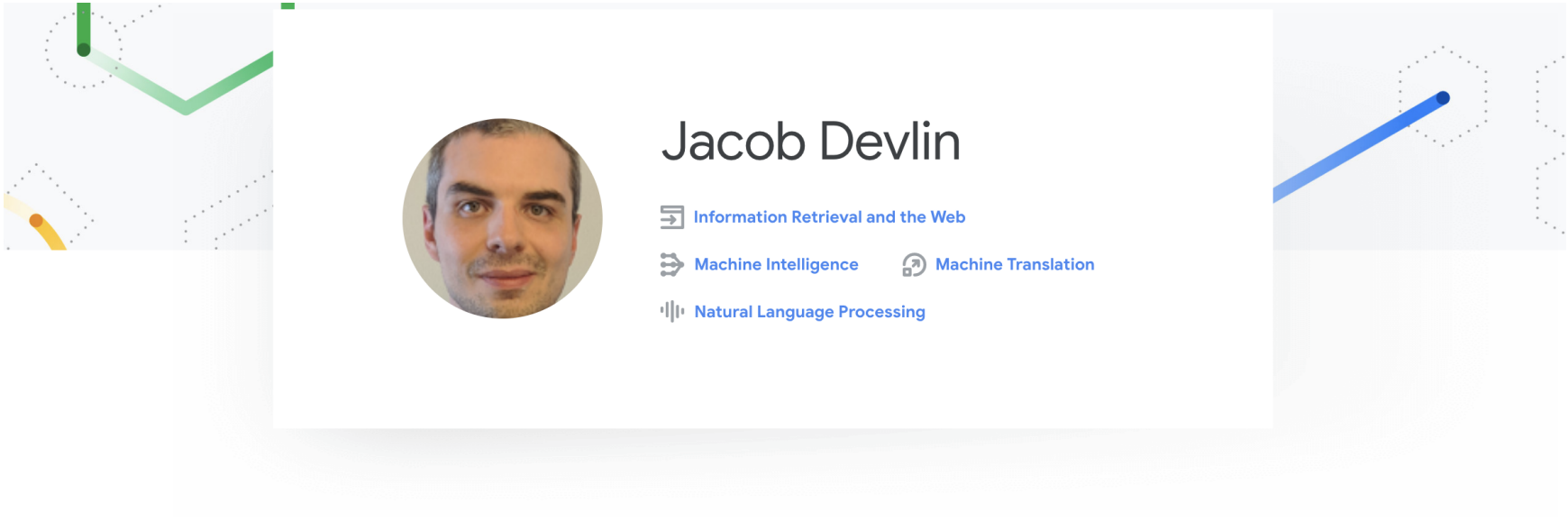
BERT- Pre-training of Deep Bidirectional Transformers for Language Understanding

- 저자:Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (**Google AI Language**)

Who is an Author?

Jacob Devlin is a **Senior Research Scientist at Google**. At Google, his primary research interest is developing fast, powerful, and scalable deep learning models for information retrieval, question answering, and other language understanding tasks. From 2014 to 2017, he worked as a Principle Research Scientist at Microsoft Research, where he led Microsoft Translate's transition from phrase-based translation to neural machine translation (NMT). He also developed state-of-the-art on-device models for mobile NMT. Mr. Devlin was the recipient of the ACL 2014 Best Long Paper award and the NAACL 2012 Best Short Paper award. He received his Master's in Computer Science from the University of Maryland in 2009, advised by Dr. Bonnie Dorr.



느낀점

- Masking 기반의 Language Model과 context 추출을 위한 문장 연관성 (NSP) Task를 동시에 학습시켜서 rich representation을 얻는다는 아이디어가 참신했음. 두마리 토끼를 한꺼번에..!
- Bidirectional feature가 상당히 중요함
- pre-train 중요함
- NSP도 매우 중요함
- 여기서도 Loss Masking이 중요함
- NSP Loss와 LM Loss를 따로 떼서 계산해야함
- gelu, masking scheme 썼을때와 안썼을때 성능차이가 꽤 남
- segment embedding 처리하는게 은근 귀찮음, 전처리 할때 아예 생성해버리는게 편하긴함
- NSP acc 올리기보다 LM acc 올리는게 더 쉬움

Abstract

- **BERT**는 **B**idirectional **E**ncoder **R**epresentations from **T**ransformers의 약자임
- 최근의 language representation models과 다르게 BERT는 Bidirectional representations을 pre-train하기 위해 디자인됨
- 결과적으로는 pre-trained BERT representations는 단지 output layer 한개만 추가해도 다양한 영역에서 SOTA를 찍는 fine-tune이 가능함
- 11개의 NLP Task에서 new SOTA기록하고, SQuAD v1.1 QA에서 사람보다 2.0 높은 F1 성능 기록함
    - GLUE benchmark에서 **80.4%** 달성함 기존 것보다 **7.6%** 향상시킴

