Improving Language Understanding by Generative Pre-Training – 2018, OpenAI

Abstarct

Textual entailment, question answering, semantic similarity assessment, and document classification 등의 다양한 주제들 포함.

Unlabeled text corpora가 많을 지라도 이러한 작업들을 위한 labeled data는 여전히 부족.

Unlabeled text를 이용해 generative pre-training에 의해 이런 작업들 활성화. (각 task에 discriminative fine tuning 적용)

기존 모델 아키텍쳐에 조금의 변화만 요구하면서 task-aware한 transfer를 fine-tuning 동안에 진행한다.

많은 benchmark(dataset)에 대해 증명.

각 task에 전문적인 모델보다 성능이 좋다. 12개 중 9개에서 더 뛰어나며 나머지 3개는 봐줄만하다. Ex) commonsense reasoning에서 8.9%(Stories Cloze Test Dataset), question answering에서 5.7%(RACE Dataset), textual entailment에서 1.5%(MultiNLI)

1. Introduction

Raw-text(no label)로부터 학습하는 것은 supervised 학습의 의존을 완화하는데 중요. 많은 labeled data를 요구하는 것은 여러 domain에(task에) applicability가 떨어짐.

심지어 supervised 보다 많은 양의 unlabeled data representation 학습이 성능이 좋을 때도 있다. 그 증거는 word embeddings에서 많이 보여주었다.(다른 논문들에서)

하지만 unlabeled text에서 단어 단위 이상의 정보를 이용하는 것은 어렵다.

1) text representations 학습에서 transfer를 위한 objective의 종류(task의 종류?)를 어떻게 세워야할지 애매하다.???

2) 뭐 task 종류는 어찌어찌 정했다 해도 target tast에 learned representation을 transfer하는 것이 합치기 어렵다.(no consensus)

본 논문에서는 unsupervised pre-training + supervised fine-tuning의 조합의 semi-supervised를 다양한 task에 대해 적용, 즉 representation learning한 것을 다양한 task에 적용.

심지어, target task의 dataset의 domain이 unlabeled의 corpus와 동일하지 않아도 된다. 그래도 잘 동작할 것이다. (비슷하면 더 좋고, 하지만 요구하지 않는다.)

간단한 두 개의 전략: 1. Unlabeled data로 학습, 2. Adapt!

Machine translation, document generation, syntactic parsing등의 다양한 task에서 강렬한 Transformer 아키텍쳐 이용. => a more structured memory for handling long-term dependencies (??이게 뭘까)을 제공해준다.

또한 각 task에 적용할 때 조금은 각 task에 맞게 변화 시키는데 그 중 한 개가 task-specific input adaptation(traversal- style) = text input을 한 개의 token들(단어 or 문자 한 개)의 연속으로 변화시킴 => 약간의 변화로 fine-tuning 잘하게 한다.

Natural language inference, question answering, semantic similarity, text classification에 대해 성능 평가를 한다. Our general task-agnostic model은 각 task에 전문적으로 차별된 model들 보다 잘 작동한다. + 위에 말한 benchmark에 GLUE multi-task benchmark에서 5.5%까지.

Pre trained model의 Zero-shot behaviors(??이게 뭘까)를 분석했고(4가지 task에 대한 다른 setting에서) 성능이 있음을 발견

2. Related Work

Semi-supervised Learning for NLP – word-level, phrase-level 통계 단위 계산에 unlabeled data를 사용하기도 했고 여기서 피처로 동작해 supervised model에 사용되었다.

그 이후로 unlabeled corpora(말뭉치)들을 통해 학습된 word embeddings의 장점을 증명했고 많은 task에서 이 embedding이 잘 작동하였다. 하지만 word-level information 정도로만 transfer 하고 반면에 여기서는 더 높은 higher-level의 정보의 semantic(의미)를 추출해보려 한다.

Phrase or senetence – level 등의 embedding이 본 논문 외에도 많이 연구되었고 다양한 task들에 대해 작동하였다.(vector representations)

Unsupervised pre-training – supervised learning의 objective를 수정해서 하는 것보다. 처음 initialization의 transfer를 위해 많이 사용. Image classification이나 regression tasks등에서 많이 사용되었다. 심지어 pre-training은 regularization 역할도 했다.(test시 일반화) => NLP 뿐만 아니라 다양한 곳에서 사용된다.

