카테고리 없음

[Euron] 8week_BERT : Pre-training of Deep Transformers f...

yejjji 2025. 4. 24. 01:53

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language

0. Abstract

BERT 라는 새로운 언어 표현 모델을 소개한다.
다른 최근의 언어 표현 모델들과 달리, BERT는 **모든 층에서 좌우 문맥 모두를 동시에 고려하여 라벨링되지 않은 문맥으로 깊은 양방향 표현들로 이루어진다.**
그 결과, 미리 훈련된 BERT 모델들은 미세 조정만으로 오직 1개의 추가 출력층으로 넓은 범위의 작업들에서 뛰어난 성능을 보인다.

BERT는 개념적으로 간단하고, 경험적으로도 강력하다.
11개의 다양한 자연어 처리 작업들에서 새롭게 기존보다 더 뛰어난 성능 결과를 내기도 하였다.

1. Introduction

언어 모델을 사전 학습시키는 것은 많은 자연어 처리 작업들에서 효과적이다.
먼저, 문장 수준으로는, 자연어 추론, 의역과 같은 문장 단위의 작업들을 포함하는데 이는 문장을 전체적으로 분석하여 문장 같의 관계가 어떤지 예측하는 것을 목표로 한다.

다음으로, 토큰 수준으로는, 세밀한 출력값을 생성하기 위해 모델이 필요로 하는 명명화된 entity 인식과 질문 응답과 같은 수준의 작업도 수행한다.

사전 학습된 언어 표현 적용 위한 2가지의 전략들이 존재한다. **하나는 feature 기반, 다른 하나는 미세 조정 기반**이다.
먼저, feature 기반의 경우 대표적으로 ELMo가 추가적인 feature값들로 사전 학습된 표현들을 포함한 작업 특이적 아키텍처를 사용한다.
반면에 미세 조정 기반의 경우, 대표적으로 GPT가 최소한의 작업 특이적 파라미터와 모든 사전 학습된 파라미터들을 미세 조정하여 다시 학습시킨다.

이 2개의 접근법 모두 사전 학습 과정에서 같은 목적함수를 고려하는데, 이는 두 방법이 일반적인 언어 표현 위해 **단방향성 언어 모델을 사용**한다는 것이다.

기존의 방식들은 사전 학습한 지식들을 충분히 잘 활용하지 못한다.
가장 주요한 한계점은 표준 언어 모델이 단방향성이라 사전 학습 과정에서 사용할 수 있는 아키텍처의 종류가 제한된다는 것이었다.

예를 들어, OpenAI GPT의 경우 left-to-right 아키텍처를 활용하는데, 이는 모든 토큰들이 자기 앞에 나온 단어들만 보고 참고할 수 있다.

이러한 여러 제한들은 문장 수준의 작업들에서는 최선이 아니다.
특히 질문 응답 같은 토큰 수준의 작업에서 미세 조정법을 적용할 때 매우 불리할 수 있다.
(왜? 문맥을 양방향으로 보고 파악하는 것이 핵심이니깐..!)

본 논문에서는 이러한 이유로 **BERT를 제안하며 미세 조정 기반의 접근법을 발전시킨다.**
실제로 BERT는 MLM 미세 조정 방식으로 이전에 언급한 단방향성을 완화시킨다.

또, 마스킹된 언어 모델은 Input으로부터 무작위로 몇몇 토큰들을 마스킹하는데, 이는 오직 문맥만으로 바스킹된 단어의 원래 단어가 무엇인지를 예측하는 것이 목적이다.

기존의 left-to-right 언어 모델들과 달리, BERT는 MLM 목적을 사용하여 **좌우 맥락을 동시에 고려하여 심층 양방향 Transformer 모델이 가능하도록** 한다.
이에 더불어, NSP를 활용하여 여러 개의 문장-쌍들에서 각각 서로끼리에 대한 표현들도 학습한다.

1) 언어 표현 측면에서 양방향 사전 학습의 중요성을 강조한다.
2) 사전 학습된 표현들은 대부분의 엔지니어링된 작업-특이적 아키텍처들에 대한 필요성을 감소시킨다.
3) BERT는 11개의 자연어 처리 작업들에서 최고 성능을 달성하였다.

