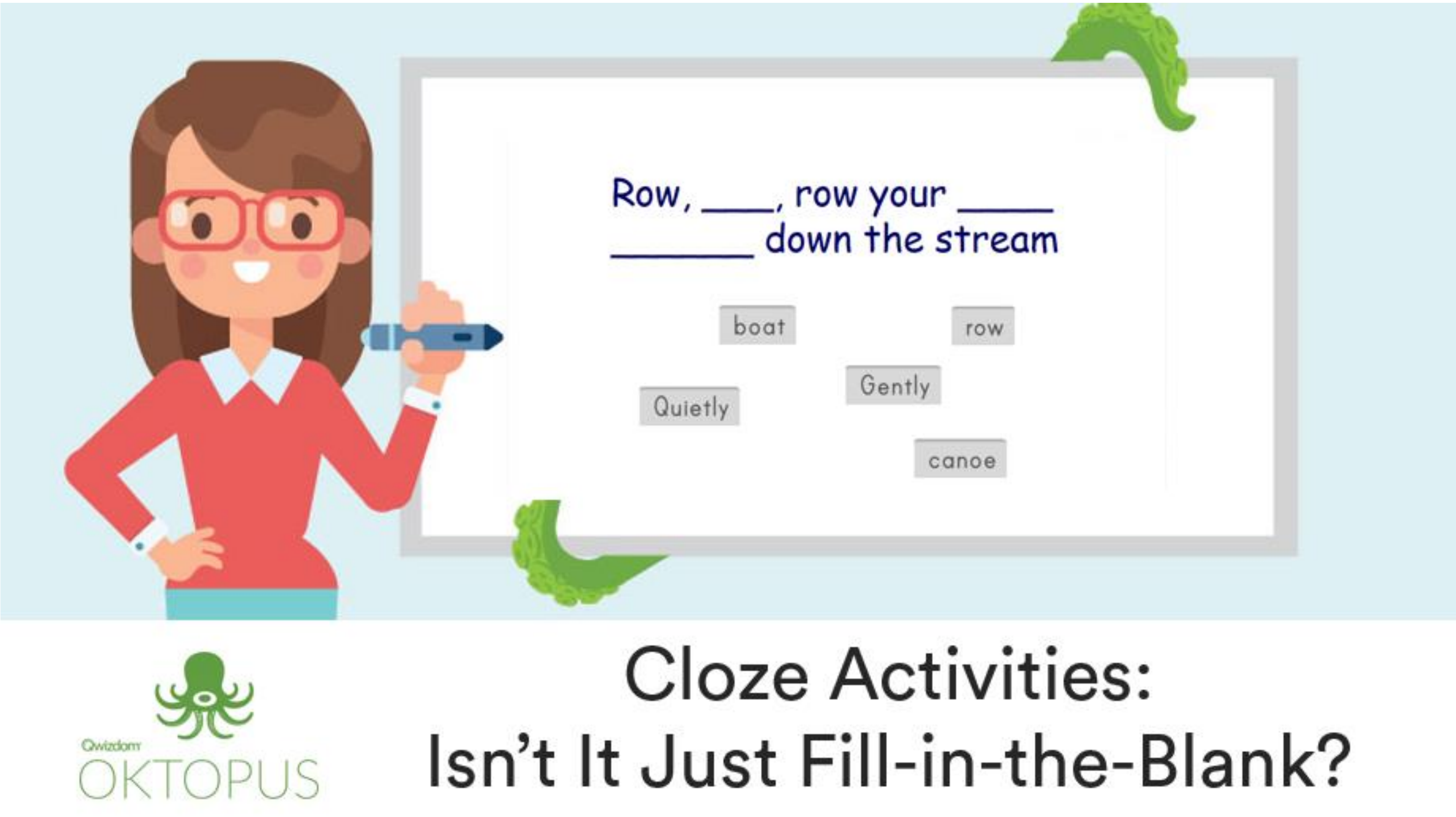
BERT vs ELMo vs GPT-1



Bidirectional repr	Unidirectional	Shallow concatenate Bidirectional
Fine-tuning	Fine-tuning	Feature-based
Single/Pair input form	Single/Pair input form	simple
BookCorpus + Wikipedia	BookCorpus	1B Word Benchmark
Transformer Encoder	Transformer Encoder	Char-CNN + biLM
Difference learning rate	Same learning rate	Difference learning rate
Better than GPT-1 and ELMo	Better than ELMo	Better than before