이 논문의 closest line은 역시 pretrain 후 finetuning인데 이전에서는 모델을 transformer 안씀 => longer-range linguistic structure 사용 불가.

앞에 이어서 natural language inference, paraphrase detection, story completion에 사용.

다른 논문에서는 translation model의 hidden representations을 사용했다(auxiliary feature로, 보조 피쳐) => 하지만 target task에 대해 상당한 새로운 파라미터를 적용하는 과정 필요 => 반면에 본 논문은 조금의 변화만을 가져온다.

Auxiliary training objectives ….?이게 뭘까

3. Framework

Training procedure은 2 단계

1) 거대한 text의 corpus에서 high-capacity language model을 학습시키는 것

2) fine-tuning 단계 => discriminative task에 labeled data와 함께 adapt 시키기

3.1. Unsupervised pre-training

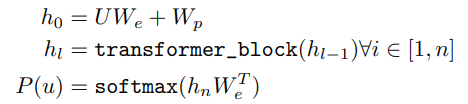


위 식은 likelihood로 이 것을 maximize하는 것이 목표이다.

Token들(단어들)에 대한 corpus가 주어졌을 때(모든 단어들의 뭉치?) 

K는 context window size?(n-gram처럼??)이고 P는 조건부확률인데 파라미터 세타를 이용. 또한 stochastic gradient descent로 학습된다.

디코더로는 역시 multi-layer transformer를 이용(레이어의 수는 정하기 마련). 이 모델은 multi-headed self-attention 과정이 있다.



U = (u-k, …. u-1)의 토큰들의 context vector, n은 레이어의 수, We는 token embedding matrix, Wp는 position embedding matrix…

Transformer 다시 보고 오기

3.2. supervised fine-tuning

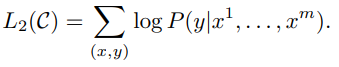
Likelihood maximize 하고나서 labeled dataset C를 이용한다. (x1,…,xm 의 tokens의 sequence와 label y를 가진)

input들은 앞서 pre-train model을 통해 final transformer block’s activation 값을 얻게 된다. 이것은 이제 추가적으로 fine-tuning하게 될 linear output layer로 feed 된다.



이때 Wy는 y를 예측하기 위한 파라미터이다.

따라서 두번째 likelihood(maximization 필요한)가 발생한다.



앞서 일반 linear output layer를 붙였지만 보조의 task들에 대해서 적용했을 때

1) supervised model의 regularization(generalization)을 높였고 convergence를 가속화 시켰다(빨리)



이것은 C도 transformer가 사용하긴 하니까 생기는 fine-tuning시 생기는 최종 loss라고 보면 된다.

따라서 오직 Wy의 파라미터와 embeddings만 fine-tuning시 요구한다.

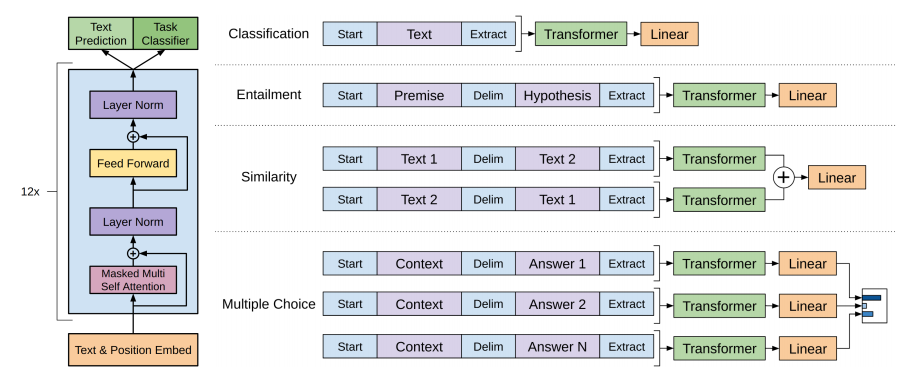
3.3 Task-specific input transformations

Text classification은 논문의 model에 바로 fine-tune 가능이다. 하지만 다른 task QA, TE, 등등은 ordered sentence pairs 또는 3개의 문서 질문 답들등의 structured한 input이다. 하지만 우리 모델은 input이 text의 contiguous한 sequence라서 약간의 수정이 필요하다(task마다).