2. Related Work

일반적인 언어 표현의 사전 학습 과정에는 긴 역사가 존재한다.
이 중에서 가장 널리 사용되는 것에 대해서만 말하고자 한다.

2.1 Unsupervised Feature-based Approaches(비지도 학습 기반의 특성 추출 접근 방식)

단어에 대해 광범위하게 적용 가능한 표현을 학습하는 것은 수십년 동안 활발히 연구되어 왔다.

사전 학습된 단어 임베딩은 현대 자연어 처리 분야에서 꽤나 중요한 부분으로, 처음부터 학습된 임베딩들보다는 성능적으로 중요한 영향을 끼친다.

단어 임베딩 벡터의 사전 학습을 위해서
1) 좌우 언어 모델링 목적
2) 좌우 문맥에서 올바른 단어를 식별하는 목적
이 사용된다.

이러한 접근법들은 문장, 문단 차원의 임베딩과 같이 더 넓은 단위로 일반화된다.
문장 표현의 학습을 위해
1) 후보 다음 문장을 순위화
2) 이전 문장으로부터 다음 문장 생성
3) 잡음 제거 등이 목적으로 사용된다.

(Peters et al., 2017, 2018a)
ELMo와 그 이전 모델은 기존의 단어 임베딩 연구를 문맥에 따라 달라지는 문맥 민감형으로 일반화한 것이다.

이 모델은 **left-to-right, right-to-left 양방향 언어 모델을 통해 각각의 토큰에 대한 문맥 표현을 추출**하며, 좌-우 표현을 결합한 벡터를 사용하는데, 이러한 문맥 임베딩을 기존의 작업별 구조에 통합할 경우, ELMo는 질문 응답, 감정 분석, 개체명 인식 등 다양한 NLP 벤치마크에서 최고 성능을 달성한다.

(Melamud et al. (2016))
LSTM을 사용해 좌우 문맥으로부터 중심 단어를 예측하는 방식을 제안했으며, 이 방식도 ELMo와 유사하게 feature 기반이며 완전한 양방향 구조는 아님을 강조한다.

(Fedus et al. (2018))
클로즈(cloze) 과제가 텍스트 생성 모델의 robustness을 향상시키는 데 효과적임을 보였다.

2.2 Unsupervised Fine-tuning Approaches(비지도 미세조정 접근 방식)

(Collobert and Weston, 2008)
feature 기반의 방식과 마찬가지로, 이 접근 방식의 초기 연구들도 **비지도 방식으로 단어 임베딩을 학습**하는 데 집중하였다.

(Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018)
최근에는 **문장 또는 문서 인코더**를 사전 학습하고, 지도 학습 작업에 미세조정하는 방식이 사용되고 있다.
이 방식의 장점은, **새롭게 학습해야 하는 매개변수가 적다**는 점이다.

(Radford et al., 2018), (Wang et al., 2018a)

이러한 장점 덕분에, **OpenAI GPT**는 **GLUE 벤치마크의 여러 문장 수준 과제에서 최고 성능을 기록**했다.

(Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015)
사전 학습에는 left-to-right방향 언어 모델링 및 오토인코더 목적이 사용되었다.

2.3 Transfer Learning from Supervised Data(지도 학습 데이터로부터의 전이 학습)

또한, **대규모 감독 학습 데이터셋을 활용한 전이 학습도 효과적이라는 것**이 여러 연구를 통해 입증되었는데, 그 예로 자연어 추론(NLI), 기계 번역(MT) 등이 있다.

(Deng et al., 2009; Yosinski et al., 2014)
실제로 컴퓨터 비전 분야에서는, ImageNet과 같은 대규모 데이터셋으로 사전 학습한 모델을 **미세조정하는 것이 강력한 성능을 내는 핵심 방법**이라는 것이 이미 입증되기도 하였다.

3. BERT

이 섹션에서는 **BERT의 구조와 구현 방식**에 대해 설명한다.

