**Multi-Task Deep Neural Networks for Natural Language (acl 2019)**

**Main Idea**

* BERT에 다양한 NLU Tasks의 Multi-Task Learning 진행 => 기존 BERT 능가
* 그 후, 특정 Downstream Task에 대해 Fine-Tuning 할 때, 단순 Linear Layer가 아닌 다른 구조(Stochastic Answer Network)를 이용 => 성능 향상
* Multi-Task Learning을 하면 Out of Distribution? 아니다. Domain Adaptation 실험을 통해 이 방식이 OOD 문제를 없애며 general representation을 더 증가시킬 수 있음
* QNLI를 Binary Classification이 아닌 Relevance Ranking으로 학습 => 더 좋은 결과(뒤에 나옴)

질문: decaNLP의 데이터셋이랑은 다른 가?

**Task**

GLUE에 있는 Tasks를 이용하여 실험 진행, 크게 4가지로 분류(분류에 따라 Fine-Tuning Layer 구조가 달라진다.)

1. Single-Sentence Classification: 하나의 문장에 대해 Label 중 한 개로 분류  
   문법적으로 타당한지 예측: CoLA  
   리뷰가 긍정인지 부정인지: SST-2
2. Text Similarity: 주어진 문장 쌍이 의미적으로 얼마나 유사한지 Regression 예측  
   STS-B
3. Pairwise Text Classification: 주어진 문장 쌍의 관계를 Label 중 한 개로 분류  
   RTE, MNLI: entailment, contradiction, neutral 중 한 개로 NLI Tasks  
   QQP, MRPC: paraphrasing dataset, 의미적으로 얼마나 동일한지 Classification 예측
4. Relevance Ranking: 쿼리와 보기들이 있을 때 보기의 연관 정도에 따라 Ranking을 매기는 Task  
   QNLI: 각 보기에 Binary 값을 부여 -> Relevance Ranking을 도입해 각 쿼리-보기 pair에 확률을 부여 => 더 좋은 성능

질문: 내 생각에는 그러면 Shared Layers 부분은 완전 더 학습이고 Task Specific Layer는 기존 BERT와 비슷하게 학습될 듯(만약 GLUE말고 다른 Task에 적용하고 싶다면 Shared Layers만 떼어서 사용??) = 근데 당연한 거 아닌가? 당연히 더 많은 데이터셋으로 학습하니 그렇겠지(기존 BERT는 1개의 Task로만 Fine-Tuning 했다..?)