02. BERT: Bidirectional Encoder Representations from Transformer

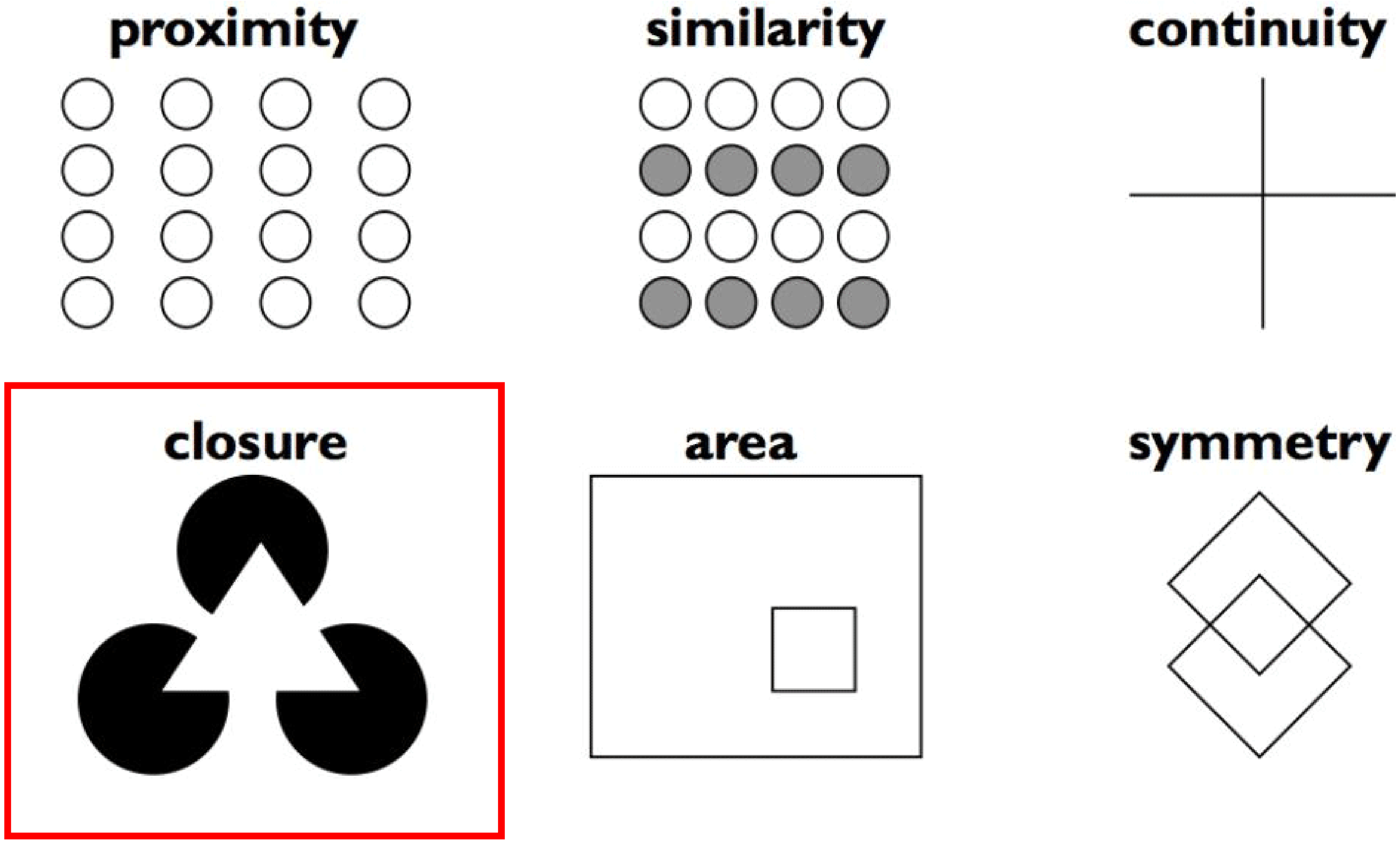
What is cloze procedure?



https://qwizdomoktopus.com/cloze-activities-interactive/

02. BERT: Bidirectional Encoder Representations from Transformer

What is cloze procedure?

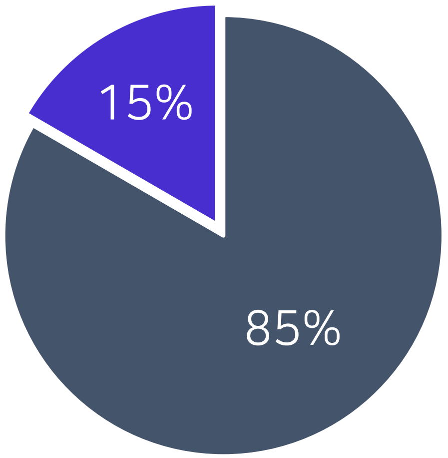


https://ditoday.com/%EC%83%81%ED%98%B8%EC%9E%91%EC%9A%A9-user-interface-%ED%95%A9%EC%9D%98-%EA%B0%80%EC%B9%98/