BERT 프레임워크는 **두 단계**로 구성되는데, **사전 훈련** 과 **미세 조정 방식이 있**다.
먼저, 사전 훈련 단계에서는 다양한 pre-training 과제를 기반으로 하여 **비지도 데이터로 모델을 훈련**시킨다.
미세 조정 단계에서는 사전 훈련된 파라미터로 초기화된 모델을 이용해, 라벨이 있는 다운스트림 과제에 대해 **모든 파라미터를 미세 조정**한다.

각 downstream 과제는 **동일한 사전 훈련 파라미터로 초기화되지만, 개별적으로는 각각 조금씩 다르게 미세 조정된 모델을 사용한다**.

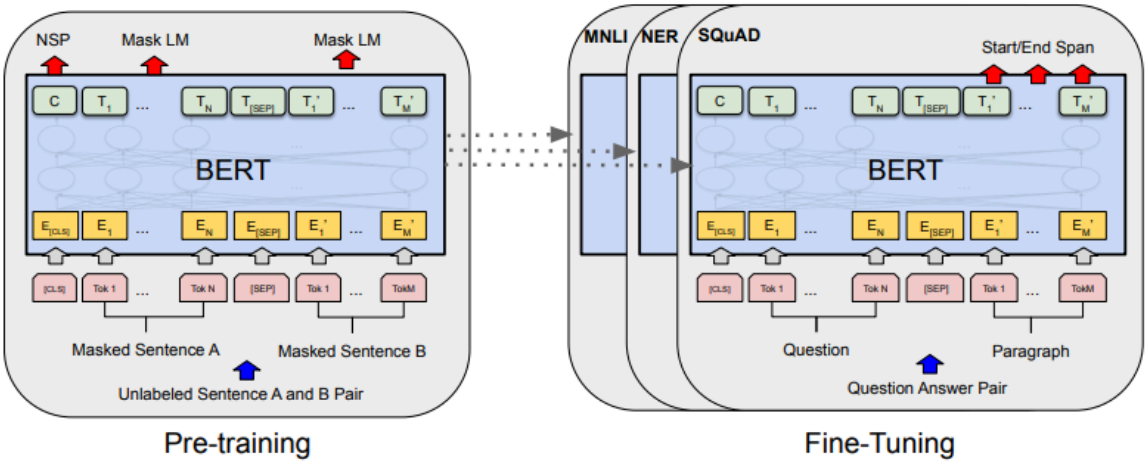

Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Figure 1 : BERT의 전체 구조와 학습 방식 왼쪽(Pre-training) / 오른쪽(Fine-Tuning)

1. **대규모의 비지도 데이터**로 사전 학습(pre-training)

- Masked LM (MLM)

→ 문장에서 일부 토큰을 [MASK]로 가리고, 이 가려진 토큰을 맞추도록 학습함

→ 예: "나는 [MASK]을 좋아해" → "[MASK] = 떡볶이"

- Next Sentence Prediction (NSP)

→ 문장 A 다음에 문장 B가 실제로 이어지는 문장인지 여부를 맞추는 이진 분류 작업.

→ 예:

- A: "오늘 날씨가 좋다."
- B: "그래서 우리는 소풍을 갔다." → ✔
- B: "나는 개를 키우고 싶다." → ❌

2. **Fine-tuning (미세 조정)**

사전 학습된 BERT는 다양한 downstream task에 맞게 **미세 조정(fine-tuning)됨**.

구조는 동일하지만, **출력층만 바뀐다**.

각 task에 대한 예시:

- MNLI, NER

→ [CLS] 벡터를 활용해서 문장의 클래스를 분류하거나, 각 토큰에 대해 태그 예측

- SQuAD (질문-응답)

→ 문단(Paragraph)과 질문(Question)을 같이 입력으로 넣고, 정답이 문단에서 시작되는 토큰과 끝나는 토큰의 위치를 예측

→ 이때 Start/End를 위한 벡터도 따로 학습 (Start Span, End Span)

📌 특징: 1) 사전 학습에서 학습한 모든 가중치를 그대로 가져와서 사용하고, **모든 파라미터를 다시 학습(fine-tune)**

2) [CLS]는 classification 태스크에 사용, [SEP]는 문장 구분용.