우리는 representations의 top이 아닌 traversal-style 방식(??이게 뭘까)을 사용한다. 추가적 아키텍쳐의 요소가 필요없고 customization이 쉽다. 단, 앞서 말했듯이 structured input을 ordered sequence로 바꿔줘야한다.

<s>와 <e>는 무작위로 초기화된 토큰이다.

이후 4개의 task에 대한 조절 설명.



음… unsupervised 시의 embed는 뭐 규격이 정해졌다 하지만 다른 task에 대해서 input 조절은 아직 안되어 있다하니 조절을 해주는 것 같다. 근데 그 방식이 좀 다른건가?? Embed 과정 다시보기(pretrain, finetune시 비교해가며)

4. Experiments

Re Review

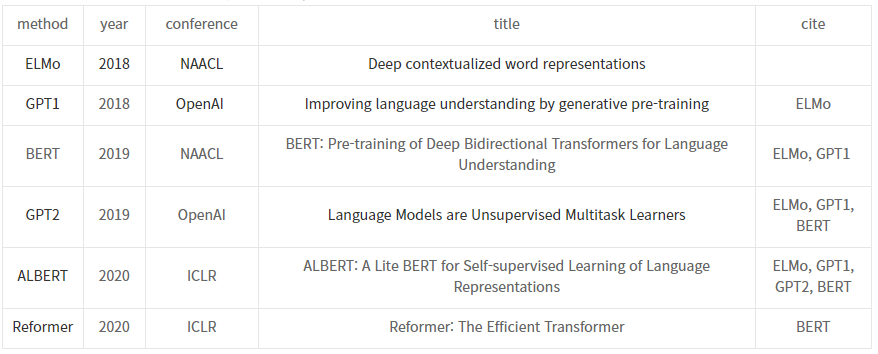
<https://www.quantumdl.com/entry/12%EC%A3%BC%EC%B0%A81-Improving-Language-Understanding-by-Generative-Pre-Training>

<https://davian-lab-junwoo.tistory.com/15>

<https://vanche.github.io/NLP_Pretrained_Model_GPT/>

<https://medium.com/@eyfydsyd97/%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-improving-language-understanding-by-generative-pre-training-gpt1-c65bed865990>

그전에 NLP에서 최근까지 제안된 pretrained model



**Improving Language Understanding by Generative Pre-Training**

**Abstract**

Labeling된 데이터는 극히 일부분

많은 데이터가 있더라도(label or unlabel), 그 데이터가 질이 좋아야 하며, 어떤 방식으로 학습 시키냐가 중요한 문제이다.

지금까지 많은 pretrain embedding을 사용했다.

여기서 확장해서 transfer learning을 한다면 더 좋지 않을까? 하지만 전이 학습도 신중히 잘 해야한다. CNN에서는 이미 전이 학습이 많이 사용되었다. CNN에서는 이미 어느정도 방법이 정해져 있었다. 피처 추출 방법 자체가 확실하게 정해져있고 여기서 확장돼 여러 학습 모델이 제시된다.

NLP에서는 이와 같은 방법이 정립되지 않았다.

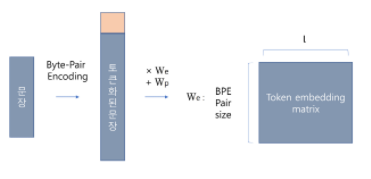
임베딩에 대한 문제도 있는데 문장(tokens)를 어떻게 임베딩할까도 논점이다. 이전에 Word-Level을 넘어 Context-Level로 임베딩하는 ELMo 등이 있었고, Unsupervised Learning을 어떻게 pretrain 할지도 현재 많은 논점이다.

