# Ch20. GPT models

Journey of the GPT models

Timeline: <https://lifearchitect.ai/timeline/>

## 20.1 GPT-1

GPT 시리즈의 첫 번째 모형으로 OpenAI에서 2018년 6월에 발표한 모형입니다.[[1]](#footnote-1)

참고로 GPT-1은 앞에서 설명한 BERT보다 더 이른 시기에 나왔습니다. BERT와 마찬가지로 자연처 처리 분야에서 전이 학습 방식을 제안한 모형이라고 생각할 수 있습니다.

-----

지금까지 주요한 방법들의 한계 => 지도학습을 위해서 많은 양의 labeled data가 필요, 하지만 이러한 많은 양의 학습 데이터를 준비하는 것은 쉽지 않아.

특정 도메인에서의 데이터를 이용해서 학습된 모형을 다른 도메인에 적용하는 것이 쉽지 않아

정답이 없는 데이터(unlabeled data)를 사용할 수 있다면 큰 도움

많은 양의 학습 데이터가 있는 경우에도 토큰들의 좋은 representation을 배우는데 있어서 (즉, 언어의 특성을 나타내는 representation) 많은 양의 unlabeled data 를 사용하는 비지도 학습 방법은 큰 도움이 될 것으로 판단됨

-----

해당 논문에서는 비지도 학습을 이용한 사전 학습 방법과 지도 학습을 이용한 미세 조정 방법을 결합한 방법을 제안합니다. 즉, 대용량의 텍스트 데이터를 비지도 학습 방법으로 사전 학습한 후, 특정한 다운스트림 작업을 위해 준비된 정답이 존재하는 데이터를 이용해서 사전 학습된 모형을 미세 조정하는 방법을 제안한 것입니다. 해당 논문에서는 이러한 방법을 semi-supervised 학습 방법이라고 일컫습니다.

해당 논문의 주된 목적은 비지도 사전 학습을 통해서 단어 혹은 문서의 일반적인 특성을 배우고, 그러게 습득된 결과를 되도록 미세 조정 없이 새로운 작업에 적용할 수 있는 모형을 만드는 것입니다.

GPT-1에서도 트랜스포머를 기본 모형으로 사용합니다.

특정 다운스트림 작업을 위한 미세 조정시에는 입력 데이터를 주어진 다운스트림 작업에 따라 적절한 형태로 변환하여 입력합니다. => During transfer, we utilize task-specific input adaptations derived from traversal-style approaches [52], which process structured text input as a single contiguous sequence of tokens. As we demonstrate in our experiments, these adaptations enable us to fine-tune effectively with minimal changes to the architecture of the pre-trained model. (굳이 설명하지 않아도 될 듯)

해당 논문에서는 제안된 모형을 natural language inference, question answering, semantic similarity, and text classification 에 대해서 평가를 합니다. 그리고 대부분의 작업에 대해서 기존의 sota 모형들 보다 좋은 성능을 보여줍니다. => For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test) [40], 5.7% on question answering (RACE) [30], 1.5% on textual entailment (MultiNLI) [66] and 5.5% on the recently introduced GLUE multi-task benchmark [64].

제로 샷 행동 관련 내용 => We also analyzed zero-shot behaviors of the pre-trained model on four different settings and demonstrate that it acquires useful linguistic knowledge for downstream tasks.

GPT-1에서의 학습

GPT-1는 학습이 두 단계로 이뤄집니다. 첫 번째는 대용량의 데이터를 이용한 비지도 사전학습 (unsupervised pre-training)이고, 두 번째는 다운스트림 작업을 위한 지도 미세조정 (supervised fine-tuning) 단계입니다. 다운스트림 작업의 경우는 별도의 학습 데이터를 사용합니다.

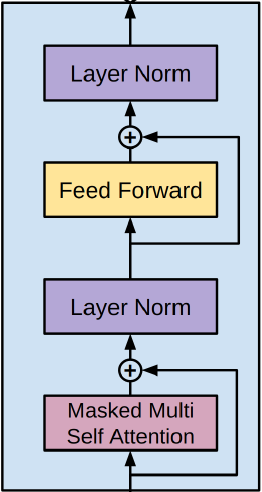
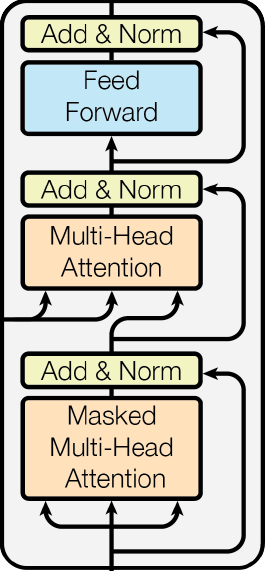
1) 비지도 사전학습

이전 단어들의 정보를 이용해서 다음에 나올 단어가 무엇인지를 예측하는 방식으로 학습이 진행됩니다.

모형의 구조

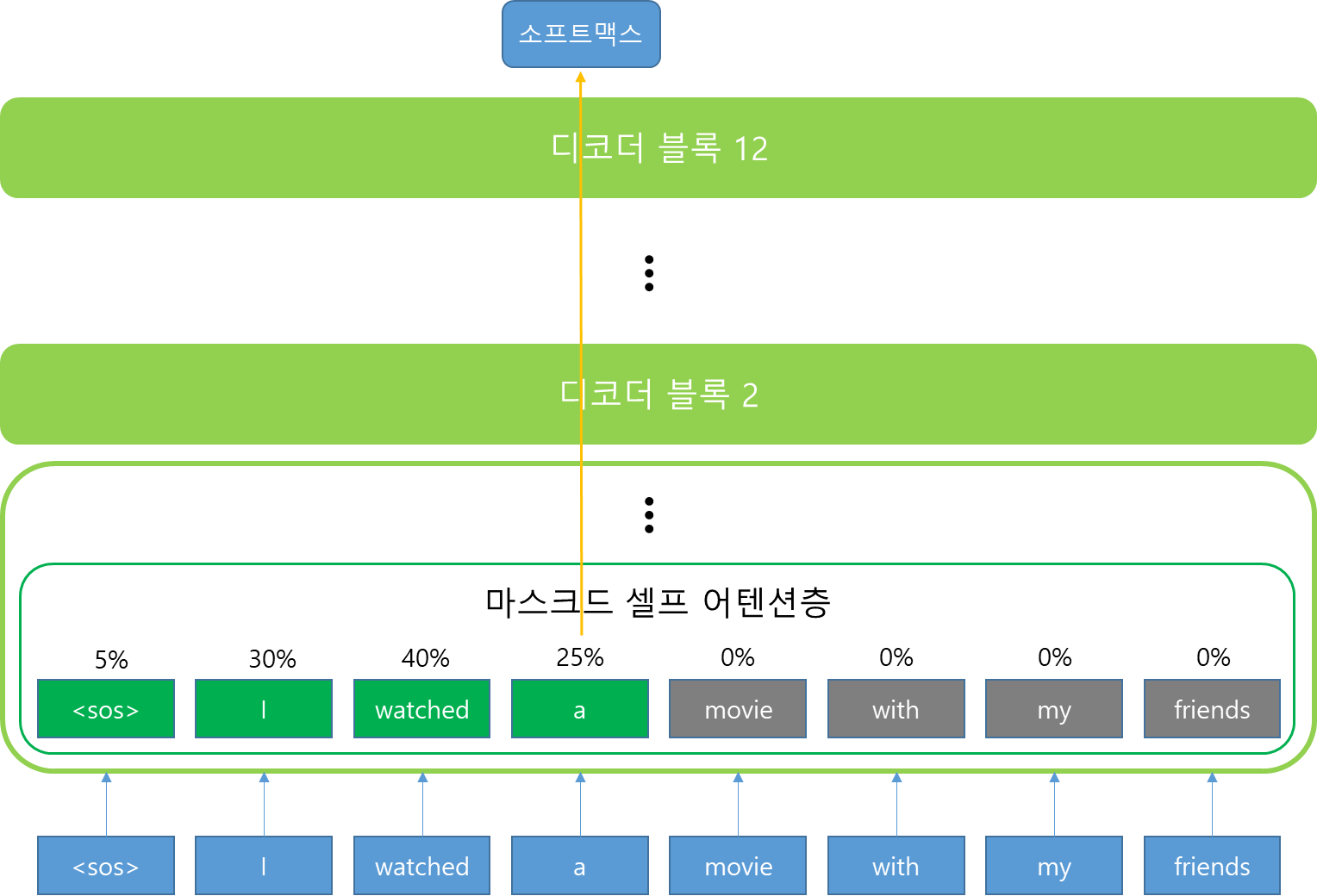
그리고 학습을 위해서 트랜스포머의 디코더를 사용합니다.