즉, **사전 학습**을 통해 일반적인 언어 지식을 익히고, **미세 조정**을 통해 각 태스크에 맞는 방식으로 학습하여 성능을 내는 원리!

출처 : 챗 GPT

Model Architecture

BERT는 다층 **양방향 Transformer 인코더** 구조를 가지고 있으며, 이는 Vaswani et al. (2017)의 원래 Transformer 구조를 기반으로 한다.

- 레이어 수: L
- 히든 사이즈: H
- 셀프 어텐션 헤드 수: A

대표적인 모델로는 2개가 있다.

1) BERTBASE: L=12, H=768, A=12, 파라미터 : 1억 1천만 개

2) BERTLARGE: L=24, H=1024, A=16, 파라미터 : 3억 4천만 개

GPT와의 주요 차이점은, **좌측 맥락만을 활용하는 제한된 self-attention**을 사용하는 반면, **BERT는 완전한 양방향 self-attention을 사용한다**는 점입니다.

Input/Output Representation

BERT는 한 문장 또는 문장 쌍을 명확히 표현할 수 있는 입력 구조를 가진다.

- 문장은 실제 언어학적 문장이 아닌 연속된 텍스트 구간(span) 일 수 있다.

- 입력 시퀀스는 하나 또는 두 개의 문장을 포함할 수 있으며, 두 문장은 [SEP] 토큰으로 구분된다.
- 각 토큰에는 **해당 문장이 A인지 B인지 나타내는 세그먼트 임베딩이 추가**된다.
- 모든 시퀀스는 [CLS] 토큰으로 시작되며, **이 [CLS] 토큰의 마지막 히든 상태는 분류 과제에서 전체 시퀀스를 대표하는 벡터로 사용**된다.

입력 임베딩은 토큰 임베딩 + 세그먼트 임베딩 + 위치 임베딩를 합한 값과 같다.

3.1 Pre-training BERT

기존의 left-to-right 또는 right-to-left 언어 모델 방식이 아닌, BERT는 2가지의 **비지도 학습 과제**를 통해 사전 훈련된다.

Task #1: Masked Language Modeling(MLM)

일반적인 조건부 언어 모델은 한 방향(left->right / right->left)만 훈련할 수 있기 때문에 진정한 양방향 문맥 정보를 반영하기 어렵다.
이를 해결하기 위해 BERT는 입력 토큰의 15%를 무작위로 마스킹(masking) 한 후, 마스킹된 토큰을 예측하는 훈련을 수행한다. **이를 "Masked LM" 또는 "Cloze 과제"라고 부른다.**
마스킹 전략은 크게 3가지가 혼합된 형태이다.
- 80%: [MASK]로 대체
- 10%: 랜덤한 다른 토큰으로 대체
- 10%: 변경 없이 그대로 유지

이렇게 하면 양방향 문맥 정보를 활용하는 언어 모델을 훈련할 수 있으며, 전통적인 오토인코더와 달리 전체 문장을 복원하지 않고 마스킹된 단어만 예측한다.

Task #2: Next Sentence Prediction (NSP)

질문 응답, 자연어 추론 등 문장 간 관계 이해가 중요한 작업에 대비하여, NSP를 도입하였다.

훈련 데이터 중 50%는 문장 B가 실제로 문장 A 다음에 오는 문장 (→ IsNext)이고, 나머지 50%의 경우 문장 B가 무작위로 선택된 문장 (→ NotNext)을 의미한다.

NSP는 문장 간 관계를 학습하는 데 매우 효과적이며, 전체 모델 파라미터를 전이(transfer)할 수 있다는 점에서 기존 방식보다 강력하다.

Pre-training data

- BooksCorpus (8억 단어) + 영문 Wikipedia (25억 단어) 사용
- Wikipedia는 텍스트만 추출 (리스트, 표, 헤더 제외)
- 문장 단위가 아닌 문서 단위로 긴 시퀀스를 학습하는 것이 핵심

3.2 Fine-tuning BERT

BERT는 Transformer의 self-attention 메커니즘을 활용해 다양한 downstream 과제를 직접 처리 가능하다.
텍스트 쌍 과제에서도 BERT는 **두 문장을 하나의 입력 시퀀스로 연결**하고, **자체적으로 양방향 상호작용을 모델링**한다.
즉, **각 작업에 맞게 입력과 출력값만 살짝 바꾸고, 모든 파라미터를 end-to-end로 미세 조정한다는 것이다.**

4. Experiments

BERT를 다양한 NLP 작업에 fine-tuning한 결과를 보여준다.

