MT - DNN

Multi-Task Deep Neural Networks for Natural Language Understanding

NLP 2조

0. Abstract

◆ MT-DNN은 BERT에 Multi-task learning(GLUE TASK 9개 활용)을 수행한 성능개선 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

1. Introduction

Language Representation

◆ Language Model Pre-Training

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

◆ Multi-task 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 스케이트 를 잘 탈 수 있다.

1. Introduction

◆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 Classification 하도록 학습

  ex) Input = the man went to the store [SEP] he bought a gallon of milk → IsNext

1. Introduction

◆Multi Task Learning

- Supervised Task를 1개의 모델을 통해 학습

  - GLUE의 9개 Task (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

- **Single Sentence Classification :** 하나의 문장이 Input으로 주어졌을 때 class 분류 Task
  - CoLA(문장이 문법적으로 맞는지 분류), SST-2(영화 Review 문장의 감정 분류)

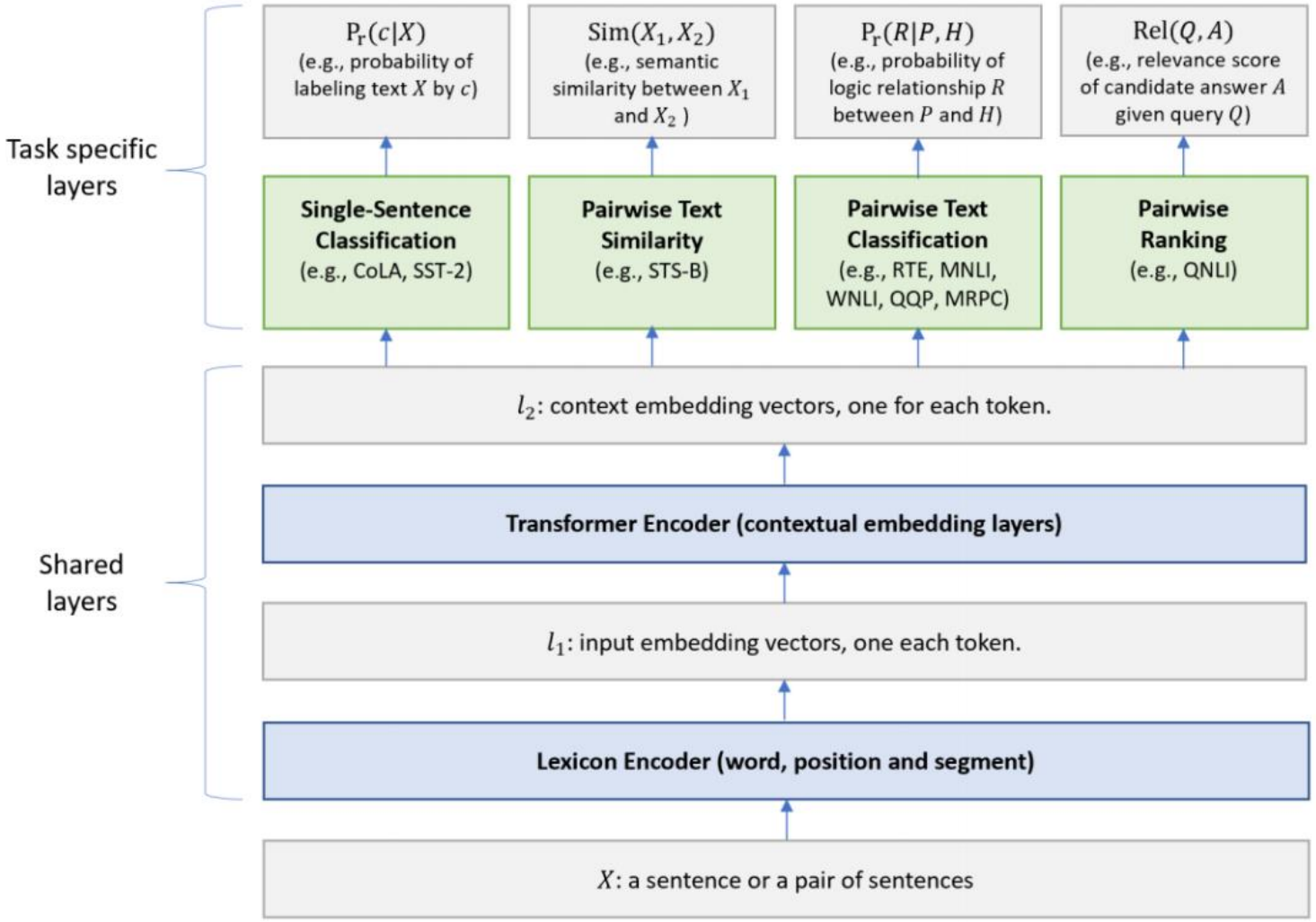
- **Text Similarity :** 문장 쌍이 주어졌을 때, 점수를 예측하는 Regression Task
  - STS-B(문장 간의 의미적 유사도를 점수로 예측)