12개의 디코더 블록을 사용합니다. 트랜스포머의 디코더 블록과 GPT-1에서 사용된 디코더 블록의 구조를 살펴보도록 하겠습니다.

여기서도 기본적으로 임베딩 정보와 위치 임베딩 정보를 사용하여 각 토큰을 표현하고, 그렇게 표현된 임베딩 벡터가 디코더 블록의 입력값으로 입력됩니다. 그리고, 디코더 부분이기 때문에 마스크드 멀티-헤드 셀프 어텐션이 사용되었습니다. 그리고 언어 모형으로 작동하기 위해서 제일 마지막에 소프트맥스 활성화 함수 기반의 출력층을 사용하였습니다.

어떻게 학습 되는지에 대한 그림 포함?



예를 들어 ‘I watched a movie with my friends’ 라고 하는 시퀀스 데이터에 대해서 I watched a 토큰 정보를 이용해서 다음에 나올 토큰을 예측한다고 가정해 보도록 하겠습니다. 이는 그림 X와 같이 표현될 수 있습니다. 트랜스포머의 디코더 부분을 이용해서 출력되는 토큰 a의 은닉 상태 벡터 정보를 이용해서 소프트맥스 활성화 함수가 존재하는 출력층을 이용해서 다음 토큰 (즉, movie)을 예측하게 됩니다. 소프트맥스 출력층의 출력값을 이용해서 교차 엔트로피 비용함수가 계산된다고 생각할 수 있습니다. 그리고 각 디코더 블록에서는 마스크드 셀프 어텐션이 사용되었기 때문에 a 다음에 나오는 단어들의 정보는 마스킹 처리되어 a의 은닉 상태 벡터를 계산할 때 사용되지 않습니다 (위 그림에서 회색으로 표현되었습니다).

학습 데이터

사전 학습에서는 BookCorpus 데이터셋을 사용하였습니다. 해당 데이터셋은 앞에서 살펴본 것 처럼 출간되지 않은 7천 권이 넘는 책에 대한 내용을 포함하고 있습니다. 저자들에 따르면 해당 논문에서는 대안적으로 사용될 수 있는 Word Benchmark 데이터셋 대신 BookCorpus를 사용했는데, 주된 이유는 해당 데이터셋이 더 긴 시퀀스 데이터로 구성되어 있어 멀리 떨어져 있는 단어들 간의 연결 관계를 파악하는데 더 적합하기 때문이라고 합니다.

모형의 구조

12개의 디코더 블록

임베딩 벡터의 차원 = 768

헤드의 수 = 12

위치 기반 완전연결층이 갖는 노드의 수 = 3,072

Adam 옵티마이저 사용

최대 학습률 = 2.5e-4

하나의 입력 시퀀스 길이 = 512

미니 배치 크기 = 64

에포크 수 = 100

가중치 파라미터는 N(0, 0.02)와 같이 분산의 크기가 동일한 정규분포를 사용하여 초기화 하였는데, 저자들에 따르면 이는 사용되는 계층 정규화층이 많기 때문에, 분산의 크기를 고정한 초기화 방법을 사용하여도 충분하다고 합니다.

그외의 내용으로는 10%의 드롭아웃을 적용하였고, L2 규제화와 유사한 방법의 규제화를 진행

활성화 함수로는 GELU를 사용

위치 임베딩의 경우 삼각함수를 사용한 원래의 트랜스포머 방법과 달리, 위치 임베딩 벡터가 갖는 원소의 값들은 학습을 통해 그 값을 결정하였습니다.

토크나이제이션: 바이트 페어 인코딩

BERT와 다르게 바이트 페어 인코딩 토크나이제이션 방법을 사용하였습니다.

2) 지도 학습 기반의 미세 조정

다운스트림 태스크를 위한 정답 데이터(labeled data)를 사용합니다. 이러한 데이터를 라고 표현하겠습니다. 각 입력 샘플은 여러 개의 토큰()과 레이블 정보 (즉, 정답 정보, )로 이루어져 있습니다.

이 입력 시퀀스는 사전 학습된 트랜스포머 디코더 모형에 입력되고 마지막 토큰에 대한 은닉 상태 벡터 ()가 출력됩니다. 그리고 이러한 은닉 상태 벡터는 소프트맥스 활성화 함수를 갖는 완전 연결층의 입력값으로 입력되고, 정답 (즉, y)에 대한 예측치가 출력됩니다. 해당 완전연결층은 파라미터를 갖습니다. 이를 식으로 나타내면 아래와 같습니다.

이에 대한 비용함수를 라고 하겠습니다. (는 미세조정에 사용되는 정답 데이터를 의미합니다).

GPT-1 논문에서는 미세 조정을 위한 학습시 이외에 보조적인 비용함수를 하나 더 사용했습니다. 사전 학습에서와 마찬가지로 데이터 에 대해서도 다음 토큰을 예측하는 언어 모형을 적용하고 그에 대한 비용함수를 추가 비용함수로 사용한 것입니다. 데이터 에 대한 언어 모형의 비용함수는 으로 표현합니다. 미세 조정에서 사용된 최종 비용함수는 아래와 같습니다.

위 식에서 는 을 얼마만큼 반영할 것인지에 대한 가중치가 됩니다. 논문에서는 로 설정되었습니다. 논문의 저자들에 따르면, 보조 비용함수 (즉, )을 사용하게 되면 학습이 보다 안정적으로 수행되고, 모형이 갖는 일반화 정도가 커지는 효과가 있다고 합니다.

미세 조정에서 추가적으로 사용되는 파라미터는 와 미세 조정에서 새롭게 추가적으로 사용되는 구분자 토큰들에 대한 임베딩 벡터를 위한 것들 뿐입니다.

Overall, the only extra parameters we require during fine-tuning areWy, and embeddings for delimiter tokens (described below in Section 3.3).

