논문리뷰

BERT : Pre-training of Deep Bi-directional Transformers for Language Understanding

Presented by | 이 혜 리
2020.01.14

Abstract

- Transfer learning

기존의 만들어진 모델을 사용하여 새로운 모델을 만들시 학습을 빠르게 하며, 예측을 더 높이는 방법

이미 잘 훈련된 모델이 있고, 특히 해당 모델과 유사한 문제를 해결시 transfer learining을 사용

- Bi-directional

ELMO, OpenAI GPT

대용량 unlabeled corpus를 통해 LM을 학습하고, 이를 토대로 뒤쪽에 특정 task를 처리하는 network를 붙이는 방식

Shallow bi-directional or uni-directional

- Fine-tuning

기존에 학습된 모델을 기반으로 아키텍처를 새로운 목적에 맞게 변형하고 이미 학습된 모델 weights로부터 학습을 업데이트 하는 법

비지도 학습으로 학습시킨다.

이후 다운스트림 태스크에서는 라벨된 데이터로 parameter를 조정한다.

Introduction

- Bert와 비슷한 접근방식을 가지고 있는 기존모델에 대한 개략적 소개
- 이런 pre-trained language representation을 적용하는 방식은 크게 **feature-based**와 **fine-tuning**방식이었다.
  1. **feature-based**
     - 이미 학습된 단어표현을 특정 task를 위한 단어표현에 덧붙이는 방식)
     - 즉 2개의 network를 붙여서 사용한다.
     - ELMO가 대표적
  2. **fine-tuning**
     - task-specific(특정 task에만 꼭 들어맞는) hyperparameter를 최대한 줄이고, pre-trained된 parameter들을 downstream task학습을 통해 조금만 바꿔주는 방식
     - OpenAI GPT가 대표적
  3. **기존방법론 한계점**
     - 일반적인 LM을 사용했다. 일반적인 LM이란 앞의 n개의 단어를 가지고 뒤의 단어를 예측하는 모델을 세우는 것이다.
     - 하지만 이는 필연적으로 uni-directional해서 pre-training할 때 아키텍처 선택지가 제한적이다. (반드시 꼭 맞는 아키텍처를 사용해야 함).
     - 이러한 단점을 없애기 위해서 ELMO에서는 Bi-LSTM으로 양방향성을 가지려 노력하지만 굉장히 shallow

하지만 BERT는 새로운 방식으로 pre-trained Lauguage representation을 학습했고 이것은 특정 task에만 꼭 들어맞는 아키텍쳐의 필요성을 없애버렸다.
또한 이 새로운 방식은 진정한 bi-directional을 이뤄낼 수 있게 했다.

Related Work

- pre-training language representaions에 대해 가장 많이 사용되는 방법 3가지
    1. 2.1 Unsupervised Feature-based Approaches
    2. 2.2 Unsupervised Fine-tuning Approaches
    3. 2.3 Transfer learning from supervised data

BERT

• Model Architecture

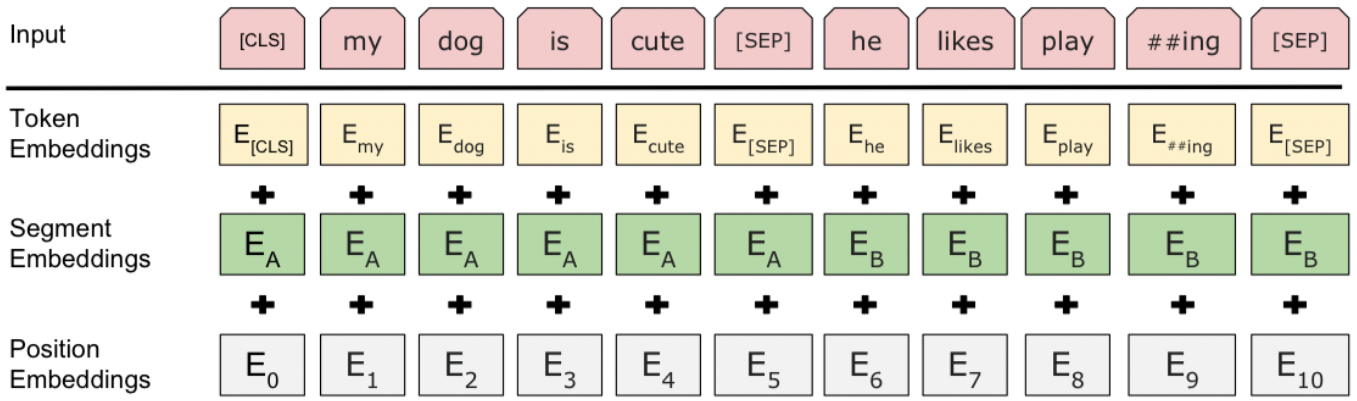
- Transformer 중에서도 encoder부분만 사용한다.
- 모델의 크기에 따라 base 모델과 large 모델을 제공한다.
  - BERT base : L=12, H=768, A=12, Total Params = 110M
  - BERT large : L=24, H=1024, A=16, Total Params = 340M
- BERT base :
  - openAI GPT의 하이퍼파라미터가 같음. 근데 *양방향 액션*을 취해줘서 성능이 좋게 나옴.
  - 모델의 하이퍼파라미터가 동일하더라도 *pre-training concept*을 바꿔주는 것만으로 훨씬 좋은 성능을 낼 수 있다는 걸 보여주려고 base를 구축한 것 같다.
- BERT large : 진짜 성능을 위해 구축, 대부분 NLP task SOTA는 이 모델로 이뤄냄

