GPT-1

Improving Language Understanding by Generative Pre-Training

▼ Abstract

Problem : unlabeled text data는 풍부하지만 labeled text data는 부족해 모델이 적절한 작업 수행이 어려움..
→ Solution : Unlabeled 말뭉치들을 generative pre training 시키고 각 task에 맞게 fine tuning
→ fine tuning 단계에서 task-aware input transformations를 사용함으로써 모델 구조를 최소한으로 변경시키고 효과적으로 transfer 시킴

▼ Introduction

💡 Unlabeled : 시간, 비용적인 면에서 labeled 대안적, 지도학습 가능한 경우에도 더 좋은 성능향상 제공
but, word-level information 이상의 정보 활용 어려움
∵ 1. 어떤 종류의 optimization objectives가 transfer에 유용한 텍스트 표현을 학습하는데 효과적인지 모름 2. 학습된 representations를 r각각의 타겟 task에 transfer하는 효과적인 방법에 대한 합의가 없음
→ 언어 처리를 위한 효과적인 준지도 학습 개발이 어렵!

그래서 우리는 **Semi-supervised** = Unsupervised Pre-training + Supervised fine-tuning를 개발! → 보편적인 representations를 학습하는 것이 목표!
stage1. 신경망 모델의 초기 파라미터를 학습하기 위해 unlabeled data를 사용?
stage2. 파라미터들을 supervised objective를 이용해 타겟 task에 적용
- Transformer 사용→ long-term dependencies를 처리하기 위한 구조화된 메모리 사용

4가지 language understandings tasks에 대해 평가했는데 성능 good!

▼ **Framework**

1. large corpus of text로 언어모델을 학습 2. labeled data로 fine tuning

1. Unsupervised pre-training

* unsupervised corpus에 대해 window size k개 token이 주어졌을때, 다음 token을 예측
* 다음 token에서 등장할 likelihood를 최대화하도록 학습

- ~~i=1부터 모든 시퀀스에 대해 바로 전 단계에서 부터, k번째 이전까지 살펴본 다음, i번째 해당하는 토큰 또는 단어가 무엇인지에 대한 likelihood를 가장 최대화하는ㄴ~~

$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$

→ U: 전체 unsupervised text corpus

- **multi layer Transformer decoder** 이용

(→ transformer는 인코딩과정과 디코딩 과정을 둘 다 가지는데 gpt에서는 디코더만 사용!)

디코더를 여러개 쌓음

: input context tokens에 multi headed self attention을 적용한 후, position wise feedforward layers를 거쳐 타겟 token들의 확률분포를 output으로 얻음

2. Supervised fine-tuning

사용된 파라미터들을 타겟 task에 맞게 fine-tuning

* labeled dataset을 pre-trained model에 통과시켜 hlm을 얻고, Wy를 갖는 linear layer에 hlm을 통과시켜 softmax probability를 계산 (x의 input sequence가 주어졌을때 y를 예측하는 확률)

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

위와 같은 목적함수를 최대화하는 방향으로 Wy를 업데이트

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

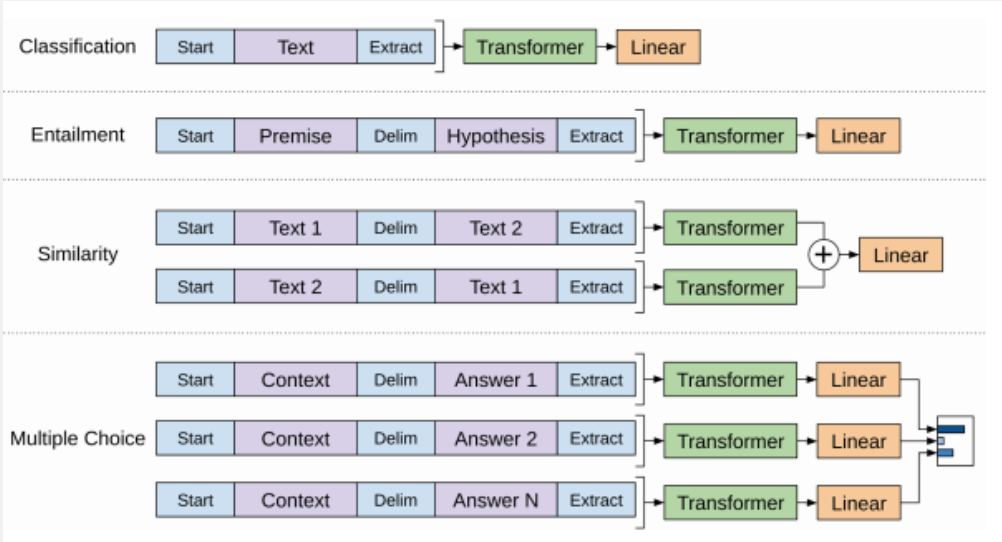
: auxiliary objective (보조 목적함수) = 람다를 이용해 L1의 반영정도를 조절

C : supervised learning text corpus

(L1: labeled dataset C에 대한 L1)

→ unsupervised learning 에 해당하는 목적함수 L2 뿐만아니라 supervised learning 용 L1도 함께 update!
위와 같이 L1을 fine tuning에 포함시키면 generalization ↑, convergence ↑

3. Task-specific input transformations
pretrained model을 통해 generative 하게 만들었지만 각 target task마다 변형이 필요함! 구조에 많은 변화를 주는 건 아니고 각 task 마다 적합한 input 데이터 구조로 변형만 주면 됨!