02. BERT: Bidirectional Encoder Representations from Transformer

Masked Language Modeling

```python
import torch
from typing import Tuple
from transformers import PreTrainedTokenizer


def mask_tokens(
    inputs: torch.Tensor,
    tokenizer: PreTrainedTokenizer,
    mlm_probability=0.15,
) -> Tuple[torch.Tensor, torch.Tensor]:
    if tokenizer.mask_token is None:
        raise ValueError(
            "This tokenizer does not have a mask token which is necessary for masked language modeling. Remove the -
        )

    labels = inputs.clone()
    # We sample a few tokens in each sequence for masked-LM trainign (with probability 0.15)
    probability_matrix = torch.full(labels.shape, mlm_probability)
    special_tokens_mask = [
        tokenizer.get_special_tokens_mask(val, already_has_special_tokens=True)
        for val in labels.tolist()
    ]
    probability_matrix.masked_fill_(torch.tensor(special_tokens_mask, dtype=torch.bool), value=0.0)
    if tokenizer._pad_token is not None:
        padding_mask = labels.eq(tokenizer.pad_token_id)
        probability_matrix.masked_fill_(padding_mask, value=0.0)
    masked_indices = torch.bernoulli(probability_matrix).bool()
    labels[~masked_indices] = -100 # We only compute loss on masked tokens

    # 80% of the time, we replace masked input tokens with tokenizer.mask_token ([MASK])
    indices_replaced = torch.bernoulli(torch.full(labels.shape, 0.8)).bool() & masked_indices
    inputs[indices_replaced] = tokenizer.convert_tokens_to_ids(tokenizer.mask_token)

    # 10% of the time, we replace masked input tokens with random word
    indices_random = torch.bernoulli(torch.full(labels.shape, 0.5)).bool() & masked_indices & ~indices_replaced
    random_words = torch.randint(len(tokenizer), labels.shape, dtype=torch.long)
    inputs[indices_random] = random_words[indices_random]

    # The rest of the time (10% of the time) we keep the masked input tokens unchanged
    return inputs, labels
```



80%	[MASK]
10%	random token
10%	hold original token

02. BERT: Bidirectional Encoder Representations from Transformer

Fine-Tuning

이런 건 repo를 직접 뜯어보면서 봐야해요!

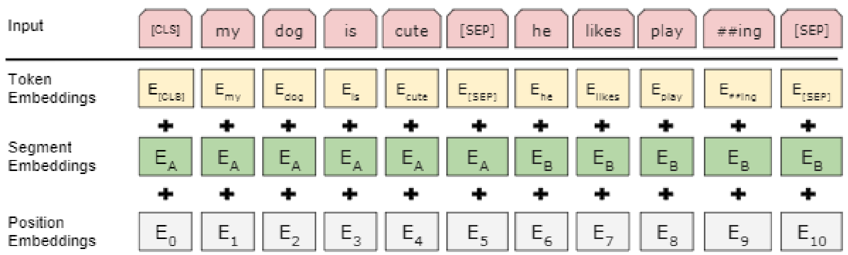
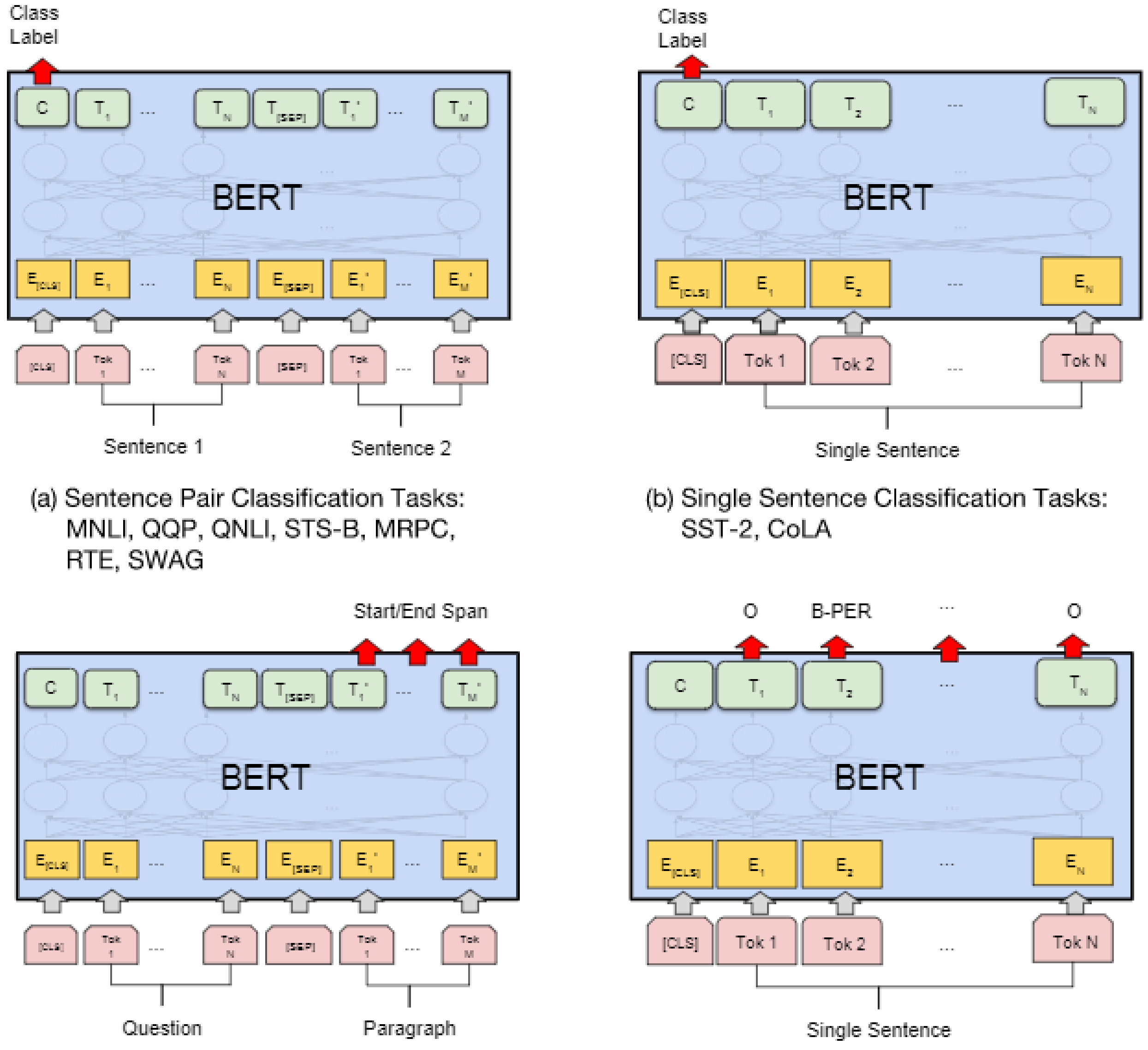