(L : transformer block layer 수, H : hidden size, A : self-attention heads(feed-forward / filter size = 4H))

• Input/Output Representations

- BERT는 Transformer(Positional Encoding)와 달리 Positional Embedding을 사용한다. 여기에 Segment Embedding을 추가해 3개의 임베딩을 합산한 결과를 취한다.
- 여기서 sentence는 **"span of continuous of text(문단 이상의 큰 토큰)"을** 의미한다.
- 모든 sentence의 첫번째 token은 언제나 [CLS]. 이 [CLS]token은 transformer 전체층을 거치고 나면 token sequence의 결합된 의미를 갖는데, 여기에 **간단한 classifier를 붙이면 단일 문장, 또는 연속된 문장의 classification을 쉽게 할 수 있다**. 만약 classification task가 아니라면 이 token은 무시한다.

➢ token embedding : Tokenization using **WordPiece Embedding(BPE기반)**
➢ segment embedding : 모든 토큰들이 문장 A or B 중 어디에 속해있냐 나타낸다.
➢ positional embedding : 토큰의 sequential한 의미를 보유하기 위해 최대 512 토큰 표현

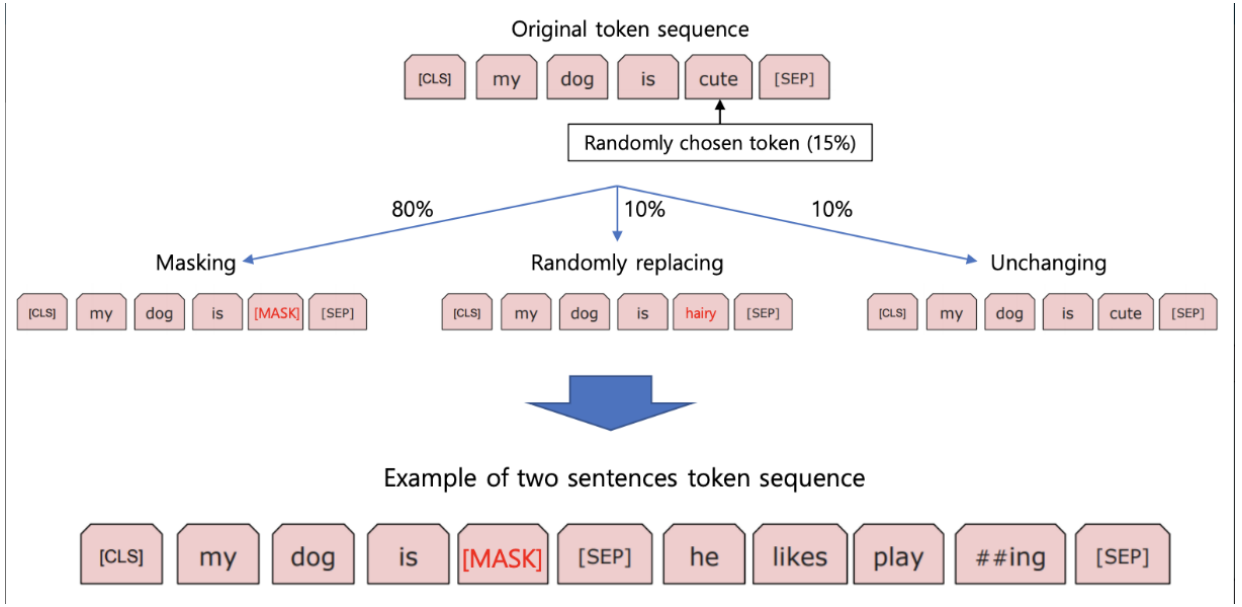


3.1 Pre-training BERT

- BERT 학습방식의 가장 큰 특징은 Bi-directional 하다는 것. MLM, NSP 2가지 방식이 가능케 함.
- 이 MLM, NSP는 또 Input Embeddings에 추가한 식별자 ([CLS], [SEP])가 가능케 함.