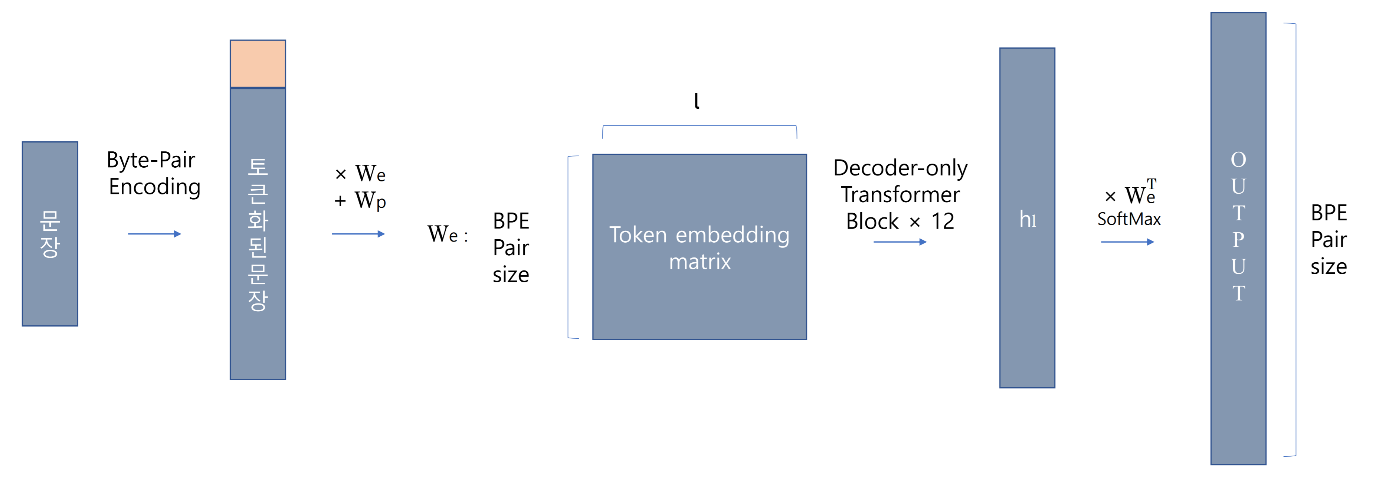
또한 supervised learning으로 fine-tune하여 특정 task에 맞게 모델을 어떻게 구성할 것인가도 논점이다.

**Contribution**

**Unsupervised**

여기도 ELMo 처럼 Unsupervised Learning + LM 이용 하지만 BiLSTM이 아닌 Transformer Block을 이용

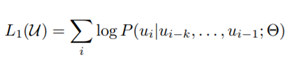




토큰화된 문장 -> 토큰 임베딩 matrix 과정에서 트랜스포머의 디코더 부분만 이용함.

대신 기존 인코더/디코더 각 6쌍 대신에 디코더 12쌍을 이용하는 것(12Layers)

요약: 기본 문장 -> BPE -> 토큰화된 문장 -> 트랜스포머 디코더들 -> 임베딩 matrix

(auxiliary objective 라고도함.)

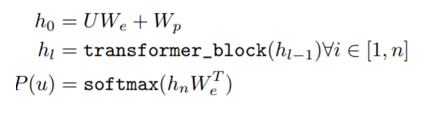
이식은 LM objective function -> 이전 단어보고 현재 단어 뭐일까 맞추는 방법, k는 context window



아래 식을 보면 U는 BPE 후의 토큰화된 문장, W-e는 토큰 임베딩 matrix, W-p는 포지셔널 임베딩 이다. 이렇게 h0은 첫번째 디코더의 입력으로 준비가 된다. 쭉쭉 디코딩을 하다보면 결론적으로 hn(n은 레이어수)이 나오고 마지막 디코더의 아웃풋이다.

그리고 마지막 WeT를 곱하고 softmax하면 backprop을 하기 위한 확률이 나오게 된다.

하지만 우리가 원하는 최종 결과는 We,Wp와 트랜스포머 블록이다. 임을 알 수 있다.(softmax+WeT는 학습을 하기 위한 장치일뿐, 앞에서 이미 We 한번 나와서 가능한듯)



**Supervised**

아래 그림이 모델 구조에 대한 내용이다.