- **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 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 Task 수행

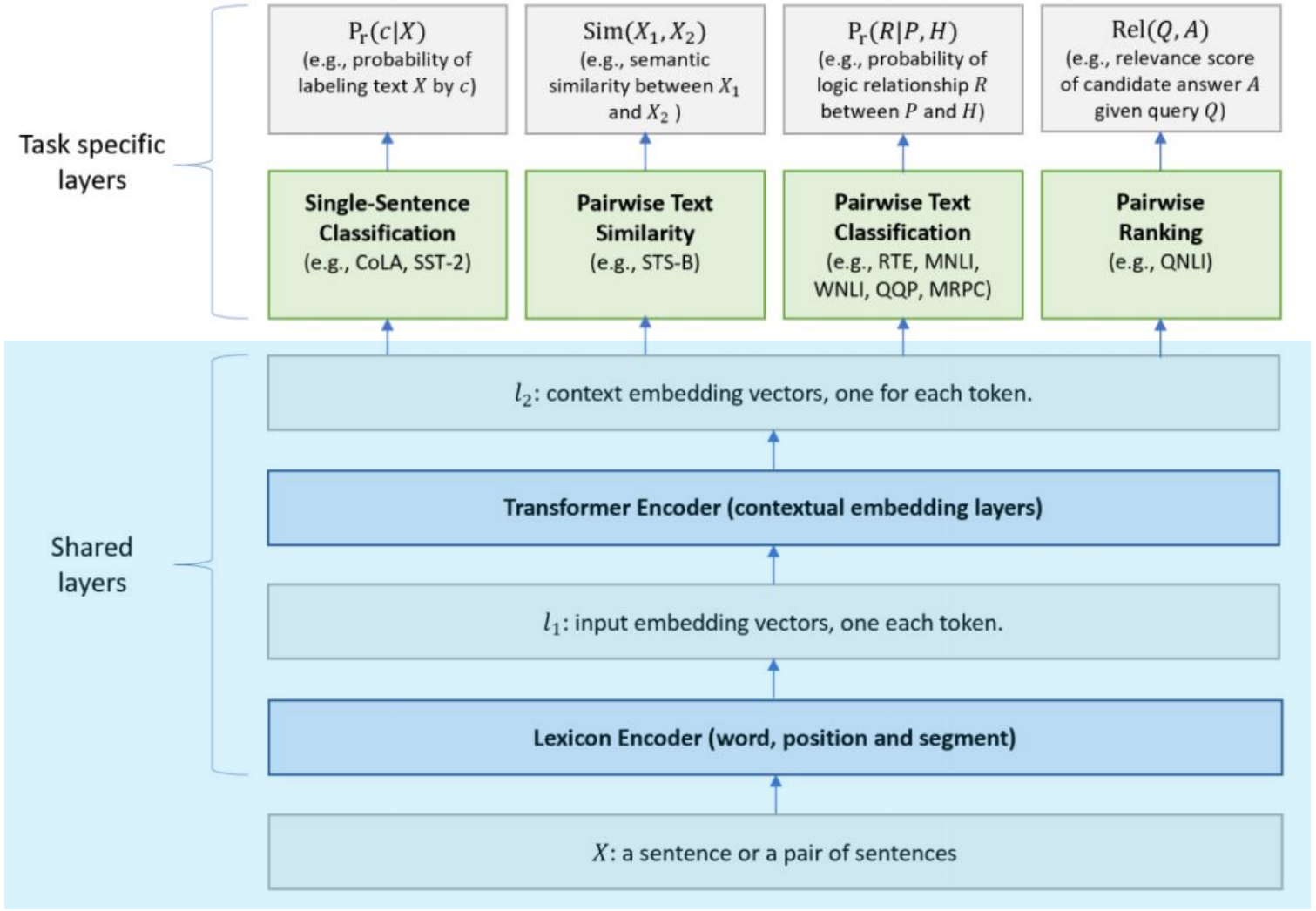
3. The Proposed MT-DNN Model

MT-DNN model의 구조



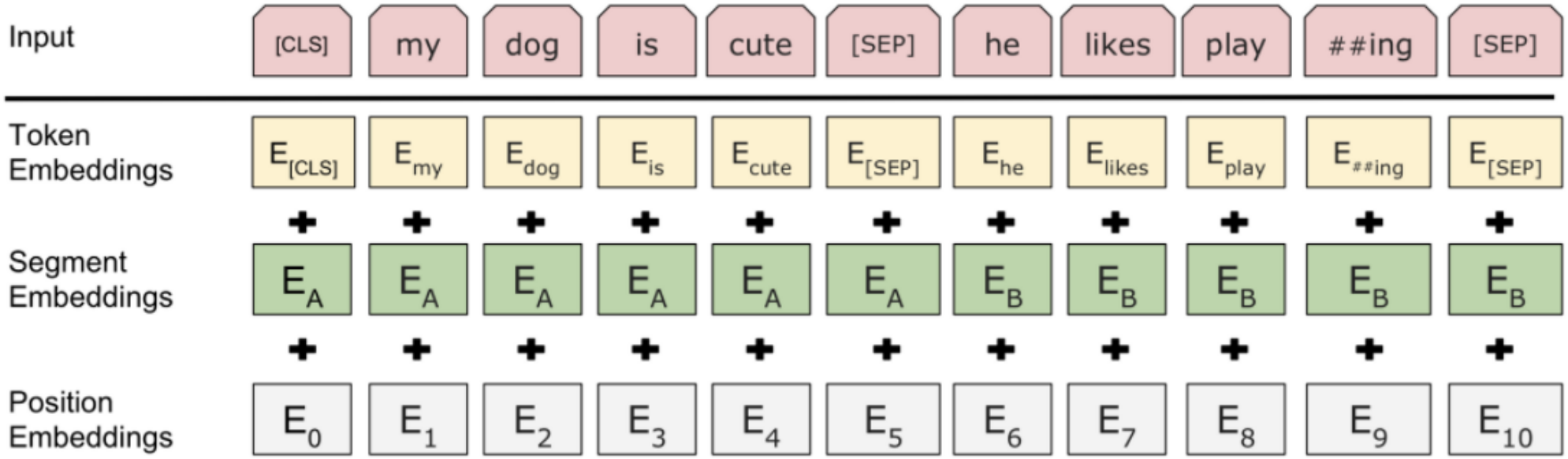
3. The Proposed MT-DNN Model