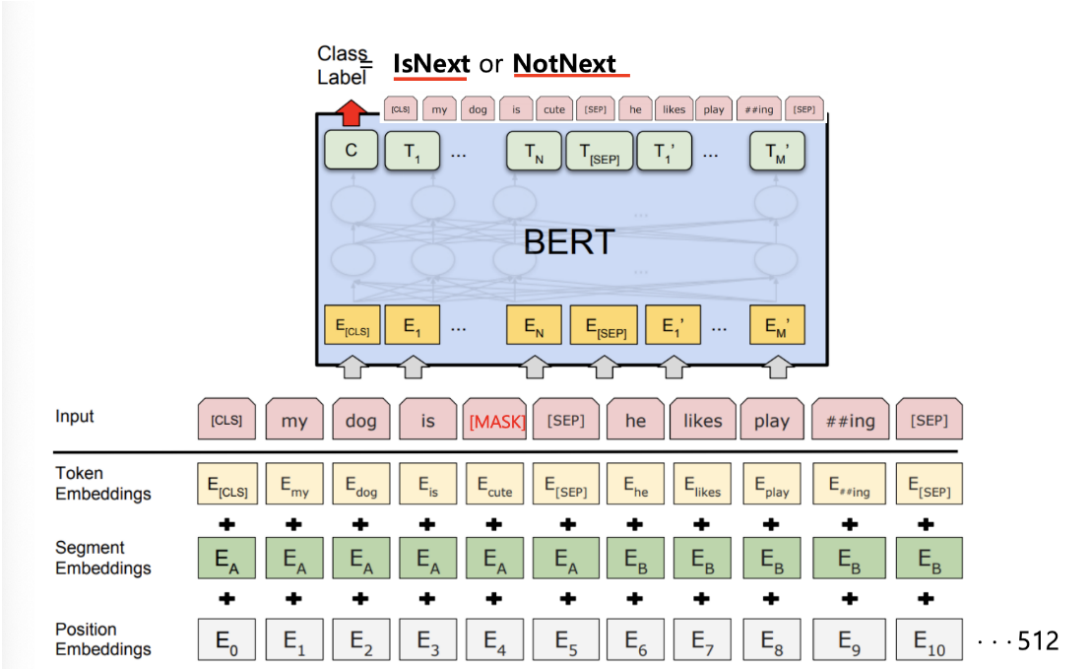
• Task #1 : Masked LM

- 트랜스포머 구조에 넣어서 주변 단어의 context만 보고 masked token을 예측하는 모델
- LM이 left-to-right(또는 r2l)을 통해 문장 전체를 predict하는 방법론과 달리, [MASK] token만을 predict하는 pre-training task를 수행
- random words로 바뀌는 건 1.5%(15% 중 10%)밖에 안되기 때문에 language understanding능력엔 해를 끼치지 않음
- **방법** : input에서 무작위로 전체 문장의 15%를 masking함. 단 특정 단어를 선택해서 마스크를 씌우는 게 아니라 다음 아래 방식으로 마스킹 함.
  - 80%는 마스크 씌우고
  - 10%은 randomly하게 다른 단어로 바꿈
  - 또 10%는 원래 그냥 맞는 문장으로 만든다.



• Task #2 : Next Sentence Prediction(NSP)

- 랜덤 추출한 두 문장을 pre-train에 같이 넣고, 두 문장이 이어지는지 50:50 비율로 binary 분류.
- **이유** : BERT는 Transfer learning으로 사용되고 파인 튜닝 할 때 *QA*, *Natural Language inference*같은 태스크에서는 전체 문장이 흘러가는 것(또는 두 문장의 관계를 이해하는 것)이 중요하며 이런 이해는 MLM으로 학습하는 것만으로는 충분하지 않기 때문에 굳이 수행함
- [CLS] 벡터의 Binary Classification 결과를 맞추도록 학습한다.



• 3.2 Fine-tuning BERT

- pre-training을 거치고 난 후 다운스크림 방식으로 파인튜닝 시행함.
- 몇 가지를 제외하고 pre-training hyperparameter와 동일.
- 다른 점은 *batch size, learning rate, number of epochs*
- 최적의 파라미터는 task마다 달라지지만, 다음 아래를 사용하면 대부분 잘 학습된다.