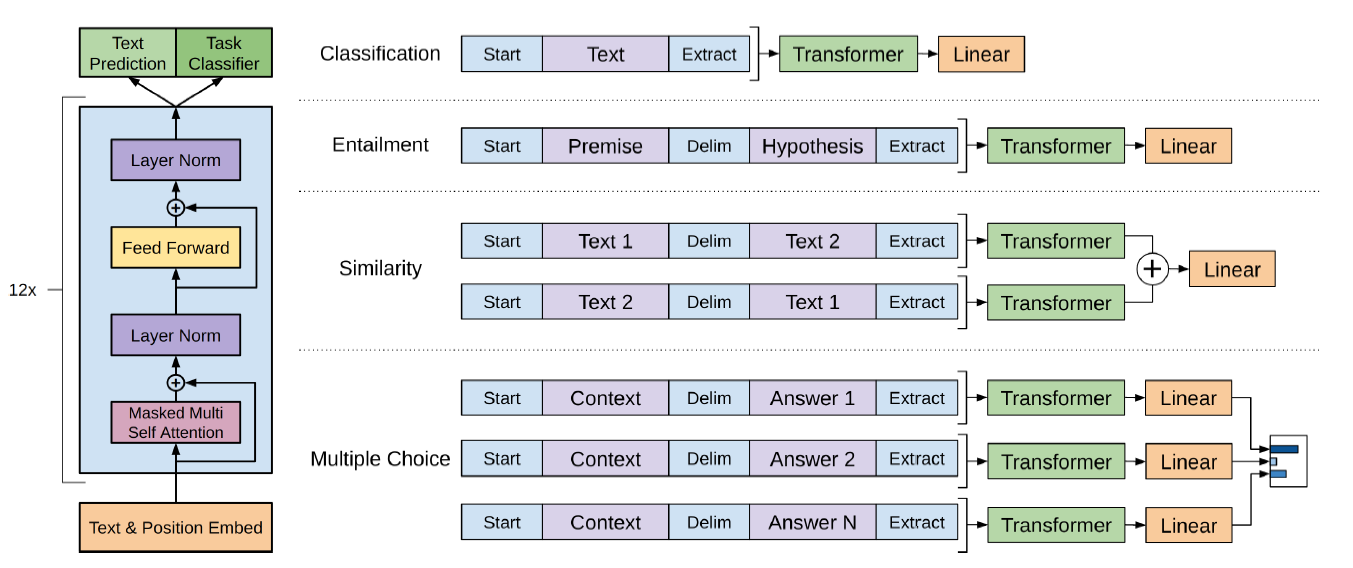
입력 데이터 형태 변환

사전 학습된 모형이 적용되는 다운스트림 작업들의 경우, 작업의 종류에 따른 특정한 형태로 입력 데이터를 입력 받아야 합니다. 예를 들어, 질의∙응답 (Question Answering) 작업의 경우, 질의와 응답의 구조로 입력되어야 하고, 자연어 추론의 경우는 서로 다른 시퀀스 데이터가 순서를 가지고 입력되어야 합니다.

본 논문에서는 일일이 입력되는 데이터의 구조를 이러한 식으로 변환하지 않았습니다. 너무 많은 작업이 필요로 하기 때문에.

GPT-1에서는 사전 학습 모형이 적용되는 다운스트림 작업의 종류와 상관없이 입력 데이터를 (사전 학습 모형에 입력될 수 있는) 순서를 갖는 여러 개의 시퀀스 데이터의 형태로 입력하였습니다.

그림 X에서 오른 쪽 부분은 다운스트림 작업에 따라 입력된 데이터의 형태를 보여주고 있습니다. 해당 그림에서 <Start>는 입력 시퀀스 데이터의 시작을 의미하는 토큰이고, <Extract>는 종료 (end) 토큰을 나타냅니다. 그리고 서로 다른 종류의 시퀀스를 구분하기 위해서 구분자(delimiter)에 해당하는 <Delim> 토큰을 사용하였습니다.

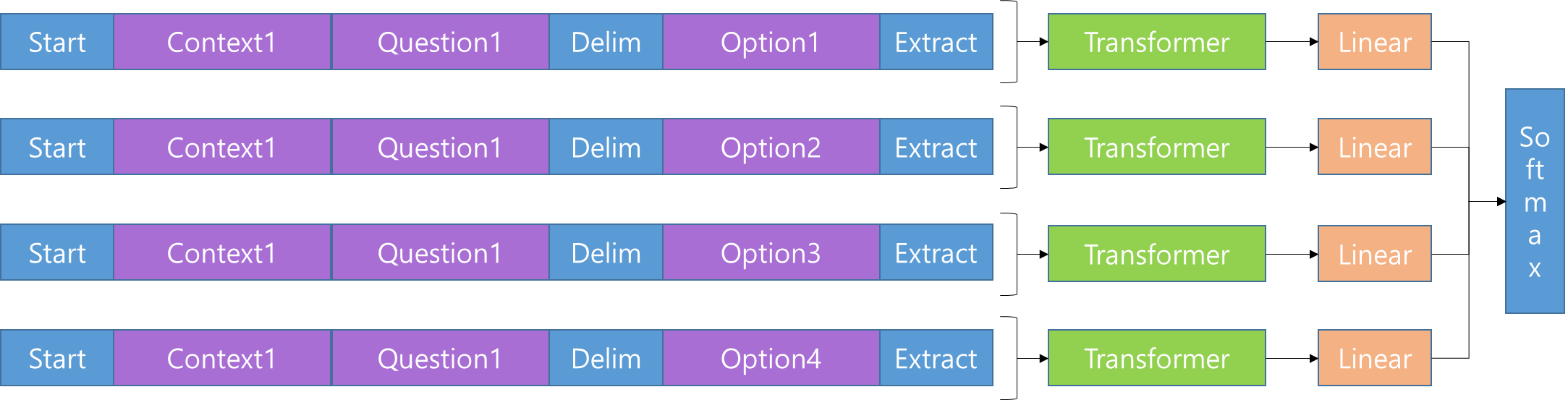


예를 들어, 문서 분류 (Classification)의 경우는 별도의 변환없이 문서에 해당하는 Text를 시작 토큰과 종료 토큰을 추가한 후 입력하여 사전 학습 모형(위 그림에서는 Transformer로 표현)이 출력하는 은닉 상태 벡터를 소프트맥스 활성화 함수를 갖는 완전연결층 (위 그림에서는 Linear로 표현)의 입력값으로 입력하여 교차 엔트로피 비용함수를 계산합니다.

텍스트 함의(Textual entailment) 작업의 경우는 전제(premise)에 해당하는 텍스트와 가설(hypothesis)에 해당하는 텍스트를 구분자를 이용해 연결하여 입력하였습니다 (역시나 마찬가지로 시작, 종류 토큰이 추가됩니다). 이러한 입력데이터에 대해서 사전 학습 모형이 출력하는 은닉 상태 벡터에 소프트맥스 활성화함수를 갖는 완전연결층을 적용하여 정답을 예측하게 됩니다. 일반적인 텍스트 함의 문제의 정답은 참, 거짓, 모름 등이 됩니다.

두 텍스트에 대한 의미 유사 (similarity) 여부 작업[[2]](#footnote-2)의 경우, 두 텍스트(예, Text1, Text2)가 어떠한 순서관계도 갖지 않는다는 것을 반영하기 위해서 순서를 달리한 두 개의 입력 데이터를 입력합니다. 하나는 Text1이 먼저 입력되는 형태이고, 다른 하나는 Text2가 먼저 입력되는 형태입니다. 각 입력 데이터를 사전학습 모형에 입력되어 출력되는 은닉 상태 벡터를 원소별로 더한 결과를 최종 출력층의 입력값으로 사용합니다. 정답은 유사, 비유사가 됩니다.

