

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

📅 Due Date	@2021년 5월 16일
📎 File	https://arxiv.org/pdf/1901.11504.pdf
👤 Presenter	유경 한
⋮ Tags	Now Presentation
≡ Title	
≡ 속성	

기본적으로 BERT를 개선한 모델!

1. Introduction

Abstract

MT-DNN은 BERT에 Multi-task learning(GLUE TASK 9개 활용)을 수행하여 성능 개선한 모델

- 다양한 Task의 Supervised Dataset을 모두 사용하여 대용량 데이터로 학습
- Multi-task Learning을 통해 특정 Task에 Overfitting되지 않도록 Regularization

다음 Task에서 SOTA 성능(BERT보다 높음)

- 8개의 GLUE Task
- SNLI와 SciTail Task로 Domain Adaptation : Fine Tuning 데이터가 적을 때에도 Classification 정확도가 꽤 높음

Language Representation "

: nlp에서 word 혹은 sentence의 vector representation을 생성하는 방법 두가지

Language Model Pre-Training

- unlabeled dataset을 활용한 학습 방법
- 대표적으로 문장에서 특정 단어를 맞추는 방식으로 Unsupervised Learning
- ELMO, BERT 등등

Multitask learning

- 여러 Task의 Labeled Dataset을 활용하여 1개의 모델 Supervised Learning
- 어떤 Task에서 학습 효과가 다른 Task의 성능에 영향을 미칠 것이라는 가정 : ex) 스키를 잘 타는 사람이 스케이트를 잘 탈 수 있다.

Language Model Pre-Training - BERT

Book Corpus, Wikipedia Data를 활용하여 다음 2가지 방식으로 학습

Masked Word Prediction

- 문장이 주어졌을 때 특정 Word를 Masking하고 다른 주변 Word들을 활용하여 해당 Word를 예측하는 방식으로 학습 : ex) my dog is [Mask] → my dog is hairy

Next Sentence Prediction

- 문장이 2개 주어졌을 때, 2개의 문장이 연결된 문장인지 아닌지를 Classification하는 방식으로 학습 : ex) Input = the man went to the store [SEP] he bought a gallon of milk → IsNext

Multi Task Learning

Supervised Task를 1개의 모델을 통해 학습

- GLUE의 9개 Task 활용 #애를 한 모델을 통해 한꺼번에 학습하는 것이 본 논문의 메인 아이디어

MTL의 이점

- 대용량 Supervised Dataset을 활용하여 학습 가능
- 모델이 특정 Task에 Overfitting되지 않도록 Regularization 효과를 줄 수 있음

2. Tasks

#9개 glue 태스크를 본 논문에서는 다음의 4가지로 분류

Single Sentence Classification

하나의 문장이 Input으로 주어졌을 때 class를 분류하는 Task - CoLA(문장이 문법적으로 맞는지 분류), SST-2(영화 Review 문장의 감정 분류)

Text Similarity

문장 쌍이 주어졌을 때, 점수를 예측하는 Regression Task - STS-B(문장 간의 의미적 유사도를 점수로 예측)

Pairwise Text Classification

문장 쌍이 주어졌을 때, 문장의 관계를 분류하는 Task - RTE,MNLI(문장 간의 의미적 관계를 3가지로 분류 : Entailment, Contradiction, Neutral), QQP,MRPC(문장 간 의미가 같은 여부를 분류)