1. Introduction

- pre-trained LM은 예로부터 NLP의 성능을 올리기에 효과적인 방법이었음
- Pre-trained Language Representation을 적용하는데는 2가지 전략이 있음 (*feature-based* and *fine-tuning*)
- **feature-based: ELMo** (Peters al., 2018), 특정 아키텍처를 사용하는데 이때, pre-trained representation이 addtional features로 얻어짐
- **fine-tuning: GPT** (Generative Pre-trained Transformer) (Radford et al., 2018; OpenAI)
- 기존 연구에선 두 접근법 모두 같은 objective function을 사용함; pre-trainining시에 **unidirectional LM**이 language representation을 학습하기 위해 쓰는 objective function
- 본 연구에서는 그러한 현재의 기법이 특별히 fine-tuning approach에서는 pre-trained representation의 power를 매우 제한(**serverly restrict**)한다고 주장함
- 주된 한계는 Standard LM이 unidirectional하다는 것임. 이는 아키텍처의 선택을 제한하게됨.
- 예를들면, OpenAI의 GPT의 경우 left-to-right 구조로써, self-attetnion에서 모든 토큰들이 previous token에만 attention을 갖게됨 (Vaswani et al., 2017)

1. Introduction

- 이러한 제한들은 sentence level에서 sub-optimal에 도달할 수 밖에 없게 함(SQuAD같은 token-level task에서 이러한 구조의 fine-tuning은 안좋을 수 있음(could be devastating))
- 결론: Bidirectional 하게 해야함. `it is crucial to incorporate context from both directions`
- 본 논문에서는 fine-tuning based approach를 BERT를 제안함으로써 개선시킴! (현재로썬 살짝 GPT에 숟가락 얹은것 같기도..)
- BERT에서는 기존에 비판했던 objective function을 쓰진 않음(left-to-right 구조에 dependent했던). 대신에 **MLM(Masked Language Model**; Taylor, 1953)의 objective function을 사용함
- MLM은 랜덤하게 input token의 일부를 masking처리 후 그 부분을 예측하는 것을 목표로함.
- `MLM objective allows the representation to fuse the left and the right context` (해석보단 원문으로)
- 본 논문의 contribution은
  - Bidirectional pre-training for LM by MLM (maksed language models)
  - task-specific architecture에 대한 model engineering 안해도됨. BERT는 fine-tuning based representation model로는 sentence-level, token-level tasks에서 첫번째로 SOTA 찍은 모델임
  - 11개의 NLP Task에서 SOTA 찍었음. 코드도 공개함(https://github.com/google-research/bert)

2. Related work

2.1. Feature-based Approaches

- non-neural과 neural(word2vec)한 방법으로 나뉨
- pre-trained word embedding은 learned from scratch로부터 얻은 embedding보다 확연히 개선된 결과를 보였었음
- ELMo는 traditional word embeddign research를 different dimension에 따라 일반화시킴
- ELMo는 context-sensitive features를 LM으로부터 추출함
- contextual word embedding과 task-specific architectures의 결합으로 ELMo는 여러 NLP task(QA on SQuAD, SA, NER)에서 SOTA를 기록함

2. Related work

2.2. Fine-tuning Approaches

- 최근 트렌드라고 할 수 있음, LM에 transfer learning을 적용하는 것임
- LM Objective에 대해서 pre-training 후에 fine-tuning하는 것임
- 장점중 하나는 few parameter만 다시 learning이 필요하다는것임
- 이러한 기법을 사용한 OpenAI GPT가 GLUE bechmark에서 SOTA 찍었었음 (Wang et al., 2018)

2. Related work

2.3. Transfer Learning from Supervised Data

- unsupervised pre-training의 장점은 거의 unlimited한 data를 쓸 수 있다는 것이지만, 최근 supervised task with large datasets로부터 transfer 하는 연구도 제안됨
    - Natural Language Inference
    - Machine Translation
- CV에서는 transfer learning이 이미 많이 사용됨 (to fine-tune models pre-trained on ImageNet)

3. BERT

- 본 섹션에서는 아래와 같은 항목을 다룸
  - Model architecture
  - input representation
  - pre-training tasks
  - pre-training procedures
  - fine-tuning procedures
  - differences between BERT and OpenAI GPT

3. BERT

3.1 Model Architecture

- BERT는 multi-layer Bidirectional Transformer `encoder` 를 기반으로 함 (tensor2tensor에 배포된 코드 참고)
- Transformer 자체는 요즘 어디에서나 쓰임 (the use of Transformer has become ubiquitous)
- Transformers의 상세한 구조는 본 논문에서 스킵함(다음 링크 참고: http://nlp.seas.harvard.edu/2018/04/03/attention.html)
- notations
  - L : the number of layers (Transformer blocks)
  - H : the hidden size
  - A : the number of self-attention heads
- Hyper params in all cases
  - the feed-forward/filter size = $4H$, i.e,. 4x768 = 3072, 4x1024 = 4096
  - **BERT$_{\text{BASE}}$**: L=12, H=768, A=12, ToTal Params=110M
  - **BERT$_{\text{LARGE}}$**: L=24, H=1024, A=16, ToTal Params=340M