선다형(multiple choice)의 질의∙응답 작업의 경우, 문맥 문서와 질문을 하나의 텍스트로 구성한 후 각 보기의 텍스트를 구분자로 연결한 여러 개의 텍스트를 입력 데이터로 사용하였습니다. 각 입력 텍스트에 대해 완전연결층이 출력하는 값을 로짓값으로 사용하여 최종적으로 소프트맥스 활성화 함수를 사용하여 각 보기가 정답일 확률을 계산하였습니다. 예를 들어, 문제 1에 대한 문맥 텍스트 (Context 1이라고 하겠습니다)가 존재하고 보기가 4개가 존재한다라고 하는 경우에는 다음과 같이 입력되는 것입니다. 그림 X에서는 Context와 Question을 합쳐서 Context로 표현하였고, 각 보기 (Option)을 Answer로 표현했습니다.

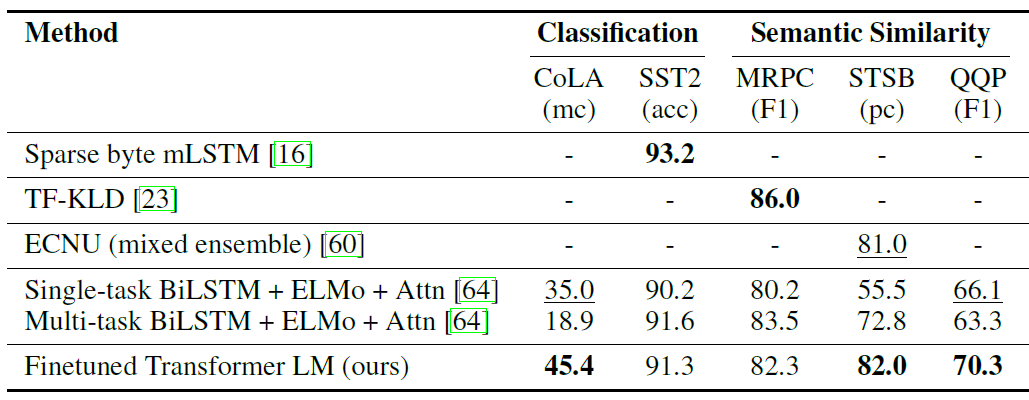


미세 조정 학습

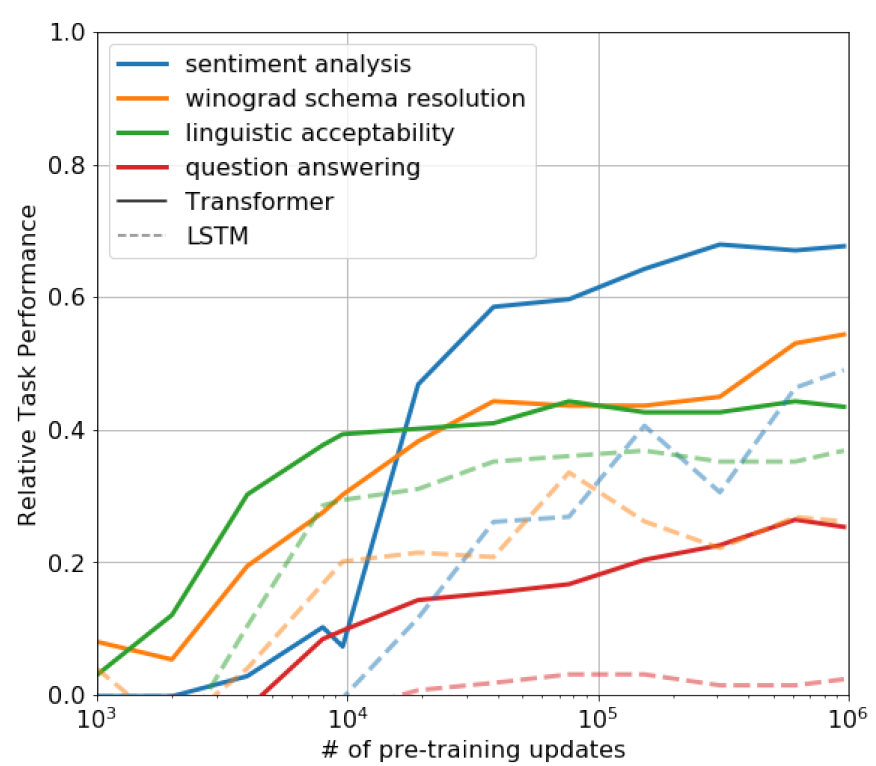
사전 학습에서의 하이퍼파라미터 값을 대부분 그대로 사용하였습니다. 학습률은 6.25e-5으로 하였고, 미니배치의 크기는 32로 하였습니다. 미세 조정의 경우에는 에포크의 값을 대부분 3 정도로 사용하였습니다. 이는 미세 조정을 많이 하지 않더라도 주어진 문제를 푸는데 최적의 파라미터 값을 찾을 수 있었다는 것을 의미합니다.

모형의 성능

12개의 작업들 중에서 9개의 작업에 대해 당시 SOTA 모형의 성능보다 우수한 성능을 보였습니다. 자연어 추론이나 질의응답 관련 작업들에서는 모두 SOTA 모형들 보다 우수한 결과를 나타냈습니다. 다만, 문서 분류나 의미 유사 여부 작업의 경우는 사용된 데이터셋에 따라 기존 모형의 성능이 더 우수한 경우도 있었습니다.



제로샷 행동 (Zero-shot behavior)



제로샷이라고 하는 것은 추가적인 미세 조정없이 주어진 다운스트림 작업을 사전 학습 모형을 그대로 이용해서 수행하는 것을 의미합니다. 물론 여기서도 특정 다운스트림 작업을 위한 분류기에 존재하는 파라미터의 값은 학습이 됩니다.

제로샷 상황에서도 성능이 지속적으로 안정적으로 지속적으로 증가하는 것을 보여주고 있습니다. 그에 반해 LSTM의 경우는 학습이 증가하더라도 제로샷 성능이 지속적을 증가하지 않고 불안정한 모습을 보이는 것을 확인할 수 있습니다. 이는 트랜스포머 디코더 기반의 모형이 사전 학습을 통해 다양한 다운스트림 작업을 수행할 수 있는 능력을 학습하기 때문이라고 저자들은 생각합니다.

## 20.2 GPT-2

이번에는 GPT-2에 대해서 살펴보겠습니다. 해당 모형에 대한 논문은 2019년에 Language Models are Unsupervised Multitask Learners 라는 이름으로 발표되었습니다.[[3]](#footnote-3) GPT-2도 GPT-1과 마찬가지로 트랜스포머의 디코더 부분을 사용하였습니다. GPT-1과의 주요한 차이는 더 많은 데이터를 이용해서 학습을 하였고, 더 많은 파라미터를 갖는 모형을 사용했다라는 것입니다.

GPT-2 논문의 가장 큰 목적은 다운스트림 작업에 따라 추가적인 미세 조정 없이 (혹은 약간의 미세 조정만 사용) 사전 학습된 모형을 이용해서 주어진 문제를 풀 수 있는 사전 학습 모형을 제안하는 것입니다. 그리고 그 결과로 제안된 모형이 GPT-2입니다.

대용량의 질이 좋은 학습 데이터로 사전 학습된 언어 모형이 다양한 다운스트림 작업에 대해 갖는 제로샷 성능을 파악해 보는 것이 본 논문의 가장 큰 목적입니다.