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.



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

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

(c) Question Answering Tasks: SQuAD v1.1

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

02. BERT: Bidirectional Encoder Representations from Transformer

Performance

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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

02. BERT: Bidirectional Encoder Representations from Transformer

Performance

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$BERT_{BASE}$ (Single)	80.8	88.5	-	-
$BERT_{LARGE}$ (Single)	84.1	90.9	-	-
$BERT_{LARGE}$ (Ensemble)	85.8	91.8	-	-
$BERT_{LARGE}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$BERT_{LARGE}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$BERT_{LARGE}$ (Single)	78.7	81.9	80.0	83.1

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

02. BERT: Bidirectional Encoder Representations from Transformer

Performance

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$BERT_{BASE}$	81.6	-
$BERT_{LARGE}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

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

Table 5: Ablation over the pre-training tasks using the $BERT_{BASE}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

02. BERT: Bidirectional Encoder Representations from Transformer

Performance

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

02. BERT: Bidirectional Encoder Representations from Transformer

Performance

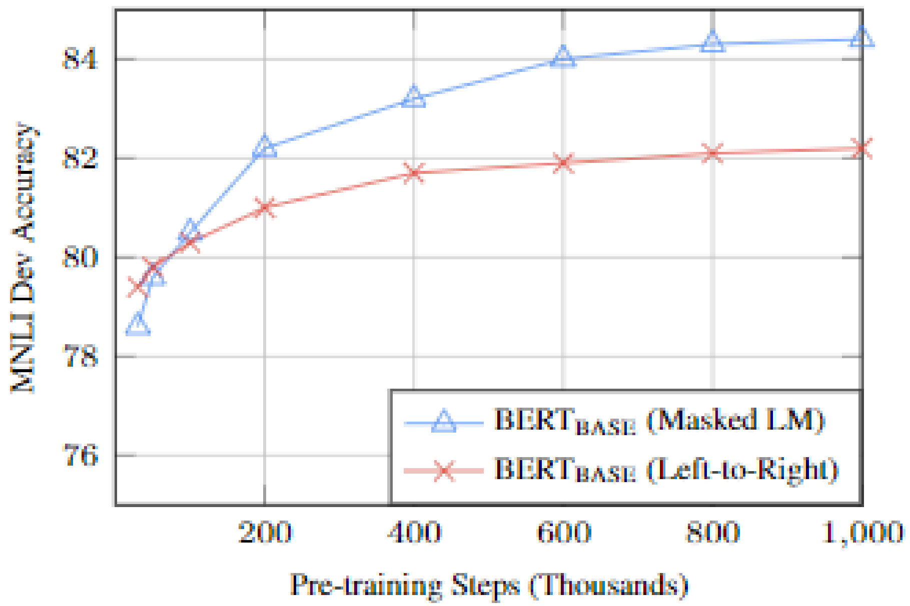


Figure 5: Ablation over number of training steps. This shows the MNLI accuracy after fine-tuning, starting from model parameters that have been pre-trained for k steps. The x-axis is the value of k.