3. BERT

3.1 Model Architecture

- $\text{BERT}_{\text{BASE}}$는 OpenAI GPT랑 같은 모델 사이즈를 갖게함
- BERT Transformer는 Bidirectional self-attention을 쓰고, GPT는 constrained self-attention (left의 context만 볼 수 있음)을 씀
- Transformer Encoder: the Bidirectional Transformer
- Transformer Decoder: the left-context-only version Transformer

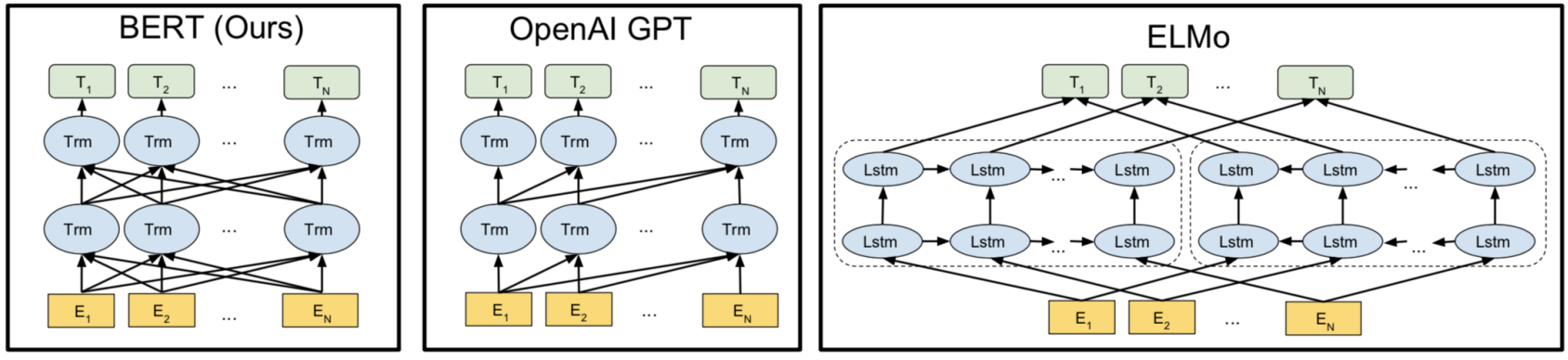


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

3. BERT

3.2 Input Representation

- BERT의 Input Reresentation은 sigle text sentence와 pair of text sentences([Qeustion, Answer])을 모두 하나의 토큰 시퀀스(one token sequence)에 담아서 처리함
  - WordPiece embeddings 사용함 (with a 30,000 token vocabulary) (Wu et al., 2016)
    - split word peices 는 ## 으로 처리함
  - learned positional embeddings 사용함 (sequence lengths up to 512 tokens)
  - 모든 sequence의 첫 토큰은 항상 the speical classification embedding([CLS])로 함. 이 토큰에 대응되는 output vector를 classification할 때 사용함
  - Sentence pairs는 single sequence에 들어가게 됨.
    - Speical token ([SEP])로 sentences를 분리함
    - 첫번째 sentence에는 every tokens마다 learned sentence A embedding을 더함
    - 두번째 sentence에는 every tokens마다 learned sentence B embedding을 더함
- Single-sentence inputs에 대해서는 sentence A embeddings만 사용함