MT-DNN model의 구조



3. The Proposed MT-DNN Model

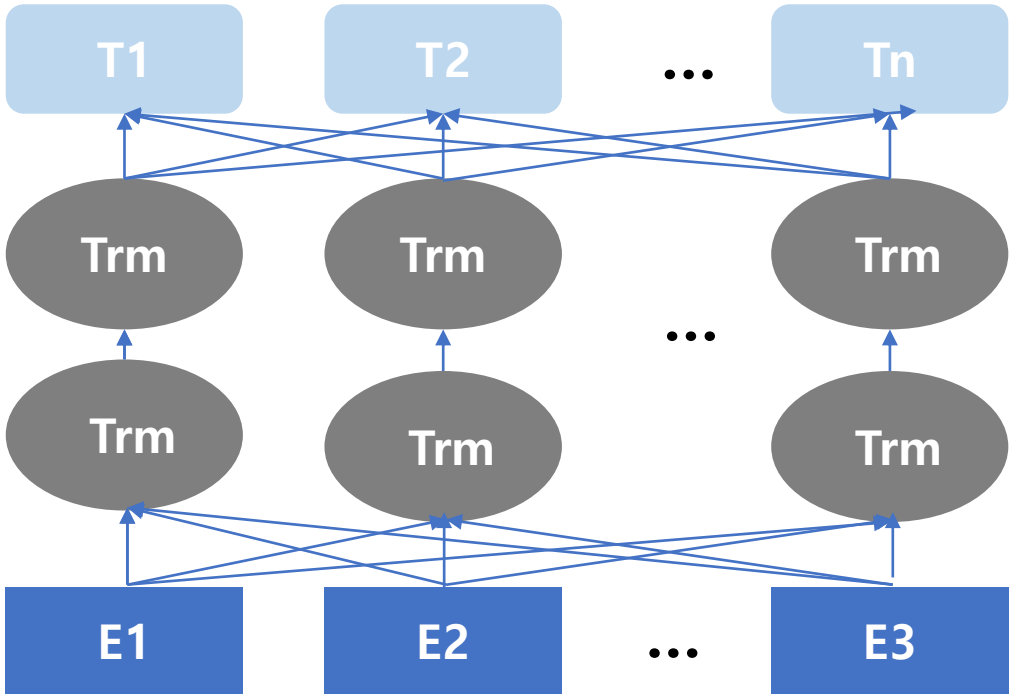
1. Shared layer - Lexicon Encoder (l1)



- BERT의 임베딩 방식과 동일
- X = {x1, ..., xm} sequence가 input으로 들어오면 Embedding vector로 변환
- token / segment / position embedding

