BERT

Part	NLP
진행 상태	Done

Pre-training of Deep Bidirectinal Transformers for Language Understanding

---

0. Abstract

BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

BERT는 Transformer의 Encoder 구조를 이용하여 unlabeled data로부터 bidirectional representation을 pre-train한 후, fine-tuning하는 모델이다. Pre-trained BERT 모델에 하나의 output layer를 추가함으로써 다양한 task에 적용될 수 있다.

---

1. Introduction

Language model의 pre-training 방법은 많은 NLP task에서 좋은 성능을 내고 있었다. Down-stream task에 pre-trained language representation을 적용하는 방법에는 두 가지 전략이 존재한다.

1. **feature-based approach**

   pre-trained representation을 추가적인 feature로 활용한 task-specific architectures를 사용한다.

   Ex. ELMO

2. **fine-tuning approach**

   모든 pre-trained 파라미터를 fine-tuning하여 down stream task를 학습한다.

   Ex. GPT

두 가지 방법 모두 pre-train 과정에서 동일한 목적함수를 공유하는데, **일반적으로 language representation을 학습하기 위해 unidirectional language model을 사용**한다.

* ELMO는 각각의 단방향(순방향, 역방향) 모델의 출력값을 concat해서 사용하는 것이기 때문에 하나의 모델 자체는 단방향이다.

본 논문에서는 기존의 방법이 pre-trained representation의 성능을 저하시킨다고 주장한다. **BERT는 “masked language model” (MLM) 을 pre-trainig의 목적으로 사용하여 단방향성의 제약을 완화시킨다.**

Masked Language Model(MLM)은 **랜덤하게 input token의 일부를 마스킹**하고, 해당 토큰이 구성하는 문장만을 기반으로 그 **마스킹된 token들의 원래 값을 정확하게 예측하는 것이 목적**이다. MLM는 양방향 context를 융합시켜 bidirectional Transformer를 가능하게 한다.

추가적으로, 두 문장이 주어졌을 때 이어지는 문장인지 예측하는 **next sentence prediction** task를 사용한다.

---

2. Related Work

▼ **Unsupervised Feature-based Approaches**

word embedding의 Pre-training은 오늘날 NLP 분야에서 중요한 부분이며, 이러한 word embedding을 통한 접근 방식은 자연스레 sentence embedding 혹은 paragraph embedding으로 이어졌다.

BERT 이전의 연구에서는 아래와 같은 방법을 사용하였다.

1. 다음 문장 후보들의 순위를 메기는 방법
2. 이전 문장이 주어졌을 때, 다음 문장의 단어를 왼쪽에서 오른쪽으로 생성하는 방법
3. 노이즈 제거 AutoEncoder에서 파생된 방법

   (노이즈가 추가된 문장은 입력을 받아 원본 문장을 복원하려고 시도한다.)

ELMo와 그 후속 모델들은, 전통적인 word embedding 연구에서 left-to-right와 right-to-left 언어 모델을 통해 context-sensitive feature들을 뽑아내는 방식으로 발전하여 높은 성능 향상을 이끌어내었으나, **deep bidirectional하지 않다.**

▼ **Unsupervised Fine-tuning Approaches**

최근에는, contextual token representation을 만들어내는 인코더가 pre-training되고, supervised downstream task에 맞춰 fine-tuning된다.

이러한 접근방식은 처음부터 학습하는데 적은 파라미터로 충분하다는 장점이 있고, OpenAI GPT는 GLUE 벤치마크의 여러 문장 수준 작업에서 최첨단 결과를 달성하였다.

▼ **Transfer Learning from Supervised Data**

큰 dataset을 가진 지도학습에서 효과적인 transfer learning이 가능하다.

---

3. BERT

Pre-training: 다양한 pre-training tasks에서 unlabled data를 활용해 학습된다.

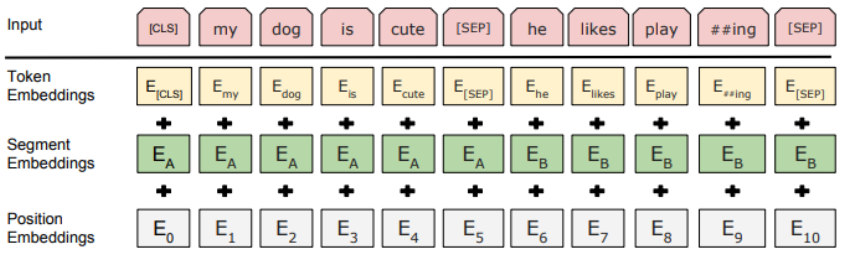
fine-tuning: 사전학습 된 파라미터로 초기화되고, downstream tasks의 labled data를 이용해 fine-tuning한다. (각각의 downstream task는 동일한 사전학습 파라미터로 초기화된다.)

- **Model Architecture**

  BERT 모델의 구조는 **양방향 Transformer의 Encoder를 여러 층 쌓은 것**이다.

- **Input/Output Representations**

  3가지 Embedding vector의 합으로 Input이 표현된다.