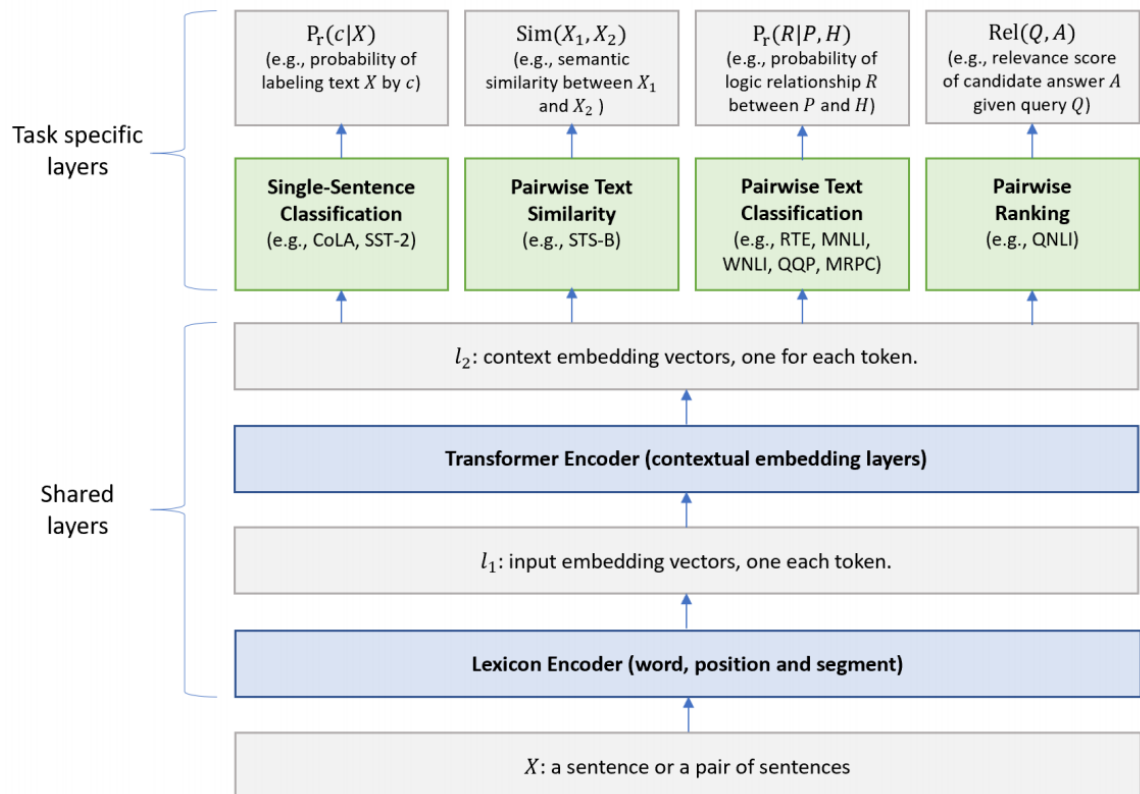
Relevance Ranking

질문 문장과 지문이 주어졌을 때, 지문 중 정답이 있는 문장을 Ranking을 통해 찾는 Task - QNLI(질문, 지문 중 한 문장이 쌍으로 주어졌을 때 해당 지문 문장에 질문의 답이 있는지 여부 분류 → MT DNN에서는 이를 Rank 방식으로 바꾸어 모든 지문 문장에 정답이 있을 가능성을 점수화하여 가장 높은 지문 문장만을 True로 분류함)

이 4개로 분류된 9개 태스크를 mt dnn 이 버트라는 하나의 모델로 학습해서 성능을 높였다는 것이 본 논문의 핵심 아이디어

3. The Proposed MT-DNN Model

MT-DNN model의 구조



하위 layer(BERT의 Transformer 구조와 동일)는 모든 task에 공유되고, 가장 상위의 레이어는 task별로 다르게 구성.

shared layer는 버트와 동일, task specific layer는 버트와 같은것도 다른것도 있음

Lexicon Encoder (l_1) :

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	#ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

BERT의 임베딩 방식과 동일. $X = \{x_1, \dots, x_m\}$ sequence가 input으로 들어오면 Embedding vector로 바꿔줌.

인풋을 생성하는 layer 로 ~의 세가지 임베딩으로 구성

token embedding 각 토큰에 대한 임베딩

sentence~ : 첫번째 문장인지 두번째 문장인지를 알려주는 임베딩

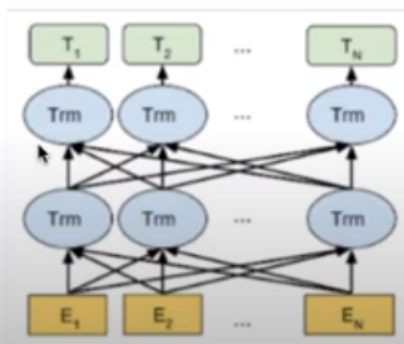
position~ : 각 토큰의 위치에 대한 임베딩

이런식으로 인풋이 형성이 되면 이를 트랜스포머의 셀프어텐션에 넣어 각 토큰의 임베딩 벡터를 추출함

Transformer Encoder (l_2) :