3. The Proposed MT-DNN Model

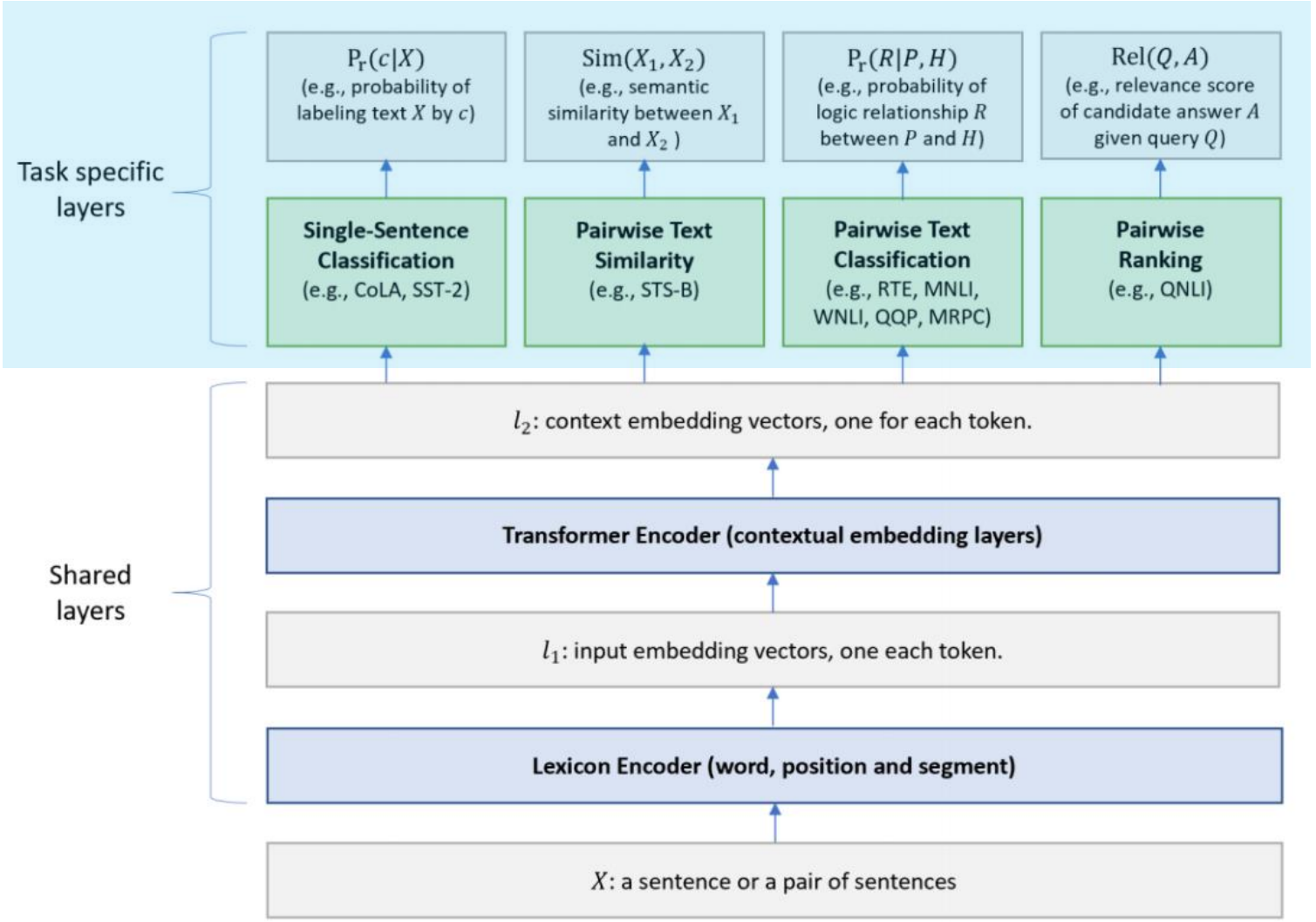
1. Shared layer - Transformer Encoder(I2)



- Lexicon Encoder (I1) 의 임베딩 벡터를 입력으로 받아서 multilayer bidirectional Transformer encoder 사용하여 각 Token의 Output Vector 추출
- Transformer → 생성된 vector는 self attention을 통해 주변 token 정보를 반영한 contextual embedding vector

3. The Proposed MT-DNN Model

MT-DNN model의 구조



3. The Proposed MT-DNN Model

2. Task specific layer

(1) Single-Sentence Classification Output

- Transformer Encoder (l2) 로부터 얻은 output vector의 [CLS] 토큰
  : contextual embedding 을 진행한 벡터
- 하나의 문장을 분류하는 Classification Output은 다음과 같이 계산

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

class

Input sentence

Task specific parameter

Class token

: [CLS] Token과 Task Specific Parameter의 곱에 Softmax를 취하여 각 Input Sentence X 가 각 class에 속할 확률 계산