**Model**

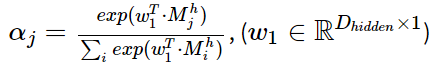
**Shared Layers**

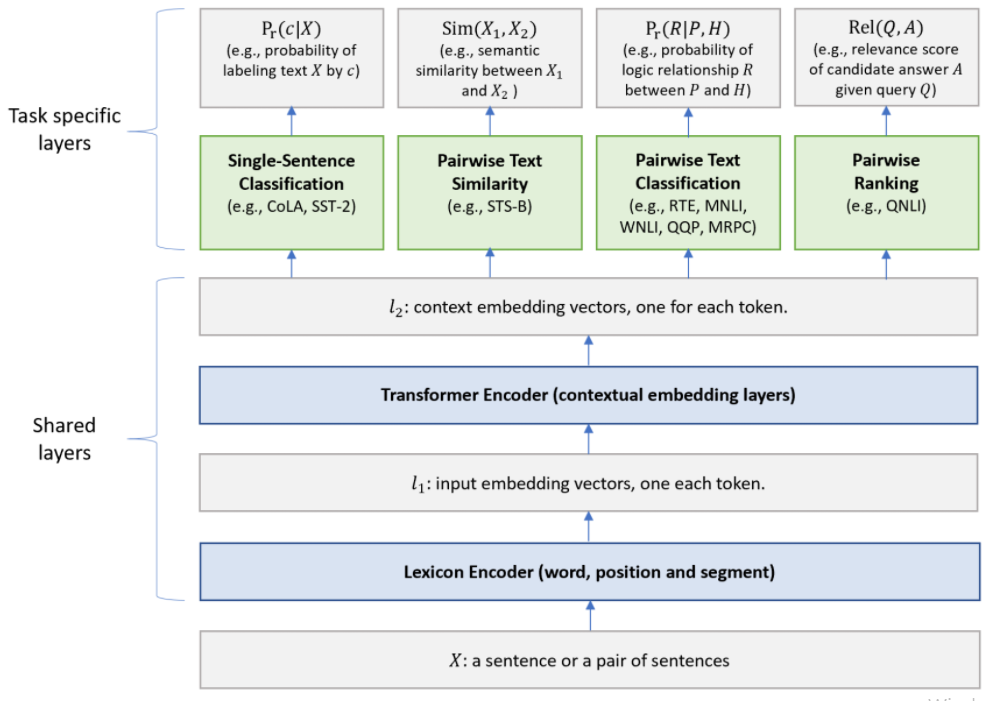
BERT와 동일한 설정을 유지

* [CLS], 문장(Segment) 경계에는 [SEP]
* 역시 BERT와 동일한 임베딩
* 역시 BERT와 동일한 인코딩

**Task Specific Layers**

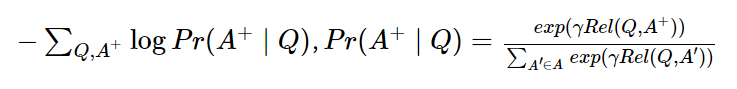
GLUE로 Multi Fine-Tuning 후에 Testing 하지만 다른 NLP Task(Generation)도 대상이 될 수있다.

* Single Sentence Classification: [CLS] 토큰 representation의 위치에서 각 클래스들의 확률 추출  
  
* Text Similarity Output: 역시 [CLS] 토큰 representation의 위치에서 하나의 점수 추출  
    
  
* Pairwise Text Classification Output: 기존 BERT와 다르게 Stochastic Answer Net(SAN)을 Linear Layer + softmax 대신 사용. 기존 NLI의 SOTA 모듈로 문장들에 대한 Multi-step Reasoning을 모델링  
    
  SAN은 주어진 문장 쌍을 GRU에 K번 반복하여 입력해 정제된 representation을 얻어 최종 예측을 진행한다.  
    
  1. [CLS] + Hypothesis(문장1) + [SEP] + Premise(문장2) + [SEP] => 트랜스포머 인코딩 => Contextual 임베딩  
    
    
  2. 그 중 H와 P에 속하는 토큰들끼리 따로 모음  
      
    
  3. M-h를 이용하여 GRU 초기 hidden state(s0, seq2seq의 context vector 같은 건가), M-p를 이용하여 t번째 입력(x-t)을 만듬  
    
  4. s0 만들기: 가중합(트랜스포머가 아닌 또 다른 self-attention)을 이용  
    
    
    
  5. t번째 input(x-t) 만들기: GRU의 t-1번째 hidden state(st-1) & M-p를 이용해 t번째 input 만듬(0번째는 s0 이용)  
  여기서 st-1과 M-p를 통해 가중치가 한번 더 만들어진다(nonself-attention) => 가중합 input x-t 생성  
    
    
  6. 이 짓을 대신에 사전에 정의된 K-step 만큼 진행 (k = t의 값) ..? 이해못함  
    
  SAN에 대해서는 다시보기
* Relevance Ranking Output: Q-A의 쌍들의 relevance score 계산  
    
  질문: 각 쌍을 1번씩 넣으면 각 쌍에 대해 0~1로 나올 텐데 이거를 다시 scaling 해주는 것인가?