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

Table 8: Ablation over different masking strategies.

02. BERT: Bidirectional Encoder Representations from Transformer

Catastrophic Forgetting

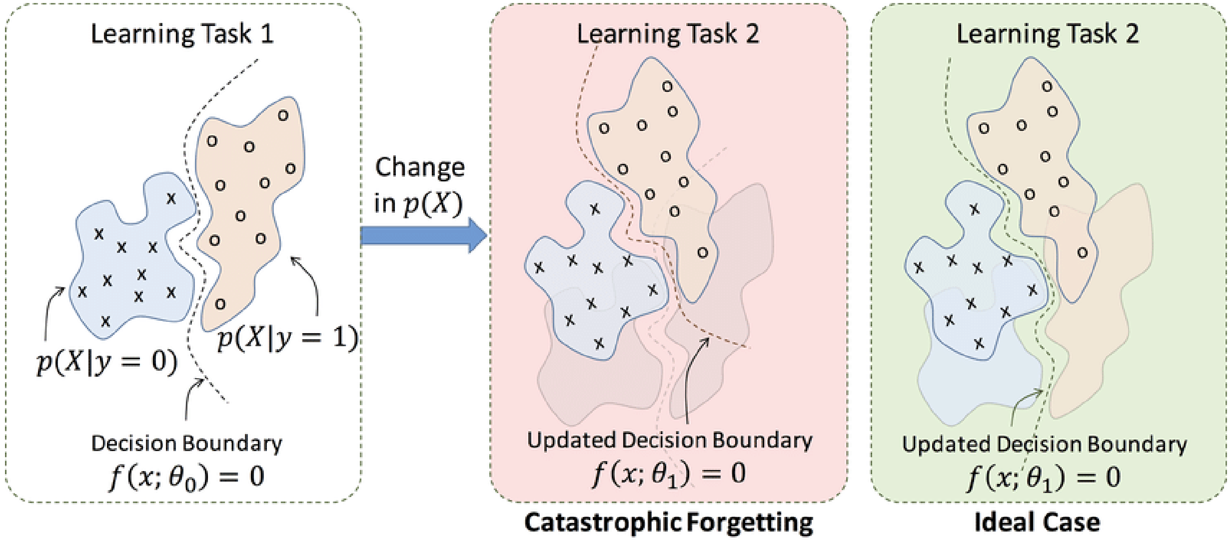
BERT 논문에서 각 task를 어떻게 학습했는지 살펴보면
아래와 같은 특징이 있다는 것을 확인할 수 있다.

- 3 Epochs
- Small Learning Rate

그리고 BERT-Large의 경우 성능이 좋지 않아서 Random Restart 후 성능이 좋은 친구를 결과로 사용했다는 보고도 있다.

왜 이렇게 해줄까? 찾아보니 Catastrophic Forgetting이라는 키워드를 찾을 수 있었다!