저자들에 따르면, 가장 큰 버전의 경우 8개 중 7개의 작업에 대해서 추가적인 미세 조정 없이 SOTA 성능을 내는 것으로 나타났습니다.

특정 작업을 위한 지도 학습 방식의 문제점 존재: 일반화가 잘 안된다. => 이러한 문제의 주요한 원인은 단일 작업을 위한 단일 데이터셋을 사용하는 것이 주된 이유일 것이라고 저자들은 생각합니다.

저자들에 따르면 기존에 제안된 multitask learning (Caruana, 1997) 방식은 많은 양의 새로운 데이터를 지속적으로 마련해야 한다는 단점으로 인해 적합하지 않다라고 얘기합니다.

논문이 출간되었던 시점에서 가장 성능이 좋은 방법은 사전 학습과 지도 학습 기반의 미세 조정 방법을 같이 사용하는 것입니다. 즉, 전이학습 방법입니다. 하지만, 이러한 방법은 여전히 특정 작업을 수행하기 위해서 지도 학습이 추가적으로 필요합니다. 즉, 지도학습을 위한 (많은 양의) 정답 데이터가 있어야 한다는 것을 의미합니다.

본 논문에서는 추가적인 미세 조정 없이도 성능이 좋은 사전 학습 기반의 언어 모형을 제안합니다. 그리고 이러한 방법을 통해서도 좋은 성능을 낼 수 있다는 것을 증명합니다.

해당 모형의 가장 중요한 핵심은 언어 모형입니다. 즉, 대용량의 텍스트 데이터에 대해서 다음 토큰을 예측하는 방식으로, 즉, 비지도 학습 방식, 사전 학습된 언어 모형을 추가적인 미세 조정없이 다양한 다운스트림 작업에 적용하는 것입니다.

단일 작업의 경우는 P(output|input)에 해당하는 확률 분포를 예측하는 것이 주요한 문제

여러 작업의 경우는 확률 분포의 형태를 P(output|input, task)의 형태로 표현해야 함 => 왜냐하면, 동일한 형태의 input이라고 할지라도 주어진 작업에 따라서 정답이 달라질 수 있기 때문입니다.

이를 Task conditioning이라고 합니다.

이는 여러 형태로 구현 가능합니다. 예) 모형의 구조 레벨 (구조를 달리한다), 알고리즘 레벨

하지만, McCann et al. (2018)의 논문에서 한 것 처럼, 작업 (task)에 대한 것도 텍스트의 형태로 입력하는 것이 가능

예) 한글을 영어로 번역하는 작업의 경우 => (translate to English, Korean text, English text)

독해 (reading comprehension) 문제의 경우 => (answer the question, document, question, answer)의 형태

위의 예에서 ‘translate to English’와 ‘answer the question’이 어떠한 작업을 수행하고자 하는지에 대한 텍스트 데이터가 됩니다.

저자들에 따르면, 어떠한 결과물을 출력해야 하는지에 대한 명시적인 제시 없이도 풀어야 하는 작업을 학습하는 것이 가능합니다. 즉, 목적이 무엇인지 명확하게 제시하지 않아도 언어 모형이 알아서 목적이 무엇인지 혹은 수행해야하는 작업이 무엇인지도 학습하는 것이 가능하다는 것을 의미합니다. 하지만, 저자들에 따르면 이를 위해서는 많은 양의 학습 데이터가 필요하다고 합니다.

Our speculation is

that a language model with sufficient capacity will begin

to learn to infer and perform the tasks demonstrated in

natural language sequences in order to better predict them,

regardless of their method of procurement.

저자들은 성능이 좋은 언어 모형의 경우, 텍스트의 형태로 제시되는 태스크가 무엇인지 (그러한 제시문이 없을지라도) 추론할 수 있고, 그러한 추론을 바탕으로 주어진 태스크를 수행할 수 있을 것이라고 생각합니다. 그리고 이러한 것이 가능하다면 다양한 작업을 학습을 비지도 방식으로 학습하는 것도 가능할 것이라고 말합니다. 그리고 그러한 가능성을 해당 논문에서 검증합니다.

학습 데이터

저자들은 많은 선행연구들이 언어 모형을 특정 도메인의 텍스트 데이터를 이용해서 학습했다고 하면서, 되도록 다양한 도메인과 문맥에서의 작업을 나타낼 수 있는 자연어를 포함하는 거대하고 다양성이 높은 데이터셋을 구축하고자 노력합니다.

인터넷에 존재하는 텍스트 데이터를 이용하는 것이 그러한 방법이 될 수 있습니다. 기존에 존재하는 Common Crawl 이라는 데이터셋이 존재하지만, 해당 데이터셋에는 질이 좋지 않은 텍스트가 다량 존재한다는 문제가 있습니다.

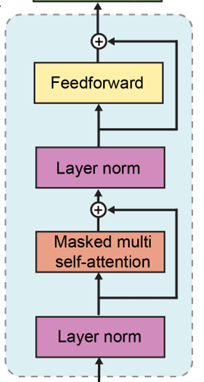
이러한 문제를 해결하기 위해서 즉, 질 높은 학습 데이터를 구축하기 위해서 저자들은 직접 인터넷으로부터 데이터를 수집합니다. 레딧이라고하는 소셜 미디어 플랫폼에서 적어도 세 개의 카르마(karma)를 받은 게시글에 입력된 링크만을 수집합니다. 레딧에서의 카르마는 좋아요수에서 싫어요수를 뺀 값이라고 생각할 수 있습니다. 즉, 카르마의 수를 3으로 제한함으로써 어느 정도 다른 사람들에게 긍정적으로 평가된 게시글에서 소스로 사용된 링크만을 수집하겠다라는 것을 의미합니다.

저자들은 이렇게 수집된 데이터를 WebText라고 표현하였습니다. WebText 데이터에는 대략적으로 800만개가 조금 넘는 문서들이 포함되어 있으며, 전체 사이즈는 40GB 정도가 됩니다. 저자들은 위키피디아 페이지에 해당하는 문서는 중복될 가능성이 높아 WebText에 포함하지 않았다고 합니다.

GPT-2에서도 GPT-1에서와 마찬가지로 BPE 토크나이제이션 방법을 사용했습니다.

모형

GPT-2도 GPT-1와 같이 기본적으로 트랜스포머의 디코더 부분을 사용합니다. 하지만 약간 변형합니다. 계층 정규화층의 순서를 바꿉니다. 가장 아랫부분으로, 그리고 추가적으로 하나 더 사용, 이는 셀프 어텐션층 이후에 추가 (아래 그림 참고)