Batch size : 16,32 **Learning rate(Adam)** : 5e-5, 3e-5, 2e-5 **Epochs** : 3~4

- 데이터셋이 클수록 하이퍼파라미터 영향을 덜 받고 잘 수행됨을 알 수 있었다.

4. Experimets

• 4.1 GLUE

- General Language Understanding Evaluation [출처]

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
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

- 모든 Dev Set에서 BERT large가 SOTA 달성
- 데이터셋 크기가 작더라도 BERT large는 고르게 성능이 잘 나온다.
- OpenAI GPT와 BERT large는 attention masking을 제외하고 model architecture가 유사하지만 후자가 더 좋은 성능을 보임
- BERT base보다 **BERT large**가 **더 좋은 성능**을 보임

—> 데이터셋 크기가 작아도 모델이 큰 게 더 좋은 결과를 낸다. 이것이 BERT의 flex.

• 4.2 ~ 4.4 SQuAD, SWAG

- SQuAD : 질문(Question)이 주어지고 Answer중에서 서브스트링을 맞추는 태스크. 즉, 답이 어디서부터 어디까지인지 찾는 것.
- SWAG : 앞 문장이 주어지고, 보기로 주어진 4개의 문장 중에서 가장 잘 이어지는 문장을 찾는 태스크.
- SQaAD v1.1, SWAG의 경우 사람보다 좋은 결과를 보임
- SQuAD 방법 및 결과
  - 방법
    1. 질문을 A embedding , 지문을 B embedding으로 처리하고, **지문에서 정답이 되는 substring의 처음과 끝을 찾는 task로 문제를 치환**한다.
    2. start vector와 end vector를 fine-tuning중에 학습하여, 지문의 각 token들과 dot product하여 substring을 찾아낸다.

  - 결과
    - **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

5. Ablation Studies

• 5.1 Effect of pre-training Tasks

- Masked LM, NSP 2개를 이용을 했는데 각각의 태스크가 어떤 영향을 미쳤는가에 대해 논증을 확실히 하기 위해서 하나씩 없애면서 실험한 것.
  - **No NSP** : MLM Ok, 다음 문장 예측(NSP)없앤 모델
  - **LTR & No NSP** : MLM대신 left-to-right사용하고, NSP도 없앤 모델

	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

- 모든 Dev Set에서 Masked LM & NSP를 모두 적용한 BERT base가 가장 성능이 좋다
- **No NSP**의 경우 **NLI**계열의 task(MNLi-m)에서 **성능이 많이 하락**하는데, 이는 NSP task가 문장 간의 논리적인 구조 파악에 중요한 역할을 하고 있음을 의미한다.
- MLM대신 LTR을 쓰면 성능 하락이 더 심해진다. BiLSTM을 붙인 모델은 성능이 제일 나빴다. —> BiLSTM은 별 능력이 없다. **—> MLM task가 더 Deep Bidirectional하다.**

• 5.2 Effect of Model Size

- 모델 사이즈는 클수록 좋다.
- 특정 태스크를 파인튜닝할 때 적은 데이터셋을 이용해도 잘 된다~!

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

BERT_base : L=12, H=768, A=12,
BERT_large : L=24, H=1024, A=16,

지금까지 *"Bert는 다운스트림 태스크를 할 때 간단한 classifier를 부착해서 모든 layer를 다시 학습하는 fine-tuning방법을 사용했다."*
그렇다면 ELMO처럼 feature-based approach로 사용했을 때도 이점이 존재한다는 걸 보여주겠다.

• 5.3 Feature-based Approach with BERT

- bert는 fine-tuning approach만을 이용해서 학습했지만, feature-based approach은 몇가지 이점이 있음.
  1. Transformer architecture encoder는 사실상 모든 NLP Task를 represent하지 못하므로, 특정 NLP task를 이행할 수 있는 model architecture가 필요
  2. computational benefits을 얻을 수 있음
- 옆 표에서 보면, **Concat Last Four Hidden**값을 사용하면, **Finetune All**과 단지 0.3 F1 score차이밖에 나지 않습니다.
- 이를 통해, BERT는 **Feature-based Approach**에서도 효과적이라고 할 수 있다.

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.

Conclusion

1. **SOTA** : transfer learning(Masked LM, NSP)을 통해서 굉장한 성능의 향상을 보임
2. **Bidirectional model** : MLM같은 방식을 사용했으니 진정한 의미의 양방향 모델이라고 할 수 있다.
3. **BERT large is better than BERT base** : 적은 데이터셋에도 불구하고 오버피팅 걱정없이 큰 모델을 사용하는 것이 이점이다.

Reference

- BERT : Pre-training of Deep Bi-directional Transformers for Language Understanding
- Medium Dissecting BERT Part 1: The Encoder