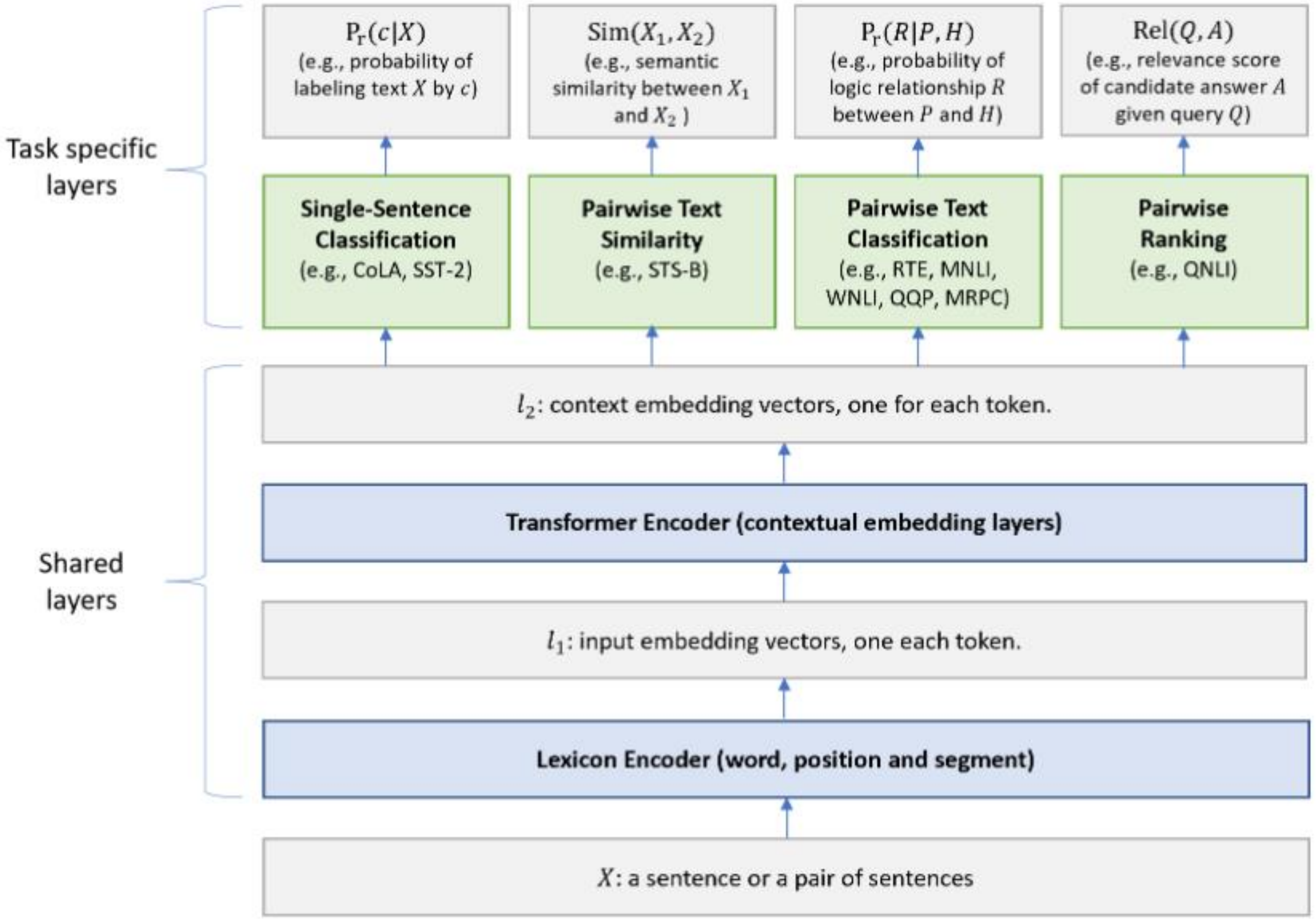
여기에 초기 BERTAdam이 Bias Compensation을 안해줘서 그런 경향도 있었다고 하네요! TMI ㅎㅎ

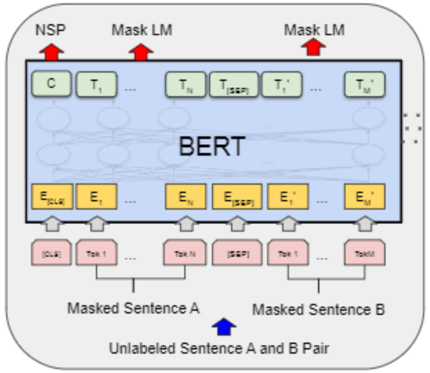


https://datascience.stackexchange.com/questions/49313/how-does-bert-deal-with-catastrophic-forgetting
https://www.researchgate.net/figure/Depiction-of-catastrophic-forgetting-in-binary-classification-tasks-when-there-is-a_fig1_331609030

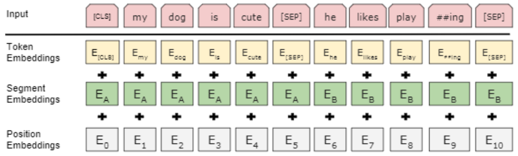
03. MT-DNN: Multi-Task Deep Neural Networks for Natural Language Understanding

BERT + Multi-Task Learning





Pre-training



03. MT-DNN: Multi-Task Deep Neural Networks for Natural Language Understanding

What is Multi-Tasking Learning?

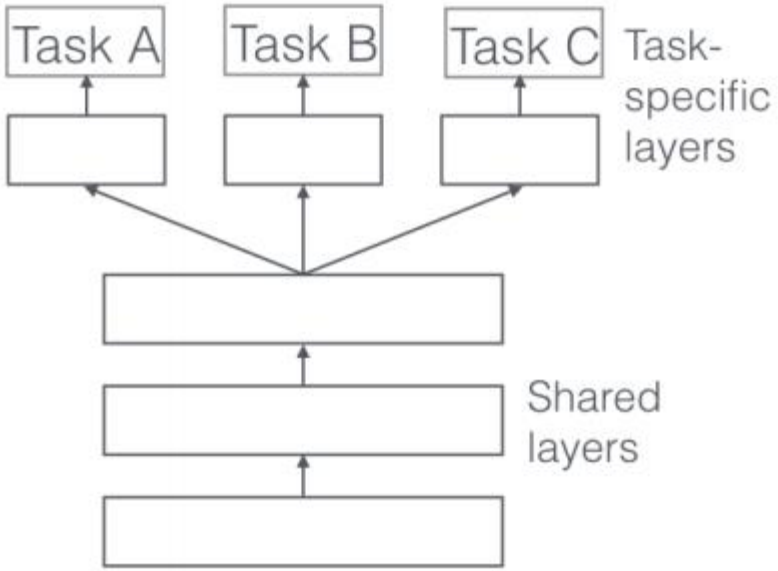


Figure 1: Hard parameter sharing for multi-task learning in deep neural networks

동시에 여러 문제를 푸는 것!