- Classification - 기존 방법 그대로 fine-tuning
- Textual Entailment - 전제(premise)를 통해 가설(hypothesis)의 참, 거짓을 밝히는 task이므로 delimeter로 나누어진 premise와 hypothesis를 concat하여 fine-tuning
- Similarity - 두 문장의 순서가 결과에 관계없음! 가능한 두가지 순서 경우로 입력 시퀀스를 나누어 input하고 독립적으로 얻은 결과 hlm 두개를 element-wise addition 해서 fine-tuning
- Question Answering and Commonsense Reasoning - context와 각각의 answer들에 대한 조합들을 입력으로 함. 각각 독립적으로 디코더

에 태운후 나오는 output hidden state vector 각각을 구한 후, linear layer에 투입하고, softmax 함수로 계산함

→ 정규화 된 답변들의 distribution이 출력됨

▼ Experiments

💡 **1. Setup**

*** Unsupervised pre-training**

- LM 학습을 위해 BooksCorpus 데이터셋 활용: 길고 연속적인 텍스트가 존재하여 long-range information을 학습할 수 있음
- Alternative dataset: 1B word Benchmark: 같은 길이지만 long-range structure가 파괴되어 있음

(생략)

2. Supervised fine-tuning

*** Natural Language Inference :** 짝지어진 문장을 읽고 모순/ 중립/ 연관으로 문장 사이 관계를 판단하는 것

MNLI: 1.5% , SciTail : 5%, QNLI: 5.8% , SNLI : 0.6% ,

→ RTE를 제외한 나머지 데이터 셋에서 유의미한 성능향상, 다수의 문장에 대해 더 나은 추론을 하고 언어적 모호성을 잘 처리하는 것으로 보임

*** QA :**

Story Cloze Test : 8.9% , RACE : 5.7% 성능향상

→ long range contexts를 효과적으로 처리

*** Semantic Similarity** : 주어진 두 문장의 의미론적 유사 정도 판단

QQP : BilSTM+ELMo+Attn 보다 4.2% 성능향상

→ rephrasing, 부정문 이해, syntactic 모호성 처리 가능

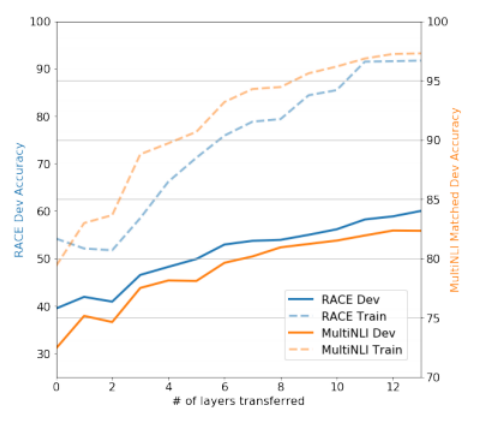
*** Classification:**

CoLA(문법적으로 맞았는지 틀렸는지 분류) : 35.0 → 45.4 , SST-2(문장 sentiment 분류) : 91.3%, GLUE: 68.9 → 72.8

❗12개 데이터셋에서 9개가 SOTA(현재 최고 수준) 달성. STS-B(5.7k)와 같은 가장 작은 데이터셋에서 SNLI(550k)와 같은 가장 큰 데이터셋에 이르기까지 잘 작동한다

▼ Analysis

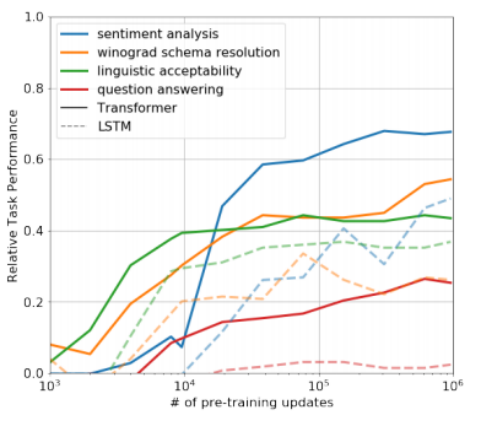
💡 📌 **Impact of number of layers transferred**
- 디코딩 블록 수



RACE와 MultiNLI에서 transformer layers 수에 따른 성능 비교

→ layer 수가 증가함에 따라 성능이 향상됨 but,12개 정도에선 수렴

📌 **Zero-shot Behaviors**



점선(LSTM 사용) , 실선(Transformer 사용)

→ 1. fine-tuning을 하지 않았을 때, pre-train 시킨 정도에 따라 대부분의 task에서 성능이 향상 = pre-train 과정이 다운스트림 task에 직접적인 도움을 줌

2. 모든 task에 대해 LSTM보다 Transformer를 사용했을때 성능이 향상되었음을 알 수 있음

📌 **Abelation studies**

1. without the auxiliary LM objective during fine-tunig → NLI와 QQP와 같은 큰 데이터셋에서 auxiliary LM 목적함수가 효과적임을 확인

2. Transformer 효과 → transformer 대신 LSTM을 사용했을때 Avg.Score 5.6점 하락
3. without pre-training → full model에 비해 성능 14.8% 하락

▼ **Conclusion**

generative pre-training & discriminative fine-tuning를 통한 frame work를 소개함
- pre-training을 통해 world knowledge와 long-range dependencies를 다룰 수 있게 됨
- QA, Semantic similarity, entailment determination, 텍스트 분류와 같은 작업들에 성공적
- 12개 dataset 中 9개 SOTA
- 비지도 사전학습을 이용한 특화된 task에 대한 성능을 올리는게 오랜 머신러닝 연구의 목표였는데 우리가 해냄!
- 우리가 어떤 모델(Transformers)와 어떤 데이터셋(긴 범위에 대한 의존성이 존재하는 텍스트)이 가장 좋은지에 대해 힌트도 줌!