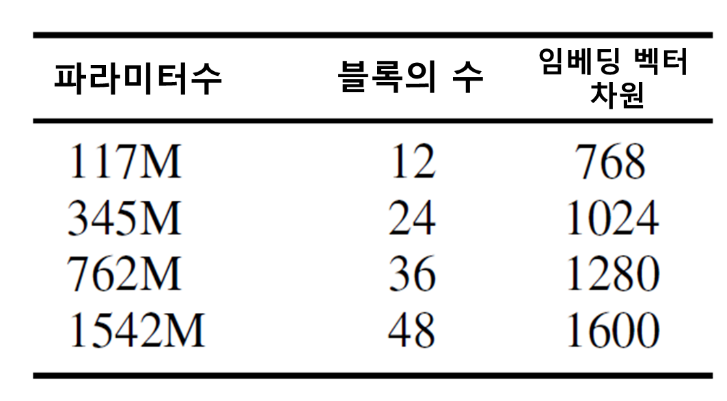


가중치 초기화를 다른 방법으로 수행 => Residual connection에 사용된 층의 수를 반영

입력되는 시퀀스의 길이를 512에서 1024로 증가

미니 배치의 크기는 512 사용

모형의 크기



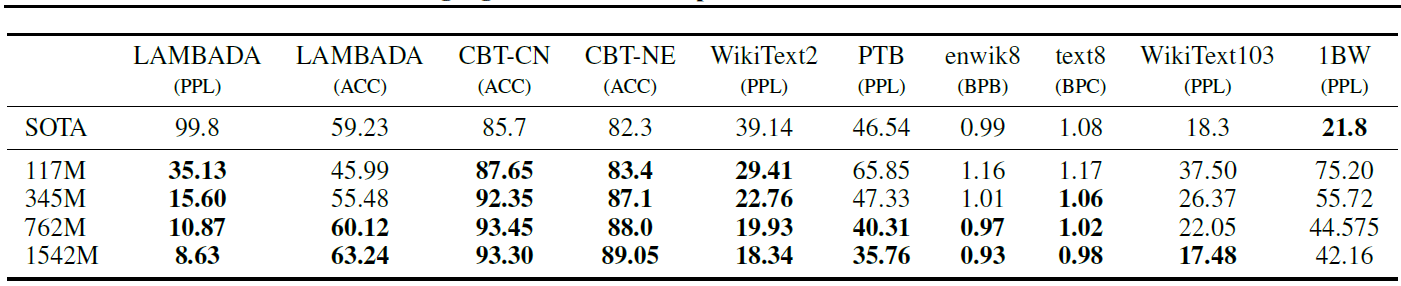
실험 결과

8개의 작업에 대해 제로샷 성능 파악

언어 모형으로서의 성능

WebText 데이터를 이용해서 사전 학습된 언어 모형 즉, GPT-2가 추가적인 미세 조정 없이 (즉, 제로샷) 다른 데이터셋에 대해서 언어 모형으로서의 성능이 어떻게 되는지를 파악했습니다. 다른 데이터셋으로는 LAMBADA, Children’s Book Text (CBT), Penn Treebank (PTB), WikiText-2, One Billion Word Benchmark 등이 있습니다.

Children’s Book Text는 다른 카테고리의 단어들에 대한 언어 모형의 성능을 검사하기 위해서 생성된 데이터셋입니다. 그리고 LAMBADA는 단어들 간의 장기의존 관계를 파악하기 위해서 구축된 데이터셋입니다. LAMBADA에서의 태스크는 사람의 관점에서 마지막 단어를 제대로 예측하기 위해 최소 50개의 토큰이 필요한 문장들의 마지막 단어를 예측하는 것입니다.



그외의 작업

그외의 작업으로는 요약 (summarization), 번역, 독해 (Reading Comprehension), 질의응답 등이 있습니다.

Task conditioning의 목적으로 작업에 따른 특정 문자열 데이터를 추가적으로 입력하였습니다. 예를 들어, 요약의 경우에는 요약을 해야하는 텍스트 다음에 요약할 것을 의미하는 ‘TL;DR:’을 추가하였으며, 번역의 경우에는 번역 작업을 나타내기 위해서 ‘korean sentence = english sentence’ 형태로 예제 데이터를 입력한 후, 번역해야하는 텍스트를 ‘korean sentence =’와 같은 형태로 입력 하였습니다. 예를 한글 문장 “오늘은 금요일입니다”를 영어로 번역한 결과를 얻고자 하는 경우, 아래와 같이 입력합니다.

‘나는 영화를 봅니다 = I watch a movie <delimiter> 오늘은 금요일입니다 =’

번역의 경우는 모형이 생성하는 첫 번째 문장을 최종 결과물로 사용했습니다.

참고: greedy decoding

디코더를 이용해서 하나씩 예측하는 것을 의미

저자들은 zero-shot task transfer 성능이 좋은 것에 대해서 혹시 모델이 학습데이터에서

기억하고 답을 내는 것인지 아니면 정말 일반화된 모델인 것인지에 대한 의문을 해소하기 위해

몇 가지 실험을 더 해보았습니다. 첫째로, 실제 학습 데이터셋과 테스트 데이터셋에 겹치는

데이터가 얼마나 있는지 확인해 보았습니다. (이미지 데이터셋인 CIFAR-10에도 3.3% 중첩이

있다고 합니다.) 그 결과, WebText에는 평균 3.2%의 중복이 있었고 일반적으로 사용하는 다른

학습 데이터셋에는 오히려 더 높은 평균 5.9%의 중복이 있었다고 하며, 중복을 제거해도 성능이

거의 떨어지지 않았습니다. 둘째로, 학습 데이터를 기억하고 모델 추론 능력이 떨어지는 것이라면

자체 데이터셋 WebText 내에서 성능이 안 좋고 과적합이 있어야 하지만, 확인 결과(Fig. 6) 모델

크기가 커짐에 따라서 성능이 개선되고 있고 과적합도 나타나지 않았습니다. 마지막으로, 말하는

유니콘 발견에 대한 뉴스기사 작성처럼 기존 학습 데이터셋에 존재하지 않는 완전히 새로운

텍스트도 생성이 가능해, 종합적으로 암기하고 답을 하는 것이 아니라 실제로 추론하는 것이라는

결론을 내렸습니다.

## 20.3 GPT-3

GPT-3는 2020년 5월에 발표된 모형입니다.[[4]](#footnote-4) GPT-3에서 가장 크게 주목한 점은 사전 학습이 충분히 된 언어 모형을 사용하는 경우, 미세 조정 과정없이도 풀고자 하는 문제 관련하여 소수의 예를 이용해서도 좋은 성능을 낼 수 있다는 것입니다. 본 논문에서는 이렇게 소수의 예를 이용해서 작업을 수행하는 것을 퓨샷(few-shot) 방법이라고 했습니다. 사실 20.2에서 살펴본 것 처럼, GPT-2에서도 추론 작업을 할 때 풀고자 하는 문제와 관련된 예제를 입력 데이터의 일부로 입력하기도 했습니다. 예를 들면, 번역의 경우, 번역의 예를 입력 데이터의 일부로 사용했습니다. 하지만, GPT-2에서는 이러한 방법을 제로샷 방법이라고 했는데, GPT-2 논문에서의 제로샷의 의미는 몇 개의 예제를 사용했느냐와 관련있는 것이 아니라, 사전 학습 모형에 대한 추가적인 학습 즉, 미세 조정 과정이 있느냐 없느냐와 관련이 있는 것으로 추가적인 파라미터 학습이 없는 경우를 제로샷이라고 표현했었습니다.

하지만, GPT-3에서의 제로샷은 조금 다른 의미를 갖습니다. GPT-3에서는 추론 과정에서 몇 개의 예제 샘플을 입력 데이터의 일부로 사용했느야에 따라서 크게 세 가지로 구분합니다. 제로샷, 원샷, 퓨샷이 그것입니다. 제로샷은 예제 샘플을 하나도 사용하지 않은 것을 의미하고, 원샷은 한 개의 예제 샘플을 사용한 것을, 퓨샷은 두 개 이상 (보통 10 - 100개의 예제 샘플)의 예제 샘플을 사용한 경우를 의미합니다.