Model – Transformer Encoder

- Lexicon Encoder로 부터 각 Token의 Input Vector를 입력으로 받아 Transformer를 통해 각 Token의 Output Vector를 추출
- Transformer이기 때문에 생성된 Vector는 Self Attention을 통해 주변 Token 정보를 반영한 Contextual Embedding Vector



Lexicon Encoder (l_1) 의 임베딩 벡터를 입력으로 받아서 multilayer bidirectional Transformer encoder 를 사용하여 각 Token의 Output Vector를 추출함. 이때 BERT에 더해서 MT-DNN 은 multi-task objectives를 사용하여 학습함.

Single-Sentence Classification Output #버트와 같은 레이어

Model – Single Sentence Classification

- Single Sentence Classification
 - CoLA - 문장이 문법적으로 맞는지 Classification
 - SST-2 – Movie Review 문장의 Sentiment Classification (Positive / Negative)

$$P_r(c|X) = \text{softmax}(\mathbf{W}_{SST}^T \cdot \mathbf{x})$$

The diagram shows the formula $P_r(c|X) = \text{softmax}(\mathbf{W}_{SST}^T \cdot \mathbf{x})$ with four labels pointing to its components: 'class' points to $P_r(c|X)$, 'Input Sentence' points to \mathbf{x} , 'Task specific Parameter' points to \mathbf{W}_{SST}^T , and 'Class Token' points to the c in the class label.

Transformer Encoder (l_2) 로부터 얻은 output vector의 [CLS] 토큰은 contextual embedding 을 진행한 벡터로 볼 수 있음.

하나의 문장을 분류하는 Classification Output은 다음과 같이 계산

$$P_r(c|X) = \text{softmax}(\mathbf{W}_{SST}^T \cdot \mathbf{x})$$

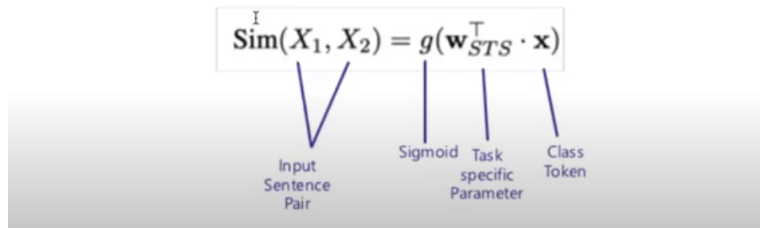
: [CLS] Token 과 Task Specific Parameter 의 곱에 Softmax를 취하여 각 Input Sentence \mathbf{x} 가 각 class에 속할 확률을 계산

- \mathbf{W}_{SST}^T : Task specific Parameter
- \mathbf{x} : Input Sentence

Text Similarity Output : #위와 비슷 리그레션이니까 sigmoid로 바꾼 것

Model – Text Similarity Output

- Text Similarity
 - STS-B – 문장 Pair의 Similarity를 1~5점으로 Regression



Transformer Encoder (l_2) 로부터 얻은 output vector의 [CLS] 토큰은 문장쌍 (X_1, X_2)의 semantic representation으로 볼 수 있음.

Sentence pair로 이루어진 (X_1, X_2)의 유사도는 다음과 같이 계산

$$\text{Sim}(X_1, X_2) = \mathbf{w}_{STS}^T \cdot \mathbf{x},$$

- w_{STS}^T : Task Specific Parameter

Pairwise Text Classification Output :

Model – Pairwise Text Classification Output (1)

Pairwise Text Classification

- MNLI – 문장 Pair의 entailment, contradiction, neutral을 Classification
- RTE – 문장 Pair의 Entailment 여부를 Binary Classification
- QQP, MRPC – 문장 Pair의 의미 같음 여부를 Binary Classification

Stochastic Answer Network(SAN)를 활용하여 Classification

- 핵심 Idea - Multi-step Reasoning
 - RNN을 활용하여 2번 이상의 예측을 통해 문장 간 관계를 분류
- 다음 2가지 문장이 있을 때 한번에 문장 간 관계를 분류하기는 쉽지 않음
 - If you need this book, it is probably too late unless you are about to take an SAT or GRE
 - It's never too late, unless you're about to take a test.
 - 두 문장이 Contradiction이라고 분류하기 위해서는 SAT, GRE가 Test라고 예측 단계가 먼저 필요함