3. The Proposed MT-DNN Model

2. Task specific layer

(2) Text Similarity Output

- Transformer Encoder (l2) 로부터 얻은 output vector의 [CLS] 토큰
  : 문장쌍 (X1,X2)의 semantic representation
- Sentence pair로 이루어진 (X1, X2)의 유사도는 다음과 같이 계산

$$\mathrm{Sim}(X_1, X_2) = \mathbf{g}(\mathbf{w}_{STS}^{\top} \cdot \mathbf{x})$$

sigmoid

Task specific parameter

Class token

3. The Proposed MT-DNN Model

2. Task specific layer

(3) Pairwise Text Classification Output

◆ stochastic answer network (SAN)를 활용하여 classification

- 핵심 IDEA : Multi -Step reasoning
- RNN을 이용하여 K-step Reasoning을 하는 것이 특징

각 K step 마다,

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

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

3. The Proposed MT-DNN Model

2. Task specific layer

(4) Relevance Ranking Output

- Sigmoid function을 통해 모든 answer candidate 문장을 scoring하고 ranking을 통해 1개의 문장만을 True로 분류

$$\mathrm{Rel}(Q, A) = g(\mathbf{w}_{QNLI}^{\top} \cdot \mathbf{x})$$

Input Sentence Pair

sigmoid

Task specific parameter

Class token

3.1 The Training Procedure

- 9개의 GLUE Dataset 을 모아 Training dataset을 구성
- Batch 단위로 training할 때 마다 shared layer 와 해당 task의 specific layer 학습

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

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

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

Answer가 포함된 문장

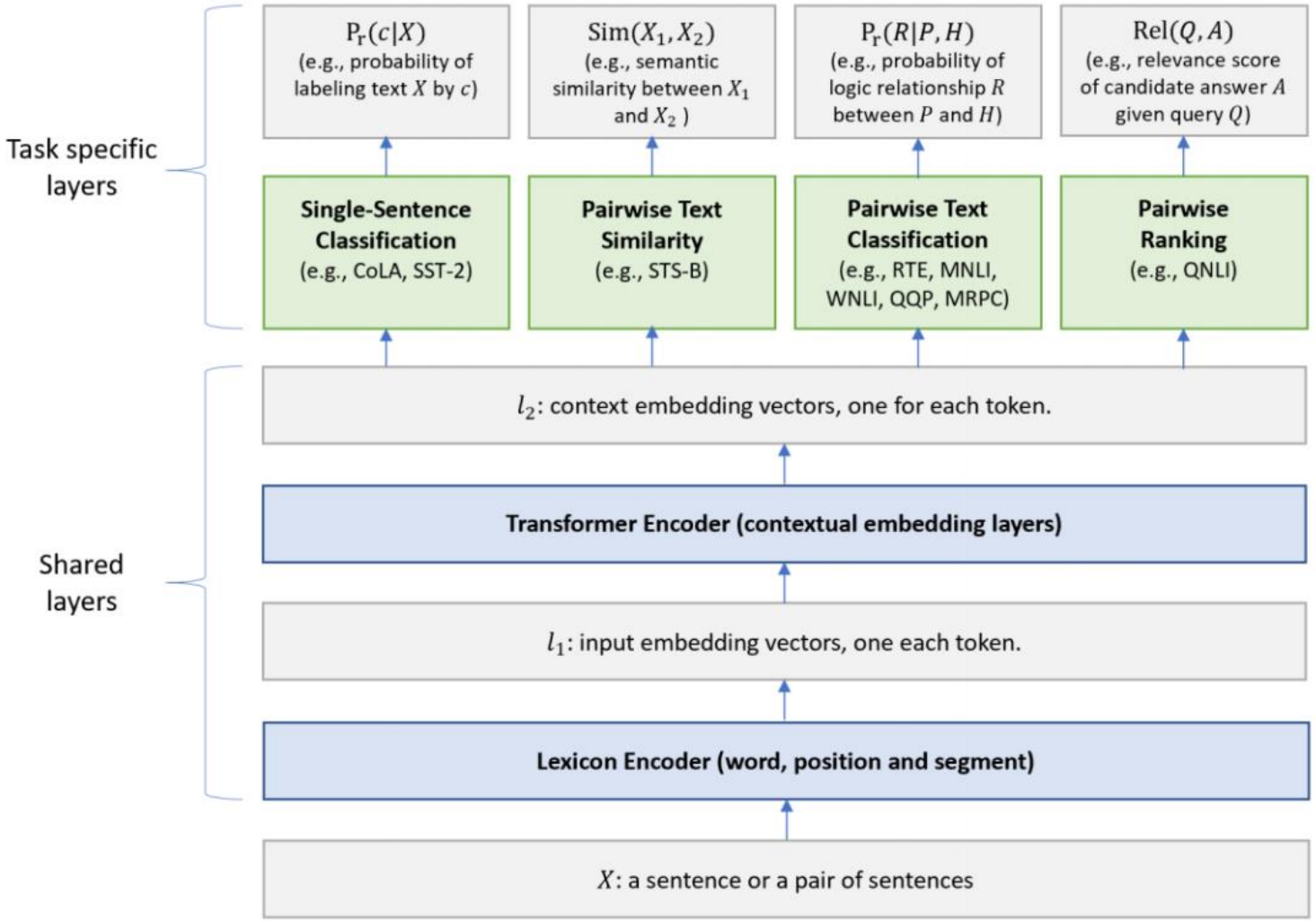
$$P_r(A^+|Q) = \frac{\exp(\gamma \text{Rel}(Q, A^+))}{\sum_{A' \in \mathcal{A}} \exp(\gamma \text{Rel}(Q, A'))}, \quad (9)$$