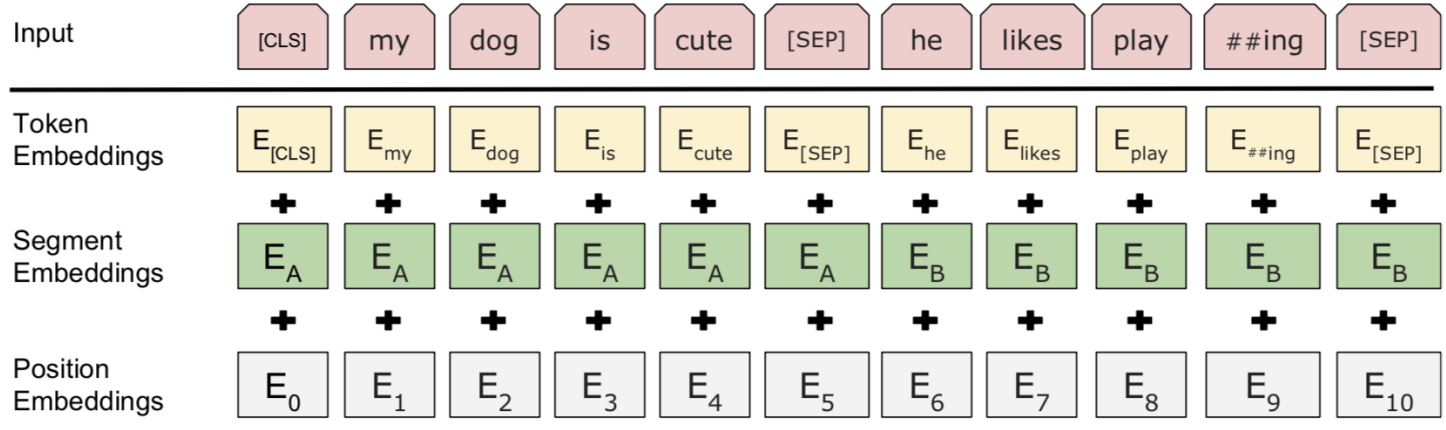


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

3. BERT

3.3 Pre-training Tasks

- traditional left-to-right or right-to-left language model 방식으로 BERT를 pre-train 하지 않음
- 대신에, two novel unsupervised prediction task로 BERT를 pre-train함

3. BERT

3.3.1 Task #1: Masked LM

- `Deep Bidirectional model`이 `left-to-right model 같은 단방향 모델보다, left-to-right와 right-to-left를 얕게 concat한 모델보다` 더 `powerful`하다는건 reasonable 함
- 하지만, Standard conditional language models은 left-to-right or right-to-left 처럼 한 방향만 학습이 가능함
  - 왜냐하면 Bidirectional condition은 결국 multi-layered context 안에서 각각의 단어가 자기 자신을 간접적으로 보기는게 가능하기 때문임 (`since bidirectional conditioning would allow each word to indirectly “see itself” in a multi-layered context`)
- Deep bidirectional representation을 학습시키기 위해, input tokens을 랜덤하게 특정 비율로 마스킹하고, 마스킹된 토큰을 맞추는 방법을 사용하고자 함(`a straightforward approach of mask- ing some percentage of the input tokens at random, and then predicting only those masked tokens`)
- 이러한 procedure를 `"masked LM" (MLM)`이라 칭하겠음
- 이는 마치 *Cloze* task (빈칸 채우기)와 비슷함
- mask token에 대응되는 hidden vector를 output softmax로 보내서 계산하는건 Standard LM과 같음
- 각 sequence안에 있는 `All WordPiece tokens`에 `15%`를 마스킹함
- denoising auto-encoder (Vincent et al., 2008)처럼 entire input을 reconstruct하기보다, masked word 만 예측함
- mismatch between pre-training and finetuning
  - [MASK] token은 fine-tuning시에 안나옴
  - 이 문제를 해결하기 위해 마스킹 해야하는 단어를 항상 [MASK]토큰으로 바꾸진 않음

3. BERT

3.3.1 Task #1: Masked LM

- 다음과 같은 방식으로 적용 (`이게 LM 성능 향상에 효과가 좋음!!`)
  - 80%: Replace the word with the [MASK] token, e.g., `my dog is hairy -> my do is [MASK]`
  - 10%: Replace thw rod with a random word, e.g., `my dog is hairy -> my dog is apple` (Noise 주고 정답 맞추게!)
  - 10%: Keep the word unchange, e.g., `my dog is hairy -> my dog is hairy`

```
if random.random() < 0.8:
  copy_training_ids[mask_LM_position_index_elm] = tokenizer.piece_to_id('[MASK]')
  masked_training_ids_batch.append(copy_training_ids)
else:
    # 10% of time, keep original
    if random.random() < 0.5:
        masked_training_ids_batch.append(copy_training_ids)
    # 10% of time, replace with random word
    else:
        copy_training_ids[mask_LM_position_index_elm] = list(vocab_word2idx.values())[random.ra
        masked_training_ids_batch.append(copy_training_ids)
```

- Transformer의 encoder는 어떤 단어를 예측하게 될지, 어떤 단어가 랜덤하게 대체될지는 모름
  - 그렇기 때문에 distributional contextual representation of every input token을 유지할 수 있음 (모든 토큰의 컨텍스트에 대한 분산 표현)
- MLM의 단점은, 15%정도만 예측에 사용하기 때문에, 모델이 학습을 통해 분포에 수렴하기 위한 pre-training step이 일반적인 LM보다 더 많이 필요함

3. BERT

3.3.2 Task #2: Next Sentence

Prediction

- QA나 NLI (Natural Language Inference)는 두 문장간의 관계를 이해하는것에 기초해 있음
- 이런 것들은 LM에서는 배울 수 없는 것임
- 이러한 문장간의 관계를 배우기 위해, `binarized next sentence prediction` task를 pre-train 시켜봄
- corpus에서 랜덤으로 50%는 임의의 문장, 나머지 50%는 다음문장으로 구성되도록 학습셋을 만듬

$$\text{Input} = \texttt{[CLS] the man went to [MASK] store [SEP]}$$
$$\texttt{he bought a gallon [MASK] milk [SEP]}$$
$$\text{Label} = \texttt{IsNext}$$

$$\text{Input} = \texttt{[CLS] the man [MASK] to the store [SEP]}$$
$$\texttt{penguin [MASK] are flight ##less birds [SEP]}$$
$$\text{Label} = \texttt{NotNext}$$