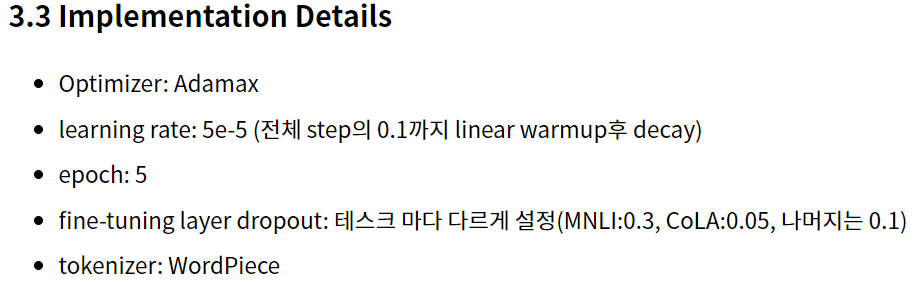
**Training**

**Objective**

* Classification Objective: Single Sentence Classification, Pairwise Classification -> cross-entropy
* Regression Objective: Text Similarity 등의 regression task => mean-squared error
* Relevance Ranking Objective: QNLI 등등  
    
  감마는 튜닝 요소이며 데이터에 따라 결정, 여기서는 1을 이용, 쿼리-정답 pair의 negative log likelihood를 최소화!

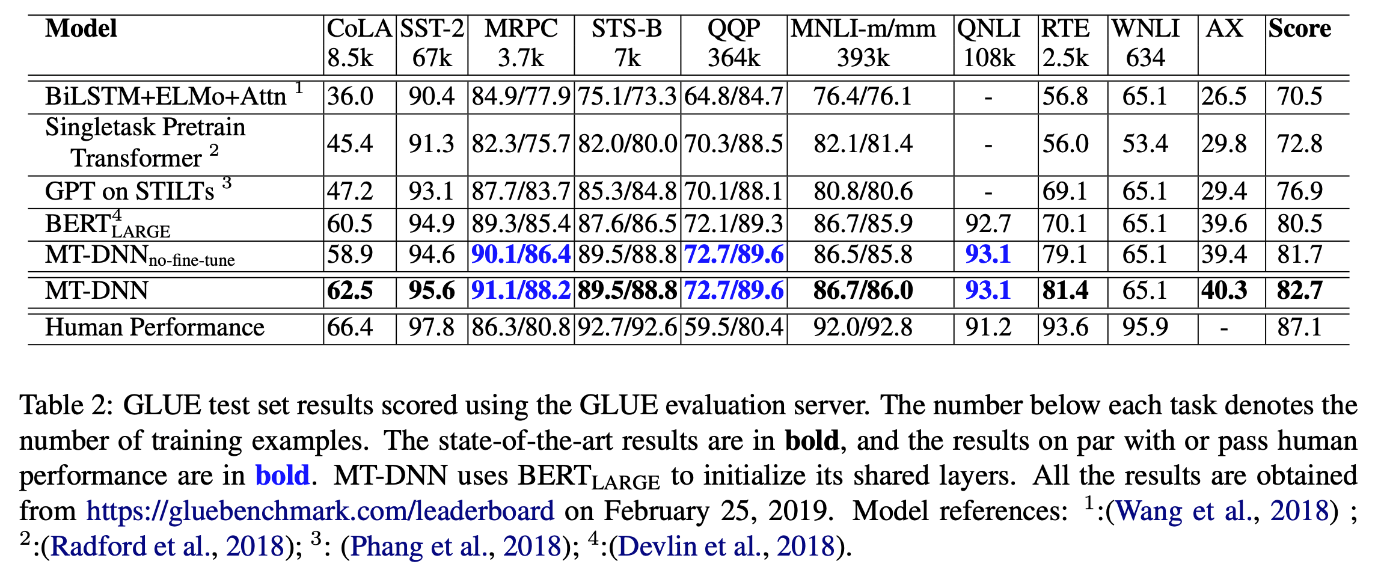
**Training Procedure**

Pre-Training -> Multi-Task Fine-Tuning(모든 Task 하나로 묶되 하나의 배치에는 하나의 Task로 만, 어떤 Task인지도 알면서) -> Specific Task Fine-Tuning