NLI task를 예로 들면, 전제 premise $P = (p_1, \dots, p_m)$ 와 가설 hypothesis $H = (h_1, \dots, h_n)$ 이 있을 때, 이 둘의 논리적 관계인 R 을 찾는 것이 목적. 이를 위해 stochastic answer network (SAN)을 사용하는데, RNN을 이용하여 K-step Reasoning을 하는 것이 특징

#두 문장이 있을 때 한번에 문장 간 관계를 분류하기가 쉽지 않음 . 여러 단계가 필요하겠다는 것이 main idea

각 K step 마다,

$$P_r^k = \text{softmax}(\mathbf{W}_3^\top [\mathbf{s}^k; \mathbf{x}^k; |\mathbf{s}^k - \mathbf{x}^k|; \mathbf{s}^k \cdot \mathbf{x}^k])$$

- $\mathbf{s}^k; \mathbf{x}^k$: 각 문장의 vector
- $|\mathbf{s}^k - \mathbf{x}^k|$: 문장간 거리
- $\mathbf{s}^k \cdot \mathbf{x}^k$: 문장간 유사도

두 문장 자체의 Embedding Vector, 그리고 두 문장 간 관계(차의 크기와 Dot Product)를 concat하여 구성된 Vector를 활용하여 문장 간 관계를 분류


→ 최종적으로 모든 K step에서 나온 output vector를 평균

$$P_r = \text{avg}([P_r^0, P_r^1, \dots, P_r^{K-1}])$$

Relevance Ranking Output :

QNLI task를 가정하면, 다음과 같이 relevance score을 계산

$$\text{Rel}(Q, A) = g(\mathbf{w}_{QNL I}^\top \cdot \mathbf{x})$$

- (Q , A) : input sentence pair
-  : the contextual embedding vector of [CLS] which is the semantic representation of a pair of Question
- g : sigmoid 함수
- $w_{QNL I}^T$: Task specific parameter

3.1 The Training Procedure

MT-DNN은 훈련 절차

- Pretraining
 - BERT의 모델을 따름.
 - lexicon encoder와 Transformer encoder의 파라미터는 mask language modeling과 next sentence prediction 이 두가지의 unsupervised prediction tasks에 의해 학습됨.
- Multi-task learning
 - 모델의 파라미터를 학습하기 위해 SGD 기반의 mini-batch를 사용.
(선택하는 방법은 모든 task(9개의 GLUE)의 데이터세트를 모아다가 batch를 끄집어내는 식)
 - 각 epoch마다 미니배치 b_t 가 선택되고 task t 에 대한 task-specific objective에 따라 모델이 업데이트됨.
 - 모든 다중 작업 목표의 합계를 거의 최적화함

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 $1, 2, \dots, T$ **do**
 | Pack the dataset t into mini-batch: D_t .
end
for $epoch$ in $1, 2, \dots, epoch_{max}$ **do**
 1. Merge all the datasets:
 $D = D_1 \cup D_2 \dots \cup D_T$
 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 task에서는 cross-entropy loss를 사용함.

$$- \sum_c \mathbb{1}(X, c) \log(P_r(c|X)), \quad (6)$$

- $\mathbb{1}(X, c)$ 는 이진수 표시기(맞는 class이면 1, 아니면 0)
- P_r 은 위의 식 1,4에 정의됨

- text similarity tasks(ex.STS-B)에서는 각 문장쌍이 실제 값 점수 y 로 명시되어 있으면 MSE를 사용함.

$$(y - \text{Sim}(X_1, X_2))^2, \quad (7)$$

- Sim은 위의 식 2에서 정의됨

- relevance ranking tasks에서는 pairwise learning-to-rank 패러다임 사용함.
QNLl를 예로 들면 쿼리 Q 가 주어지면, 정답을 포함하는 긍정적인 예 A^+ 와 $|A|-1$ 개의 부정적인 예를 포함하는 후보 답변 목록을 얻음. (8)

$$- \sum_{(Q, A^+)} P_r(A^+|Q), \quad (8)$$

$$P_r(A^+|Q) = \frac{\exp(\gamma \text{Rel}(Q, A^+))}{\sum_{A' \in \mathcal{A}} \exp(\gamma \text{Rel}(Q, A'))}, \quad (9)$$

(9) Rel은 위의 식 (5)에서 정의됨.