- pre-trained model은 97~98% Accuracy를 기록함

3. BERT

3.4 Pre-training Procedure

- Corpus
  - BooksCorpus (800M words) (Zhu et al., 2015) + English Wikipedia (2,500M words)
- Next Sentence Prediction task를 위해 문장 2개씩 샘플링함
  - 50%는 진짜 다음 문장, 나머지 50%는 랜덤한 문장
- 첫번째 문장엔 A embedding 더하고 두번째 문장엔 B embedding 더함
- combined length is $\leq$ 512 tokens
- LM masking은 WordPiece tokenization 적용 후 적용됨 (15%)
- speicial token에 대해서는 partial word pieces 없음 (`<s> 이런 애들은 word piece로 나누지 않았다는 뜻인듯?`)

3. BERT

3.4 Pre-training Procedure

- Hyper Params
  - Batch Size
    - 256 sentences (256 sequences * 512 tokens = 128,000 tokens/batch)
  - Steps
    - 1,000,000 steps == 40 epochs over the 3.3 billion word corpus
  - Adam
    - lr = 1e-4
    - β_1 = 0.9
    - β_2 = 0.999
    - L2 weight decay = 0.01
    - learning rate warmup : first 10,000 steps and linear decay of the learning rate
  - Dropout prob: 0.1 on all layers
  - Activation: `gelu(Gaussian Error Linear Units) activation` (OpenAI GPT 따라함, `이게 은근 성능 향상에 효과가 좋음!`)

```
def gelu(x):
  """Gaussian Error Linear Unit.
  This is a smoother version of the RELU.
  Original paper: https://arxiv.org/abs/1606.08415
  Args:
    x: float Tensor to perform activation.
  Returns:
    `x` with the GELU activation applied.
  """
  # ref: https://github.com/google-research/bert/blob/master/modeling.py#L264

  cdf = 0.5 * (1.0 + tf.tanh(
      (np.sqrt(2 / np.pi) * (x + 0.044715 * tf.pow(x, 3)))))
  return x * cdf
```

  - Training loss: `sum of (the mean masked LM likelihood) and (mean next setence prediction likelihood)`
  - GPUs
    - $\text{BERT}_{\text{BASE}}$: 4 Cloud TPUs in Pod configuration (16 TPU chips total)
    - $\text{BERT}_{\text{LARGE}}$: 16 Cloud TPUs (64 TPU chips total)
    - 4 days to complete

3. BERT

3.5 Fine-tuning Procedure

- sequence-level classification task: fixed-dimensional pooled representation을 얻기 위해 [CLS] 토큰에 대응되는 final hidden state를 사용함
  - [CLS] 토큰에 대응되는 output vector를 $C \in \mathbb{R}^H$라 표현하겠음
  - 약간의 파라미터만 추가되는데, classification layer $W \in \mathbb{R}^{K \times H}$ 만 추가된다고 보면됨. K는 classifier의 labels 개수임
  - Label prob: $P = softmax(CW^T)$
  - BERT의 모든 params + W가 함께 학습됨
- span-level, token-level prediction task: Section 4 참조
- 대부분의 hyper parms은 pre-training 때와 같지만(dropout은 항상 0.1로~!) 3가지가 다름
  - Batch size: 16, 32
  - Learning rate(Adam): 5e-5, 3e-5, 2e-5
  - Number of epcohs: 3, 4
- Dataset 크면 hyper params 선택에 영향 덜 받음
- Fine-tuning은 대게 되게 빨리됨, 그러므로 validation set에 대해서 exhaustive search를 해서 최적의 hyper parms을 갖는 모델을 고르는게 좋음

3. BERT

3.6 Comparison of BERT and OpenAI GPT

- 기존에 존재하는 pre-training 방법과 비교하면, OpenAI GPT가 가장 유사함
  - GPT는 left-to-right Transfomer LM 사용
- Dataset
  - GPT is trained on the BooksCorpus (800M words)
  - BERT is trained on the BooksCorpus (800M words) and Wikipedia (2,500M words)
- Sentence Separator
  - GPT uses a sentence separator ([SEP]) and classifier token ([CLS]) which are only introduced at `fine-tuning time`;
  - BERT learns [SEP], [CLS] and sentence A/B embed- dings during pre-training
- Batch Size
  - GPT was trained for 1M steps with a batch size of 32,000 words
  - BERT was trained for 1M steps with a batch size of 128,000 words
- Learning Rate
  - GPT used the same learning rate of 5e-5 for all fine-tuning experiments
  - BERT chooses a task-specific fine-tuning learning rate which performs the best on the development set