**Experiments**

**GLUE**

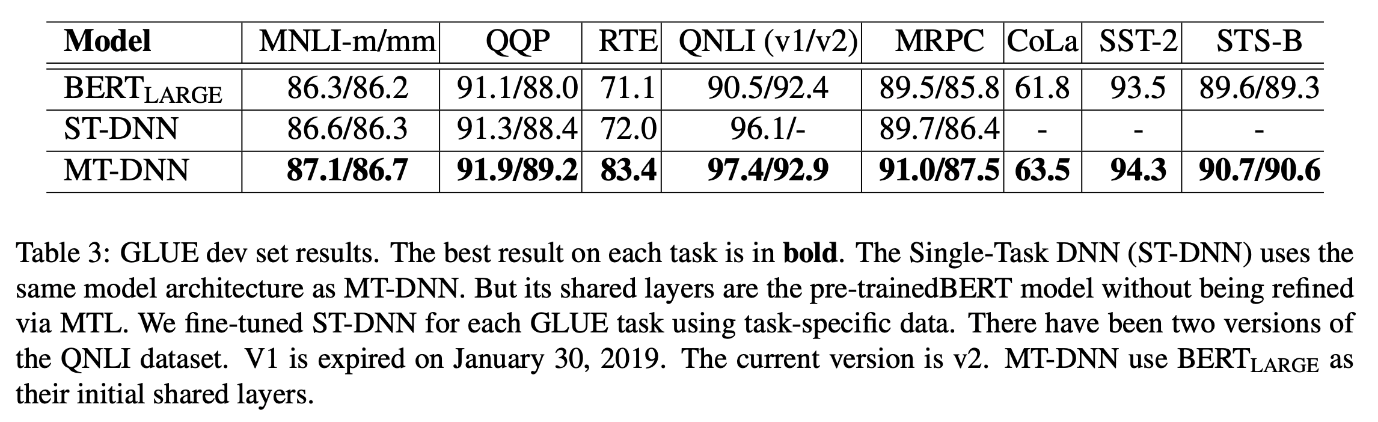


no-fine-tune: 학습 과정에서 specific Fine-Tuning은 진행 안함 = 1 개는 기존 BERT보다 성능이 떨어짐(CoLA서 그럼, CoLA와 유사한 Task가 없기 때문 = 유사한 데이터셋 적음)

그것까지 진행했을 때는 모든 Task에서 성능 증가 => 다른 점이 Multi-Task Learning이기 때문에 이 효과라는 것을 명확히 알 수 있음

또한, 적은 양의 Task의 경우 비슷한 많은 양의 Task의 Multi-Task Fine-Tuning의 도움을 받아 성능이 더 크게 증가

Specific Fine-Tuning을 했을 때 CoLA의 성능이 증가되었다 => Multi-Task Learning 후 + Specific Fine-Tuning을 한다면 여러 Task에 대한 Domain Adaptation을 높여준다라고 볼 수 있다.



그리고 Multi-Task Learning 빼고 Single-Task Learning + SAN vs 기존 BERT 비교 했을 때  
Single Task Learning + Relevance Ranking vs 기존 BERT 비교 했을 때 전자가 더 성능 증가

**Domain Adaptation**

다른 Domain에 얼마나 더 Overall 하게 성능을 잘 보이는지

앞서 CoLA를 예시로 들었지만 완전 New Domain은 아니였다 이번에는 완전 New Domain인 SNLI와 SciTail 으로 Specific Fine-Tuning 했을 때(GLUE에 대해 MT-DNN vs BERT 둘 다 진행했을 때, BERT는 아마 Single을 각 Task에 대해 Fine-Tuning 하는 듯?) MT-DNN이 우세