Unsupervised 과정에서는 Global한 NLP 피처를 학습하게 된다.

임베딩할 때 BPE를 사용했었는데 이는 context에 대한 소실이 거의 없이 학습하는 representation 방법이다. 이후 디코더를 통해 task(specific task fine-tune)에 맞는 정답 피처를 추출하며 답을 도출한다.

기존 부분은 동결되는지 아닌지가 궁금하다. (절대 아니겠지)

즉, 트랜스포머를 치면은 각 timestep(위치)에서 을 구할 수 있다.(각 단어의 최종 임베딩이라 보면 될 것 같다.) 여기서 task에 맞게 예측을 해야 되는데 그때에 따른 specific-task weight가 Wy이다.(즉, Wy는 task마다 다르다.)

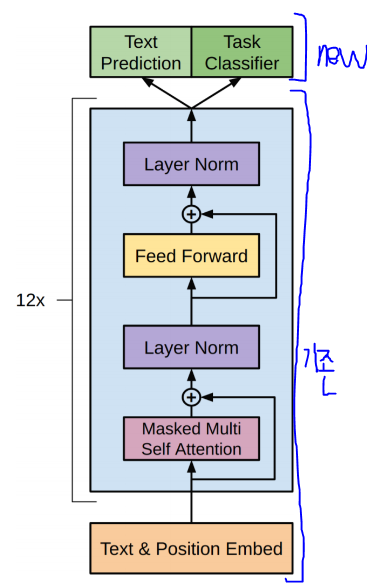
 따라서 이 확률을 최대로 하는 y가 예측 값이고 이걸로 objective를 구성하면 된다.



또한 다음과 같이 regularization 역할을 auxiliary objective가 할 수 있다.

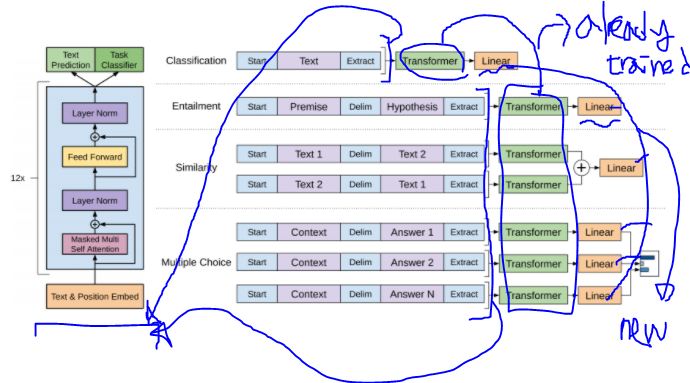


아마 근데 데이터셋이 labeled, unlabeled로 다르니까 end-to-end는 아니고 2번 진행하는 듯 싶다.



위 그림처럼

아래 그림처럼 각 task마다 불편하지만, fine-tune 모델 구조를 조금씩 다르게 해야한다.