4 Experiments

BERT를 fine-tuning해서 11개의 NLP Task에 적용함

4.1 GLUE Datasets

- The General Langauge Understanding Evaluation (GLUE) benchmark (Wang et al., 2018)는 여러개의 NLU task의 모음임
  - MNLI: `Multi-Genre Natural Language Inference`는 주어진 문장들을 entailment, contradiction, or neutral로 분류
  - QQP: `Quora Question Pairs`는 문장이 semantically 일치하는지 여부를 이진분류
  - QNLI: `Question Natural Language Inference`는 SQuAD를 binary classification task로 답을 포함한 문장인지 아닌지를 분류
  - SST-2: `The Stanford Sentiment Treebank`는 single-sentence에 대한 감성분석 이진 분류
  - CoLA: `The Corpus of Linguistic Acceptability`는 single-sentence classification task로써 영어 문장이 언어적으로 acceptable한지 아닌지 분류
  - STS-B: `The Semantic Textual Similarity Benchmark`는 문장 페어가 얼마나 유사한지 1~5점으로 스코어링 해놓은 데이터셋임. 의미적으로 얼마나 유사한지 분류
  - MRPC: `Microsoft Research Paraphrase Corpus`도 문장페어가 얼마나 의미적으로 유사한지를 판단하는 데이터셋
  - RTE: `Recognizing Textual Entailment`MNLI의 binary 분류 버전
  - WNLI: `Winograd NLI`는 natural language inference dataset임

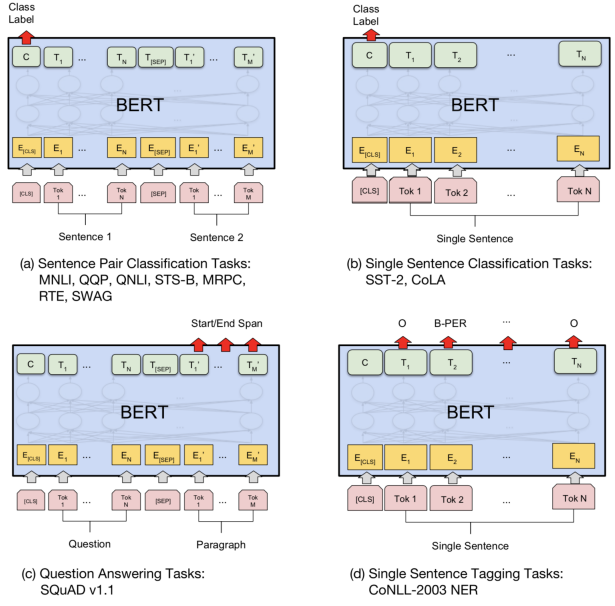


Figure 3: Our task specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input embedding, T_i represents the contextual representation of token i, [CLS] is the special symbol for classification output, and [SEP] is the special symbol to separate non-consecutive token sequences.