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

3.1 The Training Procedure

MT-DNN model의 구조



4. Evaluation – GLUE test set

- BERT (80.5) → MT-DNN (82.7) 2.2% 성능 향상

  →데이터가 적을 때 더 높은 성능 향상

Single Sentence Classification | Text Similarity | Pairwise Text Classification | Relevance Ranking

Model	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	Score
BiLSTM+ELMo+Attn [1]	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4/76.1	-	56.8	65.1	26.5	70.5
Singletask Pretrain Transformer [2]	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	-	56.0	53.4	29.8	72.8
GPT on STILTs [3]	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.8/80.6	-	69.1	65.1	29.4	76.9
$BERT^{4}_{LARGE}$	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
$MT\text{-}DNN_{no\text{-}fine\text{-}tune}$	58.9	94.6	**90.1/86.4**	89.5/88.8	**72.7/89.6**	86.5/85.8	**93.1**	79.1	65.1	39.4	81.7
MT-DNN	**62.5**	**95.6**	**91.1/88.2**	**89.5/88.8**	**72.7/89.6**	**86.7/86.0**	**93.1**	**81.4**	65.1	**40.3**	**82.7**
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1

Sementically equivalent problem

NLI problem

4. Evaluation – GLUE dev set

◆ ST-DNN :

Multi Task Learning을 하지 않고 Task specific layer만 바꾸어서 (SAN, Pairwise Ranking Loss) 학습한 모델

- SAN 을 사용하여 얻은 성능 향상 효과는 그다지 크지 않음
- QNLI는 Ranking approach로 큰 성능 향상

SAN

Relevance Ranking Loss

Model	MNLI-m/mm	QQP	RTE	QNLI (v1/v2)	MRPC	CoLa	SST-2	STS-B
$BERT_{LARGE}$	86.3/86.2	91.1/88.0	71.1	90.5/92.4	89.5/85.8	61.8	93.5	89.6/89.3
ST-DNN	86.6/86.3	91.3/88.4	72.0	96.1/-	89.7/86.4	-	-	-
MT-DNN	**87.1/86.7**	**91.9/89.2**	**83.4**	**97.4/92.9**	**91.0/87.5**	**63.5**	**94.3**	**90.7/90.6**