  - **Token Embeddings**

    ▼ **Wordpiece Embedding**

    단어의 출현 빈도수를 기반으로 하여, 자주 등장하지 않는 단어는 더 작은 하위 단어로 분해한다.

    Ex)

    1. playing → play + ing
    2. playtime → play + time

3. displayed → dis + play + ed

모든 문자들을 분해하고, 가장 자주 등장하는 문자 쌍을 묶어 새로운 하위 단어를 만드는 과정을 반복하여 출현 빈도가 높은 문자 쌍을 기반으로 하위 단어를 생성한다.

→ 희소한 단어, OOV 단어 등의 문제를 해결하고 전체적으로 더 나은 성능을 발휘한다.

- **Segment Embeddings**

  input sequence는 문장 쌍으로 구성된다. 문장 쌍의 각 문장들은 [SEP] 토큰으로 분리되며, 각 문장이 A문장인지 B문장인지 구분하기 위한 Segment Embeddings을 진행한다.

- **Position Embeddings**

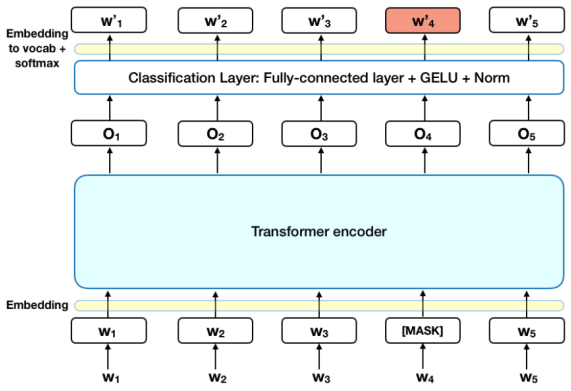
Pre-training BERT

BERT의 **Pre-training 과정에서는 크게 두가지 unsupervised-tasks** (Masked LM, Next Sentence Prediction(NSP))가 사용된다.

- **Task1: Masked LM**

  input tokens의 일정 비율을 마스킹하고, 마스킹 된 토큰을 예측하는 과정이다.

  본 논문에서의 실험 결과, 15%의 비율로 마스킹을 할 때 가장 성능이 좋았다고 한다.



[MASK] token은 pre-training에만 사용되고, fine-tuning 과정에서는 사용되지 않기 때문에 pre-training과 fine-tuning 사이의 불일치가 발생할 수 있다.

이를 해결하기 위해 15%의 [MASK] token에서 추가적인 처리를 해준다.

  - 80%: **[MASK] token**으로 바꾼다.

    Ex. My dog is hairy → My dog is [MASK]

  - 10%: **random token**으로 치환한다.

    Ex. My dog is hairy → My dog is apple

  - 10%: **원래의 token** 그대로를 사용한다.

    Ex. My dog is hairy → My dog is hairy

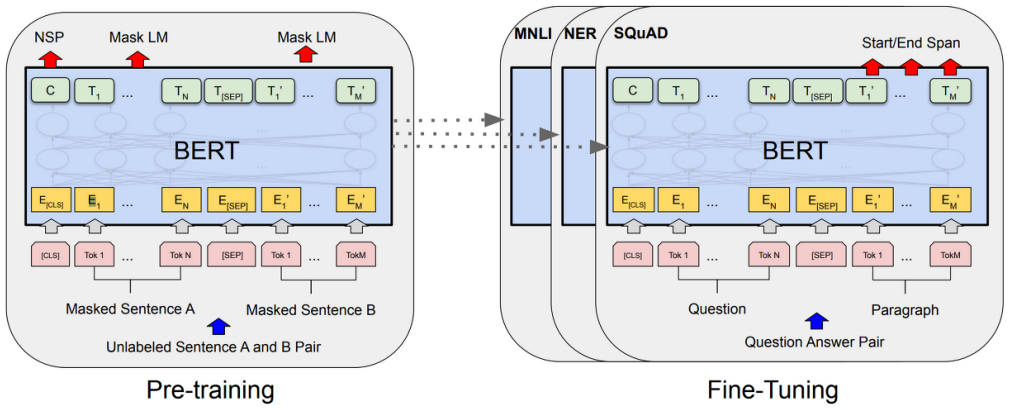
- **Task2: Next Sentence Prediction (NSP)**

  많은 NLP의 downstream task(QA, NLI 등)는 **두 문장 사이의 관계**를 이해하는것이 중요하다.

  이를 학습하기 위해 모든 단일 언어 말뭉치에서 쉽게 생성될 수 있는 **binarized next sentence prediction** task에 대해 pre-train한다. 즉 두 문장이 주어졌을 때, **두 번째 문장이 첫 번째 문장의 바로 다음에 오는 문장인지 여부를 예측**하도록 한다.

  - 50%는 실제로 이어진 것, 50%는 실제로 이어지지 않는 것에서 뽑아와 학습시킨다.

Fine-tuning BERT



가장 위에 하나의 layer를 추가로 쌓아 fine-tuning을 진행한다.

수행하고자 하는 downstream task에 따라 **task-specific input**을 받는다.