γ (감마)은 tuning factor로 데이터에 따라 결정되며 이 실험에선 단순히 1로 정한다.

$A = A^+ \cup A^-$ (A^+ : 정답, A^- : negative sample)

다른 후보들에 비해 정답이 가장 큰 확률을 갖도록 정답 확률의 negative log likelihood를 최소화하는 방향으로 학습.

4. Experiments

GLUE, SNLI, SciTail의 세가지 인기있는 NLU 벤치마크에서 MT-DNN을 평가한다.

4.1 Datasets

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

- GLUE(General Language Understanding Evaluation)
: 질의 응답, 정서 분석, 텍스트 유사성 및 텍스트 수반을 포함한 9개의 NLU 과제 모음
NLU 모델의 일반화와 견고성을 평가하기 위해 설계됨
- SNLI(Stanford National Language Inference)
: 570k 개의 human annotated 문장쌍이 포함되어 있음. NLI에 대해 가장 많이 사용됨
이 논문에서 Domain adaptation을 위한 데이터셋으로 사용
- SciTail
: SciQ(과학 질의응답) 데이터셋에서 도출된 텍스트 수반 데이터셋
주어진 전제가 주어진 가설을 포함하는지 여부를 평가하는 것을 포함
언어학적으로 어려운 내용이기 때문에 challenge함
이 논문에서 Domain adaptation을 위한 데이터셋으로 사용

4.4 Domain Adaptation Results on SNLI and SciTail

실험 방법

1. MT-DNN or BERT를 initial model로 함(base, large 모델 세팅 포함)
2. task-specific training data로 학습된 MT-DNN을 적용시켜서 SNLI, SciTail 각각에 맞는 모델을 만들
3. task-specific test data로 평가