4.1 GLUE

GLUE(General Language Understanding Evaluation)는 여러 자연어 이해 작업들의 모음이다.
Fine-tuning할 때 만든 입력 시퀀스를 가지고, 첫 번째 토큰([CLS])에 해당하는 최종 히든 벡터 C를 전체 문장의 대표값으로 사용한다.
새로 도입되는 파라미터는 분류층 가중치 W (크기 K×H)인데, 이때 K는 레이블 수를 의미한

다.

loss function은 C와 W를 사용해서 softmax classification loss를 계산한다.

이때, batch size는 32, epoch은 3번이고, learning rate는 5e-5, 4e-5, 3e-5, 2e-5 중에서 개발셋에서 가장 좋은 rate값을 선택한다.

이때 BERTLARGE는 작은 데이터셋에서 불안정한 경우가 있어서 랜덤으로 여러 번 학습 재시작해서 제일 좋은 모델을 선택한다.

BERTBASE랑 BERTLARGE 두 모델 모두 모든 task에서 기존 모델들보다 성능이 훨씬 좋다는 것을 확인할 수 있었다.

BERTBASE는 기존 최고 모델보다 평균 정확도가 4.5% 높았으며, BERTLARGE는 무려 7.0% 정도 높다는 것을 알 수 있었다.

특히 MNLI 태스크에서는 BERT가 정확도 4.6% 향상을 보여주었고, GLUE 리더보드에서 BERTLARGE가 80.5점으로 72.8점의 GPT보다 훨씬 좋았다.

BERTLARGE는 BERTBASE보다 모든 task에서 성능이 더 좋았고, 특히 데이터가 적은 경우 성능에서 더 큰 차이를 보여주었다.

4.2 SQuAD v1.1

SQuAD는 위키피디아 문단과 질문이 주어졌을 때, 정답 구간을 찾는 문제이다.

- 입력: 질문(A 임베딩)과 문단(B 임베딩)을 하나의 시퀀스로 합쳐 넣은 형태
- 새로 추가되는 파라미터: 시작 벡터 S, 종료 벡터 E
- 답이 될 단어 i의 확률은 Ti와 S의 내적값에 소프트맥스를 적용해서 계산

System	Dev		Test	
	EM	F1	EM	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				
$\text{BERT}_{\text{BASE}}$ (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

< Figure2에서 확인할 수 있는 것 >

- 단일 BERT 모델이 기존 최고 앙상블 모델보다 F1 점수가 더 높다.

- TriviaQA 데이터를 먼저 파인튜닝한 후 SQuAD에 파인튜닝해서 더 좋은 성능을 낸다.

- TriviaQA 안 쓰고도 기존 모델들보다 성능이 훨씬 좋다.

4.3 SQuAD v2.0

답이 아예 존재하지 않는 경우도 포함된 문제이다.

이때, 답이 없는 질문은 [CLS] 토큰을 시작과 끝 위치로 간주한다.

no-answer 점수는 $S \cdot C + E \cdot C$로 계산하고, 가장 좋은 구간 점수 $\hat{s}_{i,j}$와 비교해서 $\hat{s}_{i,j}$가 더 크면 그 구간을 답으로 예측하는 원리이다.

System	Dev		Test	
	EM	F1	EM	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				
$\text{BERT}_{\text{LARGE}}$ (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.

기존의 최고 모델보다 F1 점수 +5.1 상승하였다는 것을 확인할 수 있었다.

4.4 SWAG

SWAG는 **상황에 맞는 문장을 이어쓰기하는 문제이다.**
문장이 주어지고, 그 뒤에 이어질 가장 그럴듯한 후속 문장을 고르는 문제라는 것이다.

입력은 문장 A와 4개의 후보 문장 B 중 하나를 합쳐서 총 4개의 시퀀스를 만들고,
[CLS] 벡터에 dot product로 점수를 매겨서 softmax로 가장 높은 후보를 선택한다.