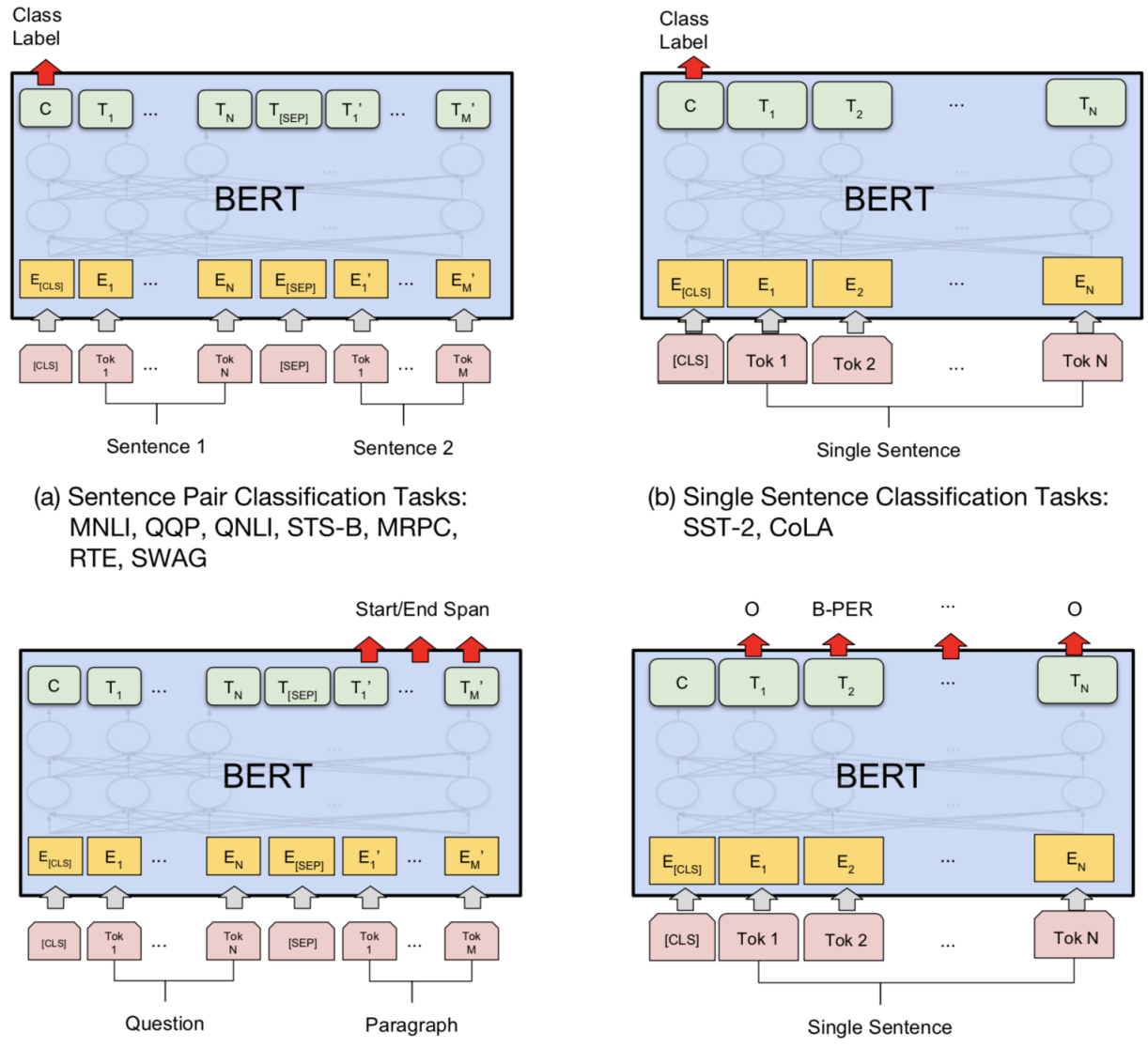


Figure 3: Our task specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input embedding, T_i represents the contextual representation of token i, [CLS] is the special symbol for classification output, and [SEP] is the special symbol to separate non-consecutive token sequences.

4.1.1 GLUE Results

- GLUE Leaderboard가 궁금한다면 클릭
- OpenAI, Language-unsupervised
- fine-tune option
  - batch size: 32
  - epoch: 3
  - lr: 5e-5, 4e-5, 3e-5, 2e-5
- $\text{BERT}_{\text{LARGE}}$의 경우 small dataset에서 finetune 할 경우 unstable한 현상이 있음
- BERT는 4.4~6.7% 정도의 견고한 차이로 SOTA보다 높은 결과를 얻음

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. 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. OpenAI GPT = (L=12, H=768, A=12); $\text{BERT}_{\text{BASE}}$ = (L=12, H=768, A=12); $\text{BERT}_{\text{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

4.2 SQuAD v1.1

- Wikipedia로부터 만든 100k의 question/answer pairs
- Input paragraph에서 Input Question에 대한 정답찾기

containing the answer, the task is to predict the answer text span in the paragraph. For example:

- Input Question:

  `Where do water droplets collide with ice crystals to form precipitation?`

- Input Paragraph:

  `... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. ...`

- Output Answer:

  `within a cloud`

4.2 SQuAD v1.1

- Classification Task와는 좀 달리, `Start Token` $S \in \mathbb{R}^H$, `End Token` $E \in \mathbb{R}^H$찾기임
- fine-tuning때는 새로 도입되는 param은 Start, End token뿐임
- input token i에 대한 Final hidden vector를 $T_i \in \mathbb{R}^H$라 할 때, word i가 Start token이 될 확률은 T_i와 S의 dot-product의 softmax로 계산함
- $P_i = \frac{e^{S \cdot T_i}}{\Sigma_j e^{S \cdot T_j}}$
- End token도 똑같이 적용함
- Start, End token의 position 일치 여부에 대한 log-likelihood를 objective로 두고 학습시킴
- Hyper params: 3 epoch, 5e-5 lr, 32 batch size
- ~~START, END token에 대한 vector를 새로 생성해서 따로 갖고 있다가 기존 seq token과 Attention 만 비교하는건가..이러면 self-attention은 아니고 기존에 알고 있던 attention vector를 쓰는 해석인데.. 체크해봐야겠음~~
- SQuAD는 testing procedure가 매우 까다로워서 submitter가 직접 SQuAD organizers에게 컨택해서 hidden test set에 대해 테스트해야 함. 본 논문에서는 자체 평가기준으로 best system만 제출함
- 결과는 **TOP**을 갱신함. Public data인 TriviaQA도 학습시켰더니 점수 더 높아짐

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

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

4.3 Named Entity Recognition

- CoNLL 2003 NER dataset (200k training words)에 대해서 fine-tune함
- Person, Organization, Location, Miscellaneous, Other 로 나뉨
- Final hidden representation $T_i \in \mathbb{R}^H$를 NER label set에 대한 classification layer에 feed 시킴
- CRF 없음, non-autoregressive임 (~~어런 상황에서 NER은 사실 굉장히 어려움~~)
- BERT는 WordPeiece tokenizer쓰기 때문에 특성상 NER 라벨과 align이 틀어질 수 있음
- 이를 해결하기 위해 CoNLL-tokenized input word를 WordPiece tokenizer로 다시 tokenization함

```
Jim   Hen   ##son was a puppet ##eer
I-PER I-PER X     O   O O      X
```