4. Evaluation – Domain Adaptation

◆ Procedure

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 만듦

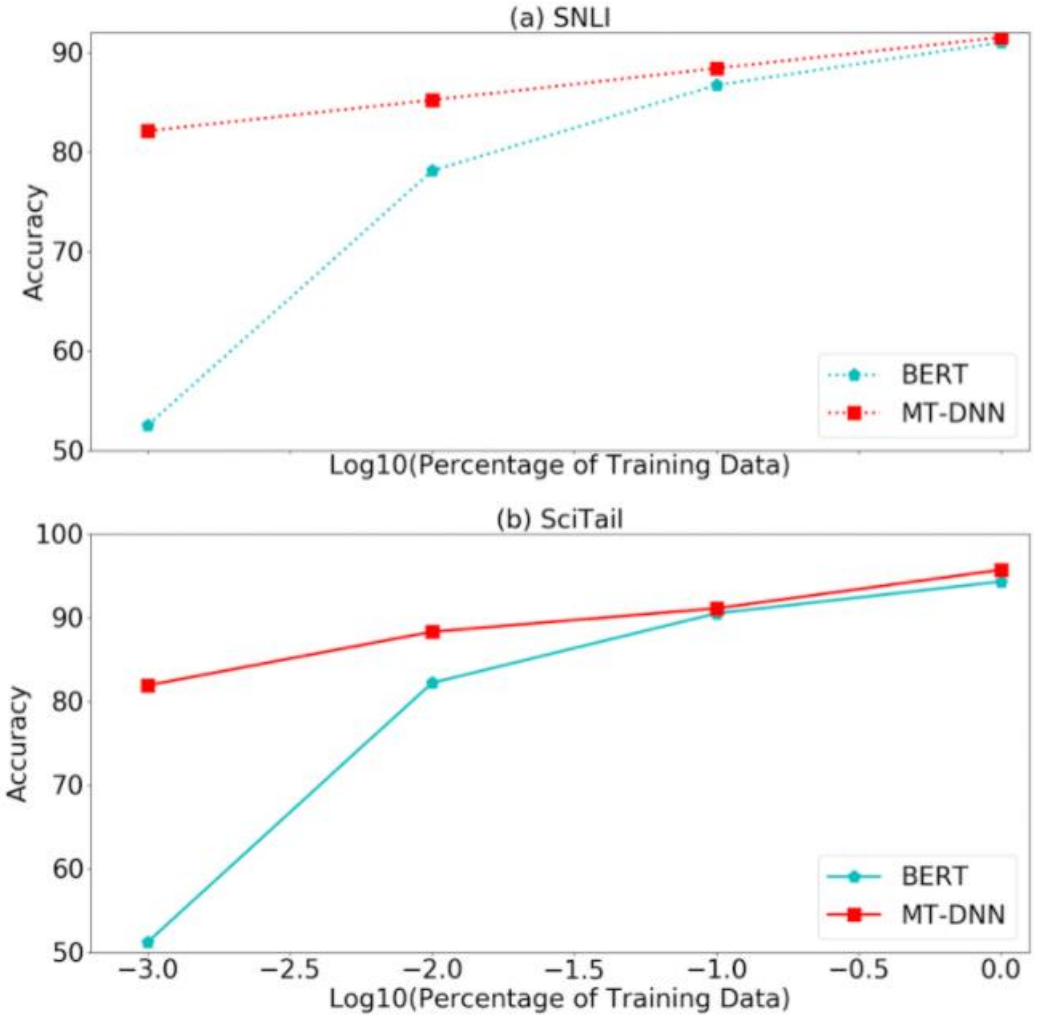
3.task-specific test data로 평가

◆ Result

•더 적게 train할수록 MT-DNN 더 성능 발달

→데이터가 적을 때 MT-DNN의 성능이 훨씬 좋음

•BERT보다 domain adaptation에 일관되게 효과적임



5. Conclusion

◆ MT-DNN

- BERT에 Multi-task learning을 수행하여 성능 개선
- SNLI, SciTail, GLUE에 대해 10개 NLU task에서 SOTA의 결과를 냄
- Domain adatation 실험에서 일관되게 좋은 결과

MT - DNN

Multi-Task Deep Neural Networks for Natural Language Understanding

NLP 2조