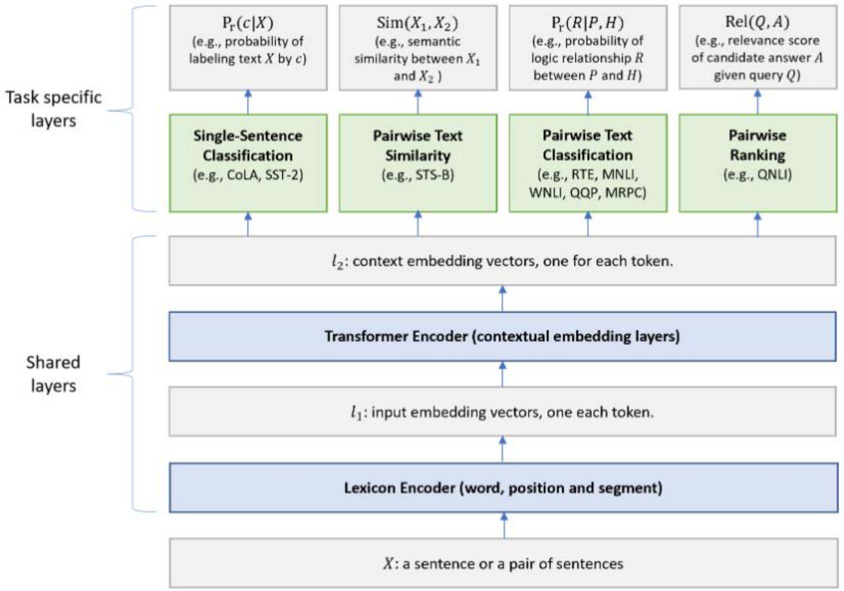
03. MT-DNN: Multi-Task Deep Neural Networks for Natural Language Understanding

What is Multi-Tasking Learning?

Algorithm 1: Training a MT-DNN model.

Initialize model parameters Θ randomly.
Pre-train the shared layers (i.e., the lexicon encoder and the transformer encoder).
Set the max number of epoch: $epoch_{max}$.
//Prepare the data for T tasks.
for t *in* $1, 2, ..., T$ **do**
  Pack the dataset t into mini-batch: D_t.
end
for $epoch$ *in* $1, 2, ..., epoch_{max}$ **do**
  1. Merge all the datasets:
  $D = D_1 \cup D_2 ... \cup D_T$
  2. Shuffle D
  for b_t *in* D **do**
    //b_t is a mini-batch of task t.
    3. Compute loss : $L(\Theta)$
      $L(\Theta) =$ Eq. 6 for classification
      $L(\Theta) =$ Eq. 7 for regression
      $L(\Theta) =$ Eq. 8 for ranking
    4. Compute gradient: $\nabla(\Theta)$
    5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$
  end
end

- Classification Task는 Cross Entropy loss 사용
  - $-\sum_c 1(X, c) \log(P_c(c|X))$
- Regression Task는 MSE 사용
  - $(y - \mathrm{Sim}(X_1, X_2))^2$
- Ranking Task는 pairwise learning-to-rank paradigm 사용
  - Positive example의 NLL Loss를 최소화
  - $-\sum_{(Q,A^+)} P_r(A^+|Q)$
  - $P_r(A^+|Q) = \frac{\exp(\gamma \mathrm{Rel}(Q, A^+))}{\sum_{A' \in \mathcal{A}} \exp(\gamma \mathrm{Rel}(Q, A'))}$
  - γ는 1로 세팅



03. MT-DNN: Multi-Task Deep Neural Networks for Natural Language Understanding

What is Multi-Tasking Learning?

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST-2	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
WNLI	NLI	634	71	146	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr
Relevance Ranking (GLUE)						
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Pairwise Text Classification						
SNLI	NLI	549k	9.8k	9.8k	3	Accuracy
SciTail	NLI	23.5k	1.3k	2.1k	2	Accuracy

4.2 Implementation Details

- PyTorch로 구현 (정확히는 킹갓 허깅페이스)
- Adamax Optimizer
  - learning rate 5e-05
  - batch size 32
- maximum epoch 5
- linear learning rate decay scheduler with warm-up 0.1
- task-specific layer에 dropout 0.1
  - MNLI엔 0.3, CoLa엔 0.05
- gradient clipping 1.
- wordpieces tokenizer 사용
- max tokens 512