Where no prediction is made for X. Since the WordPiece tokenization boundaries are a known part of the input, this is done for both training and test. A visual representation is also

- X 라고 표시된 곳은 prediction하지 않음 (~~loss를 계산하지 않는건가?~~)
- 결과는 ELMo+BiLSTM+CRF 같이 기존의 LSTM 계열의 NER 끝판왕 모델을 이기고 SOTA 찍음

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Table 3: CoNLL-2003 Named Entity Recognition results. The hyperparameters were selected using the Dev set, and the reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

4.4 SWAG

- 생략

5. Ablation Studies

- BERT의 요소를 한개씩 분해해서 실험해봄으로써 관계적인 중요성을 이해하고자함
- parameter나 pre-training 데이터는 같은거 씀

5. Ablation Studies

5.1 Effect of Pre-training Tasks

- **No NSP**: "masked LM" (MLM)은 적용했지만, "next sentence prediction" (NSP)는 적용 안함
- **LTR & No NSP**: Left-to-Right (LTR) LM 적용, every input word를 예측하고 masking은 따로 안함. left-only constraint 적용함, fine-tuning때 bidirectional context를 적용하면 성능이 항상 떨어졌기 때문 (~~left-only constraint가 뭘까? look-ahead masking 같은건가~~). NSP도 적용 안함. **GPT와 directly하게 비슷함!**
- LTR과 RTL 모델을 각각 학습시키고, ELMo처럼 concat하면 성능 오르지만
  - 비용이 2배가 되고
  - QA 같은 태스크에서 직관적이지 않고 (RTL이 적합하지 않음)
  - Deep bidirectional model에 비해서 less powerful함 (deep 에서는 left or right context 쓰는 것에 대해서 선택이 가능하기 때문)

	Dev Set				
Tasks	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	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 $\text{BERT}_{\text{BASE}}$ 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.

5.2 Effect of Model Size

- Deep하고 params 많은게 좋음

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.

5.3 Effect of Number of Training Steps

- 정말 100만 스텝이나 필요한가요? Yes!

1. Question: Does BERT really need such a large amount of pre-training (128,000 words/batch * 1,000,000 steps) to achieve high fine-tuning accuracy?
   Answer: Yes, $\text{BERT}_{\text{BASE}}$ achieves almost 1.0% additional accuracy on MNLI when trained on 1M steps compared to 500k steps.

2. Question: Does MLM pre-training converge slower than LTR pre-training, since only 15% of words are predicted in each batch rather than every word?
   Answer: The MLM model does converge slightly slower than the LTR model. However, in terms of absolute accuracy the MLM model begins to outperform the LTR model almost immediately.

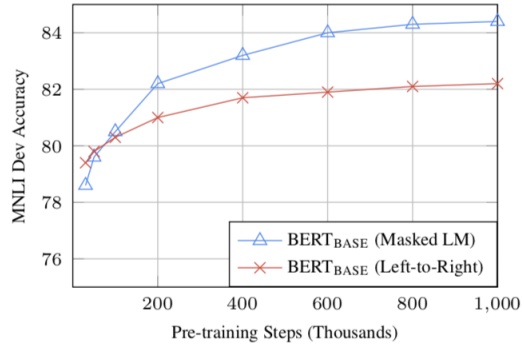


Figure 4: Ablation over number of training steps. This shows the MNLI accuracy after fine-tuning, starting from model parameters that have been pre-trained for k steps. The x-axis is the value of k.

5.4 Feature-based Approach with BERT

- 모든 NLP task를 Transformer Encoder architecture로 표현하는게 쉬운건 아님
- pre-compute 할 수 있으면 비용이 절약될 것임
- ELMo가 contextual representation을 만든것 처럼 BERT도 얼마나 잘 만들었는지 CoNLL-2003 NER Task에 대해서 평가함
- BERT의 어떤 param도 fine-tuning하지 않고 two-layer 768-dim BiLSTM을 추가해서 테스트함
- BERT 전체를 fine-tune한게 96.4인데 BERT는 건드리지 않고 마지막 Layer 4개를 concat한게 96.1이 나옴
- BERT는 fine-tuning해서도 잘 쓰이지만, feature-based approach에서도 효과적임이 입증됨

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.

6. Conclusion

- Transfer learning with language models이 뜨고 있음, 많이 개선됨
- Unsupervised pre-training이 NLU system에 합쳐질 수 있음
- `Contribution` 은 deep *bidirectional* architecture를 갖는 pre-trained model이 generalization 될 수 있음을 보인 것임
- 실험결과는 매우 뛰어남, 어떤 케이스는 사람보다 잘 함. 앞으론 BERT가 잡아내거나 그러지 못한 liguistic phenomena에 대해서 연구할 것임