**Experiments**

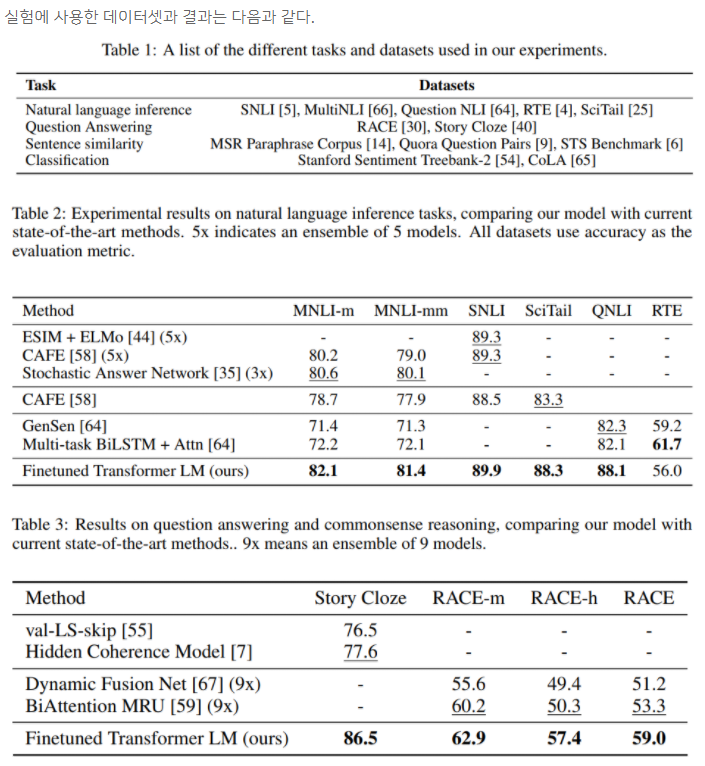
Pretraining 데이터로 BookCorpus Dataset

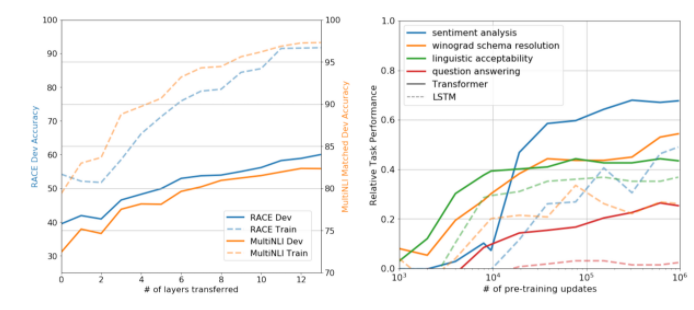
BPE로 임베딩을 했다는점

Activation function? 으로 GELU(Gaussian Error Linear Unit)을 사용한 점

Raw text를 클리닝할 때 fifty library 사용

일부 구두점, 스페이스를 표준화하고 spacy tokenizer를 사용했다.

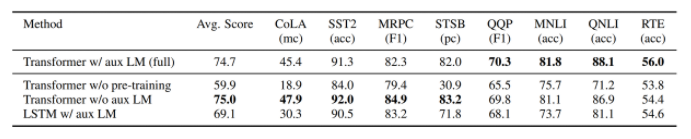




왼쪽은 디코더의 layer를 늘려가며 측정한 RACE, MultiNLI에 대한 정확도(각각 QA Task, textual entailment or NLI Task) => 12개 정도에서 Converge!

오른쪽은 각 Task에 대한 Transformer vs LSTM

아래는 Ablation Studies



이것은 Auxiliary Objective(objective regularization)가 없을 때, pretrain 자체가 없을 때 vs 있을 때

왼쪽 4개와 오른쪽 4개의 결과가 다른데 오른쪽 4개는 데이터셋 크기가 작기 때문이다. 만약 컸다면 더 높을 것으로 예상된다.

또한 pretrain이 없다면 성능 엄청 저하.

**Conclusion**

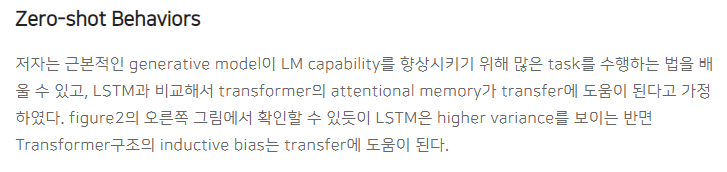
사실 GPT-1은 BERT에 비해 주목 그리 x, BERT가 좀 더 범용적이며 용이, 성능도 더 떨어진다는 소문(SQuAD 1.1, 2.0 결과를 숨긴 이유가 아닐까?)

하지만 그래도 디코더로 트랜스포머 사용한 것이 최초? -> BERT가 이 부분은 캐치한 듯

즉, generative pre-training + discriminative fine-tunning -> task-agnostic model(task를 가리지 않는?), 길고 연속된 텍스트로 이루어진 다양한 종류의 코퍼스로 pre-train -> 상당한 world knowledge와 long-range dependencies!

모델이 개개의 task를 풀 수 있도록함.

그 당시 12개 부문에서 9개 부문 SOTA!(BERT 나오기전)



이건 무슨말..?