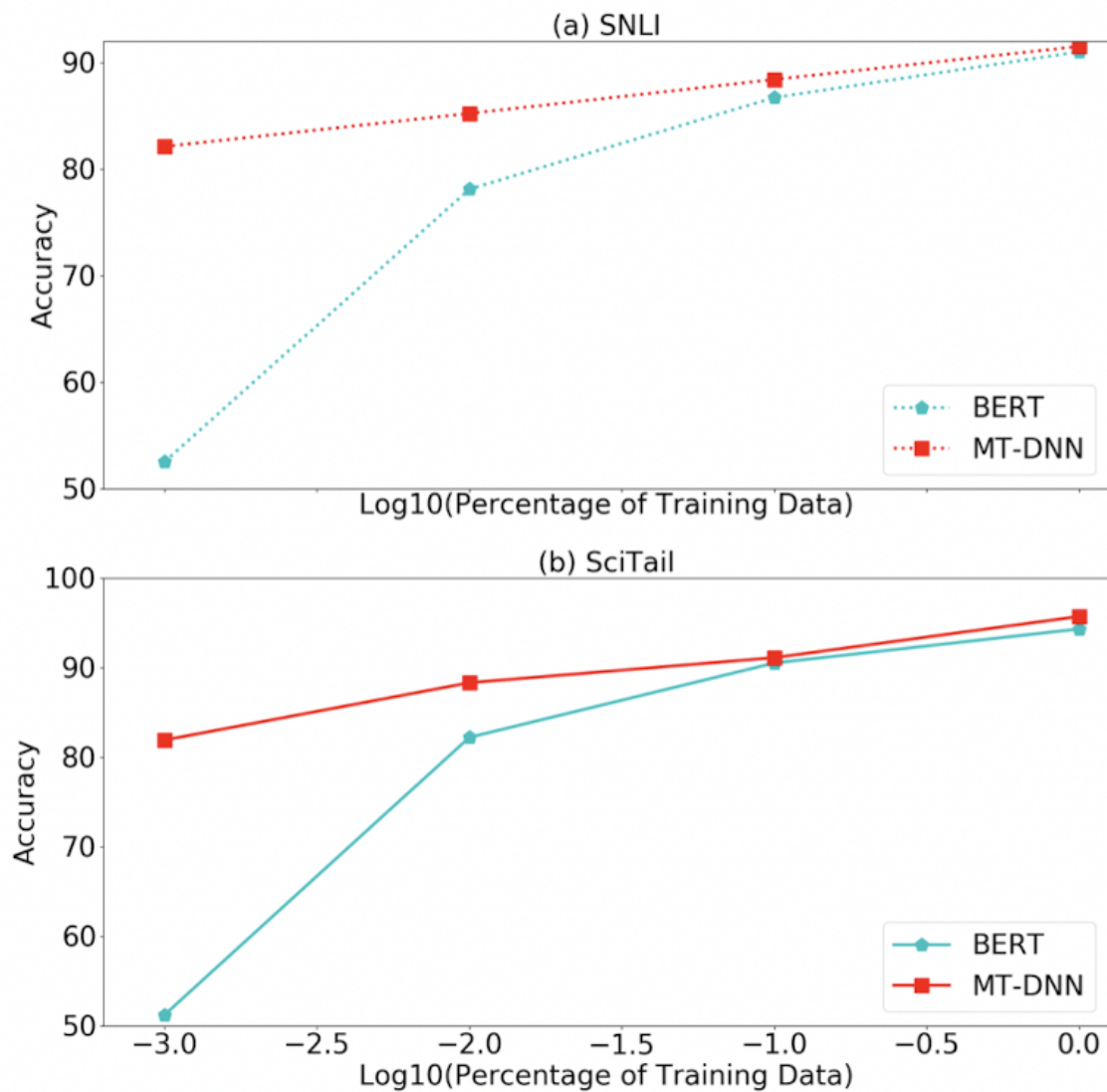


Figure 2

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

Table 4: Domain adaptation results on SNLI and SciTail, as shown in Figure 2.

실험결과

- 더 적게 train할수록 MT-DNN 더 성능 발달
- BERT보다 domain adaptation에 일관되게 효과적임

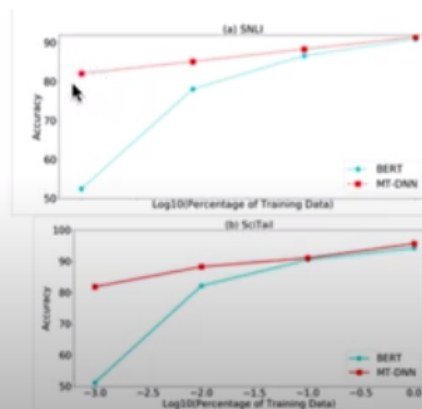
Adapted model을 비교해본 결과, 다른 모델에 비해 MT-DNN(LARGE)가 두 데이터셋에서 모두 SOTA 상태.

Model	Dev	Test
SNLI Dataset (Accuracy%)		
GPT (Radford et al., 2018)	-	89.9
Kim et al. (2018)*	-	90.1
BERT _{BASE}	91.0	90.8
MT-DNN _{BASE}	91.5	91.1
BERT _{LARGE}	91.7	91.0
MT-DNN _{LARGE}	92.2	91.6
SciTail Dataset (Accuracy%)		
GPT (Radford et al., 2018)*	-	88.3
BERT _{BASE}	94.3	92.0
MT-DNN _{BASE}	95.7	94.1
BERT _{LARGE}	95.7	94.4
MT-DNN _{LARGE}	96.3	95.0

Table 5: Results on the SNLI and SciTail dataset. Previous state-of-the-art results are marked by *, obtained from the official SNLI leaderboard (<https://nlp.stanford.edu/projects/snli/>) and the official SciTail leaderboard maintained by AI2 (<https://leaderboard.allenai.org/scitail>).

Evaluation – Domain Adaptation

- BERT, MT-DNN에 각각 새로운 Task Specific Layer를 붙여서 학습 및 평가 결과
 - NLI Dataset인 SNLI, SciTail을 활용하여 Domain Adaptation 평가
 - 0.1%, 1%, 10%, 100%로 평가
- 데이터가 적을 때 MT-DNN의 성능이 BERT 보다 훨씬 높은 성능을 보임



5 Conclusion

MT-DNN

- MTL과 Language representation learning을 위한 LM pre-training을 접목시킨 것
- SNLI, SciTail, GLUE에 대해 10개 NLU task에서 SOTA의 결과를 냄
- Domain adaption 실험에서 일관되게 좋은 결과.