03. MT-DNN: Multi-Task Deep Neural Networks for Natural Language Understanding

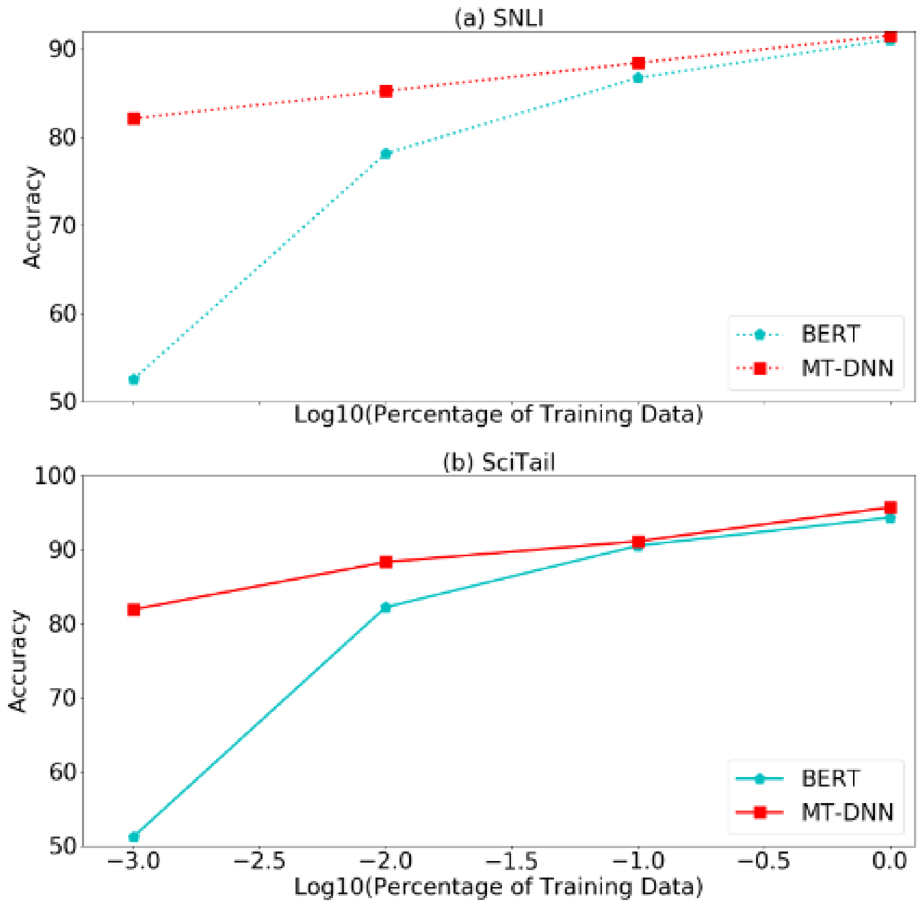
What is Multi-Tasking Learning?

Model	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	**Score**
BiLSTM+ELMo+Attn [1]	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4/76.1	-	56.8	65.1	26.5	70.5
Singletask Pretrain Transformer [2]	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	-	56.0	53.4	29.8	72.8
GPT on STILTs [3]	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.8/80.6	-	69.1	65.1	29.4	76.9
$\text{BERT}^{4}_{\text{LARGE}}$	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
$\text{MT-DNN}_{\text{no-fine-tune}}$	58.9	94.6	**90.1/86.4**	89.5/88.8	**72.7/89.6**	86.5/85.8	**93.1**	79.1	65.1	39.4	81.7
MT-DNN	**62.5**	**95.6**	**91.1/88.2**	**89.5/88.8**	**72.7/89.6**	**86.7/86.0**	**93.1**	**81.4**	65.1	**40.3**	**82.7**
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1

Model	MNLI-m/mm	QQP	RTE	QNLI (v1/v2)	MRPC	CoLa	SST-2	STS-B
$\text{BERT}_{\text{LARGE}}$	86.3/86.2	91.1/88.0	71.1	90.5/92.4	89.5/85.8	61.8	93.5	89.6/89.3
ST-DNN	86.6/86.3	91.3/88.4	72.0	96.1/-	89.7/86.4	-	-	-
MT-DNN	**87.1/86.7**	**91.9/89.2**	**83.4**	**97.4/92.9**	**91.0/87.5**	**63.5**	**94.3**	**90.7/90.6**

03. MT-DNN: Multi-Task Deep Neural Networks for Natural Language Understanding

What is Multi-Tasking Learning?



Model	0.1%	1%	10%	100%
SNLI Dataset (Dev Accuracy%)				
#Training Data	549	5,493	54,936	549,367
BERT	52.5	78.1	86.7	91.0
MT-DNN	82.1	85.2	88.4	91.5
SciTail Dataset (Dev Accuracy%)				
#Training Data	23	235	2,359	23,596
BERT	51.2	82.2	90.5	94.3
MT-DNN	81.9	88.3	91.1	95.7

부스트캠프 AI Tech 2기

Discussion

Email : jinmang2@gmail.com
GitHub : github.com/jinmang2
Huggingface Hub: huggingface.co/jinmang2

진명훈_T2216