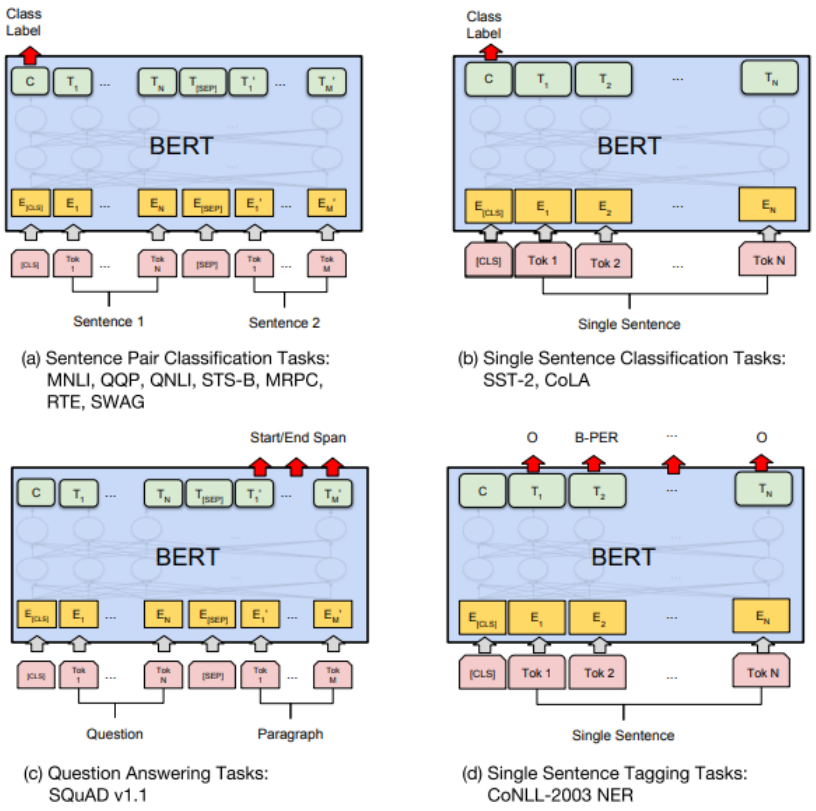
(1) 문장 쌍 (paraphrasing)

(2) 가설-전제 쌍 (entailment)

(3) 질문-문단 쌍 (question answering)

(4) 텍스트-None 쌍 (text classification or sequence tagging)

Output도 downstream task에 따라 달라진다.



token-level task) 토큰 표현이 출력 계층으로 전달된다. Ex. Question Answering, Seqeunce tagging

classification) 모든 sequence의 첫번째 token인 [CLS]와 대응되는 최종 hidden state가 분류 문제를 해결하기 위한 표현들을 종합하고, 이것이 분류를 위한 출력 계층으로 전달된다. Ex. Entailment, sentiment analysis

---

4. Experiments

GLUE

General Language Understanding Evaluation

: 다양한 자연어 이해 task가 담겨있는 benchmark

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**

SQuAD

The Stanford Question Answering Dataset

: Wikipedia를 바탕으로 만든 Question-Answering Task benchmark

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**

SWAG

Situations With Adversarial Generation

: 앞 문장이 주어지고, 보기로 주어지는 4가지 보기 중에 가장 적절한 문장을 고르는 Task

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

→ 모든 Task에서 SOTA 성능을 달성하였다.

---

5. Ablation Studies

BERT에서 어떤 요소가 성능 향상에 어떻게 기여했는지 구체적으로 알아보기 위해 동일한 환경에서 특정 조건만을 변경해 어느 정도의 영향을 끼치는지 분석했다.

Effect of Pre-training Tasks

No NSP: pre-training 단계에서 MLM만 수행하고, NSP는 수행하지 않은 모델

LTR & No NSP: MLM 대신 Left to Right의 Unidirectional attention을 적용하고 NSP는 수행하지 않은 모델

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.

BERT - No NSP → **NSP Pre-training이 성능 향상에 영향을 끼친다.**

NO NSP - LTR&No NSP → **Bidirectional 모델이 성능 향상에 영향을 끼친다.**

Effect of Model Size

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.

→ **Model Size가 증가할수록 성능이 높아지는 경향이 있다.**

Feature-based Approach with BERT

지금까지의 BERT는 모두 fine-tuning 모델이었는데, BERT를 Feature-based로 사용했을 때 성능을 분석하였다.

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		
$BERT_{LARGE}$	96.6	92.8
$BERT_{BASE}$	96.4	92.4
Feature-based approach ($BERT_{BASE}$)		
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.

개체명 인식 task에 대해 Feature-based Approach를 적용해보았다.

Fine-tuning 단계를 제거하고, Transformer Layer의 output을 그대로 768-dimensional BiLSTM의 input으로 사용한다. 그 뒤 classification layer를 통과시켜 결과를 도출해낸 결과 기존 모델들을 뛰어넘었다.

---

6. Conclusion

Deep Bidirectional Model을 통해 다양한 NLP 작업을 성공적으로 수행할 수 있게 한다.