## 참고 시작 ##

메타 학습 (Meta-learning)

이 표현은 일반적으로 모형이 학습을 통해서 다양한 스킬을 발전시키고 패턴을 인식할 수 있는 능력을 얻고, 그리고 그러한 스킬을 이용해서 추론 과정에서 주어진 문제를 빠르게 인지하고 문제에 적응하는 것을 의미합니다. 혹은 이러한 방식으로 작동되는 전반적인 학습 방법을 일컫는 표현입니다.

## 참고 끝 ##

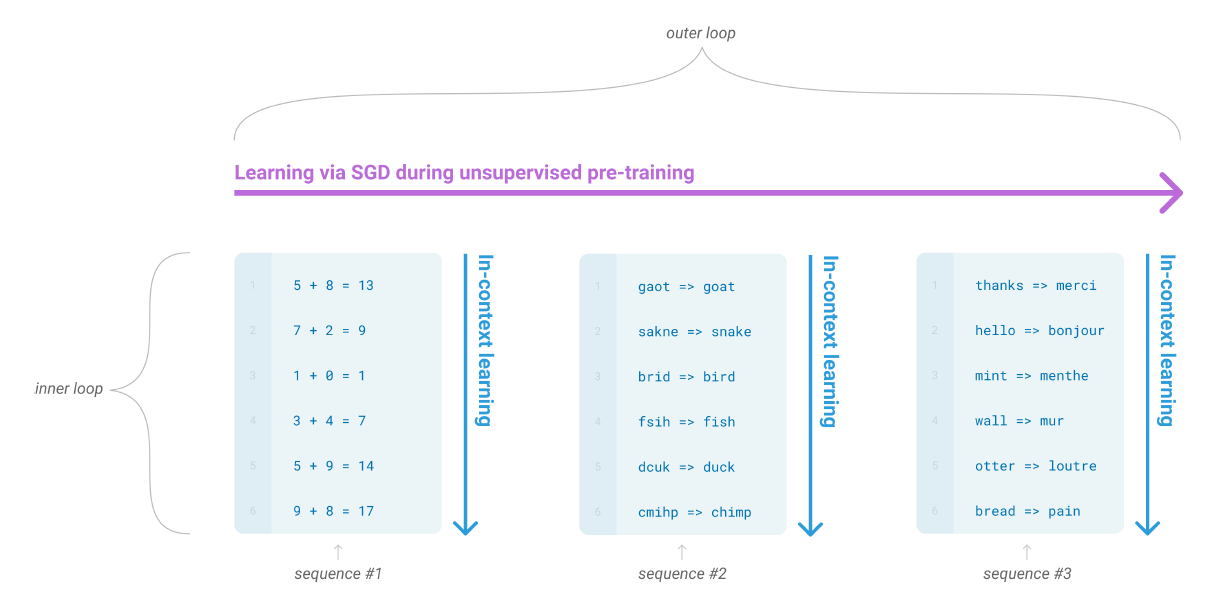


Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term “in-context learning” to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

해당 논문에서는 1,759억(175 billion)개의 파라미터를 갖는 사전 학습 언어모형 (즉, GPT-3)이 퓨샷 상황에서 갖는 성능을 평가합니다.

GPT-3는 다양한 다운스트림 작업에 대해서 좋은 성능을 내는 것으로 나타났습니다. 그리고 기사 생성에도 훌륭한 성능을 나타냈습니다. 하지만, 일부 데이터에서는 여전히 기존의 SOTA 모형들에 비해서 성능이 많이 떨어지는 모습을 보이기도 했습니다.

GPT-2 논문에서와 마찬가지로 GPT-3 논문에서는 기존의 사전 학습 + 미세 조정 방식의 문제점을 지적합니다. 즉, 특정한 다운스트림 작업에 사전 학습 모형을 적용하기 위해 미세 조정하기 위해서는 해당 작업에 대한 여전히 많은 양의 정답 데이터가 존재해야하는데, 이러한 데이터를 준비하는 것이 쉽지 않다는 것을 언급합니다. 그리고 이러한 한계를 해결하는 것이 여러 가지 이유에서 중요하다고 합니다. 첫 번째로 자연어 처리 관련된 문제 혹은 작업들 중에는 정답 데이터를 준비하는 것이 어려운 경우가 다수 있다고 하면서, 이러한 경우에 추가적인 미세 조정없이 사전 학습 모형을 적용할 수 있는 것이 필요하다고 합니다.

모형의 일반화 가능성이 크게 떨어진다는 문제가 있습니다. 미세 조정을 특정한 데이터셋을 통해서 수행하게 되면 많은 경우, 실제 데이터에 대해서는 모형의 성능이 좋지 않다는 문제가 발생할 수 있습니다.

사람의 경우는 어떠한 언어 관련된 문제에 대한 간단한 설명 (예, 다음 문장이 긍정의 문장인지 부정의 문장인지 맞혀보세요)이나 몇 개의 예를 통해서도 주어진 문제를 쉽게 풀 수 있습니다. 우리가 개발하고자 하는 자연어 처리 시스템도 이러한 능력을 갖는 것이 필요합니다.

해당 논문에서는 대용량의 데이터를 이용해서 사전 학습된 언어 모형의 성능이 모형이 갖는 파라미터수가 많을 수록 좋아지는 점에 착안해서 기존에 제시되었던 모형들 보다 훨씬 많은 수 (즉, 1,750억개)의 파라미터를 갖는 초 거대 언어 모형을 제안합니다.

여러 가지 작업들에 대해서 모형의 성능을 파악합니다. 각 작업에 대해서 서로 다른 세 가지 상황에서의 성능을 파악합니다.

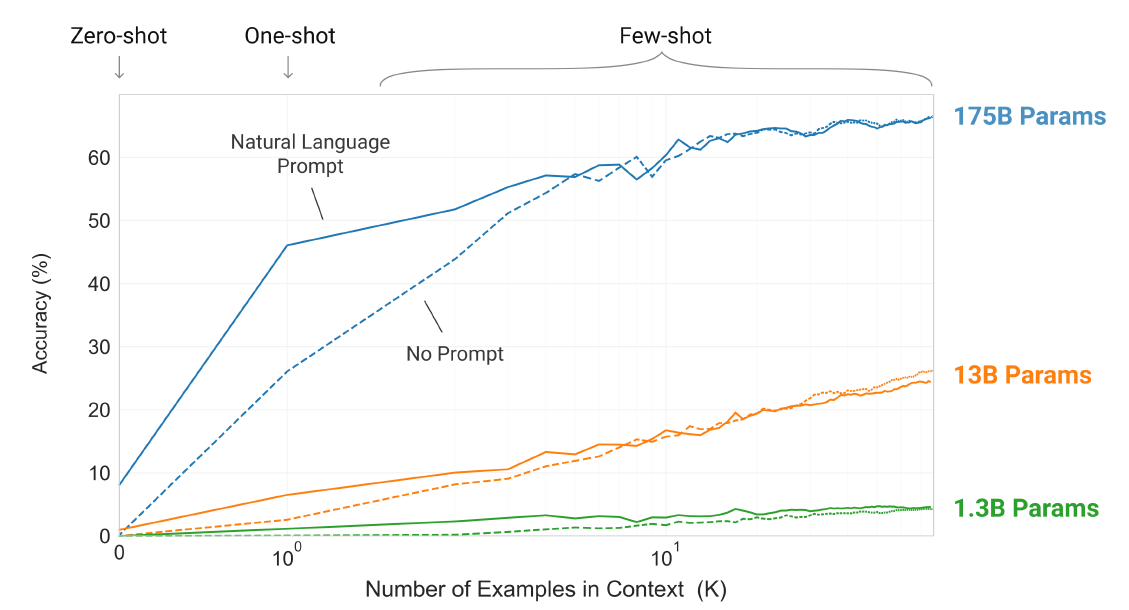
1) “few-shot learning” where we allow as many demonstrations as will fit into the model’s context window (typically 10 to 100)

2) “one-shot learning”,where we allow only one demonstration, and

3) “zero-shot” learning, where no demonstrations are allowed and only an instruction in natural language is given to the model.