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**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.

그 결과,BERTLARGE는 기존 ESIM+ELMo 모델보다 **27.1%나 향상되었고,** GPT보다도
8.3%가 향상되었다는 사실을 확인할 수 있었다.

5. Ablation Studies

BERT의 여러 구성 요소가 얼마나 중요한지를 알아보기 위해 Ablation 실험을 수행하였다.

5.1 Effect of Pre-training Tasks

BERT의 양방향성(bidirectionality)이 얼마나 중요한지를 보여주기 위해 두 가지 사전학습 방식을 비교하였다.
모두 동일한 데이터를 가지고, 같은 하이퍼파라미터와 fine-tuning 방식을 사용한다.

1) No NSP: masked LM(MLM)만 사용하고 Next Sentence Prediction (NSP)은 사용하지 않은 모델이다.
2) LTR & No NSP: left-to-right 방향으로 읽는 언어 모델로, GPT랑 비슷하게 만들었고, 역시 NSP는 없다. fine-tuning할 때에도 LTR로 제한하였다.

	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

위 표의 계산 결과 NSP를 빼면 QNLI, MNLI, SQuAD 성능이 확 떨어지는 것을 확인할 수 있었다.
또, MLM 방식이 LTR보다 항상 성능이 좋았고, 특히 MRPC랑 SQuAD에서 **차이가 크다는 것도 확인할 수 있었다.**

참고로, ELMo처럼 LTR과 RTL 모델을 따로 학습해서 붙이는 방법도 있지만, 계산량이 2배가 되고, 질문응답처럼 질문-문맥 관계를 고려해야 하는 작업엔 비직관적인 방법이다.
또, **BERT처럼 모든 층에서 양방향 정보를 쓰는 게 더 강력하다.**

5.2 Effect of Model Size

BERT 모델의 크기 차이가 fine-tuning의 성능에 어떤 영향을 주는지 실험하였다.
이때, layer의 수, hidden unit의 수, attention head의 수를 바꾸면서 실험하였고, 나머지 조건은 동일하게 만들었다.

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.

GLUE의 여러 task에서 실험했고, 모든 task에서 **모델이 클수록 성능이 좋아졌다.**
심지어 데이터가 적은 MRPC에서도 성능 향상되었고, 사전학습이 충분히 잘 된 경우, 작은 task에서도 **모델 크기의 효과가 컸다.**

feature-based 방식의 경우 모델이 너무 커지면 효과가 없다는 말도 있었는데, 본 논문은 **fine-tuning 기반이기 때문에 대형 모델의 표현력을 그대로 활용이 가능하였다.**

5.3 Feature-based Approach with BERT

BERT는 **고정된 벡터**만 뽑고, 그 위에 가볍고 실험하기 쉬운 모델을 올리는 방식이다.
이를 NER 태스크(CoNLL-2003)에 적용해보았다. 입력의 경우 WordPiece 토크나이저를 그대로 쓰고, 문맥 정보를 최대한 살렸으며, 각각의 단어 첫 번째 sub-token 벡터를 이용해서 분류하였다.

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.

위의 표에서도 확인할 수 있듯이, BERTLARGE는 **SOTA급 성능**을 낸다는 것을 확인할 수 있었다. 또, **상위 4개 층의 벡터를 concat한 feature-based 방식이 fine-tuning보다 겨우 0.3 만큼만 F1값이 낮았다는 것도 확인할 수 있었다.**

6. Conclusion

최근의 경험적 연구들은 언어 모델을 활용한 전이 학습이 성능 부분에서 큰 향상을 가져오며, **풍부한 비지도 사전학습**이 많은 언어 이해 시스템에서 핵심적인 역할을 한다는 것을 보여주었다.

특히, 이런 결과들은 데이터가 적은 작업에서도 **깊은 단방향(unidirectional) 모델 구조**의 장점을 쓸 수 있게 해주었다.
본 논문의 가장 큰 기여는 바로 이걸 더 일반화해서, **깊은 양방향(bidirectional) 구조에도 적용**했다는 사실이다.