미세 조정 방식도 적용할 수 있지만, 해당 논문에서는 미세 조정 방식은 적용하지 않았습니다.

아래 그림은 각 경우에 대한 모형의 일반적인 성능을 보여주고 있습니다. 그리고 모형이 갖는 파라미터의 수에 따라서도 성능 차이가 많이 나는 것을 확인할 수 있습니다.



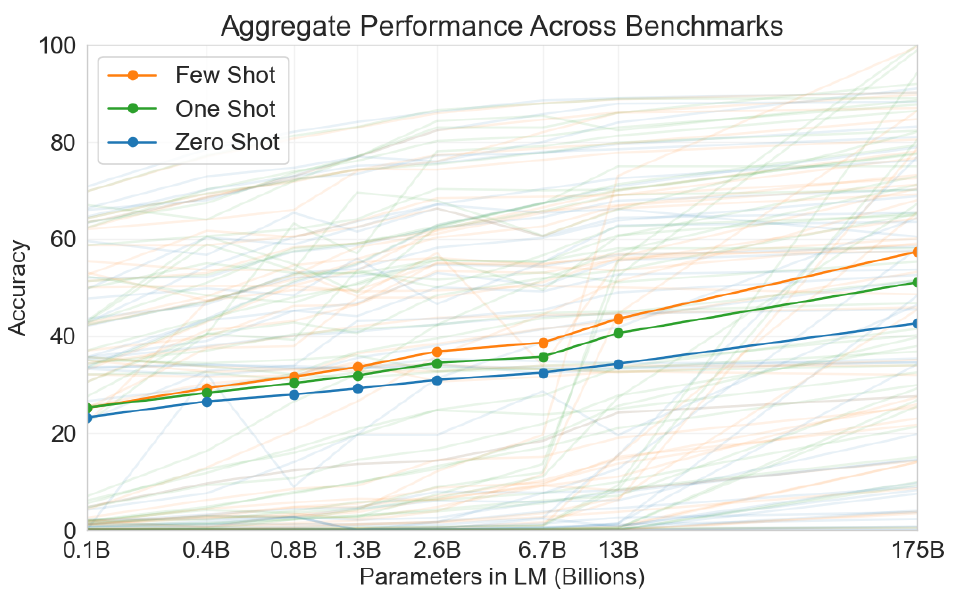
기존 SOTA 모형들과 유사한 성능을 보이거나 경우에 따라서는 더 좋은 성능을 보이는 경우도 있었습니다. For example, GPT-3 achieves 81.5 F1 on CoQA in the zero-shot setting, 84.0 F1 on CoQA in the one-shot setting, 85.0 F1 in the few-shot setting. Similarly, GPT-3 achieves 64.3% accuracy on TriviaQA in the zero-shot setting, 68.0% in the one-shot setting, and 71.2% in the few-shot setting, the last of which is state-of-the-art relative to fine-tuned models operating in the same closed-book setting.

GPT-3는 빠른 적응이 필요하거나 빠른 추론이 필요한 작업/문제(예, 뒤섞인 철자들을 이용해서 원래 단어 맞히기 (unscrambling words) ,숫자 연산 등), 에서도 좋은 성능을 내는 것으로 나타났습니다. GPT-3 also displays one-shot and few-shot proficiency at tasks designed to test rapid adaption or on-the-fly reasoning, which include unscrambling words, performing arithmetic, and using novel words in a sentence after seeing them defined only once.

기사 작성에도 좋은 성능을 보이는 것으로 나타났습니다.

하지만, 성능이 좋지 못한 경우도 있었습니다. At the same time, we also find some tasks on which few-shot performance struggles, even at the scale of GPT-3. This includes natural language inference tasks like the ANLI dataset, and some reading comprehension datasets like RACE or QuAC.

전체적인 모형의 성능



정확도를 이용해서 모형의 성능을 측정할 수 있는 42개의 서로 다른 태스크에서의 모형들의 성능을 보여주고 있습니다. 진한 색 선은 그러한 결과의 평균이라고 생각할 수 있습니다. While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning

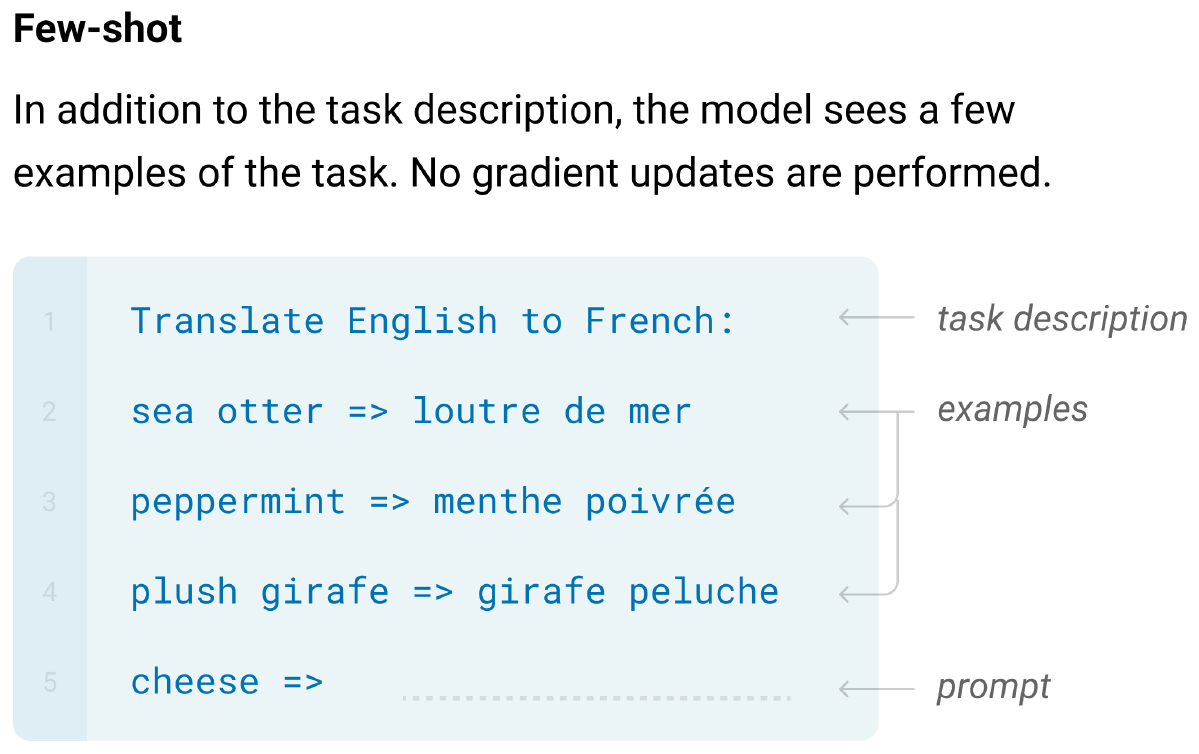
모형의 구조

GPT-3의 모형의 구조는 기본적으로 GPT-2와 유사합니다. 즉, 트랜스포머의 디코더 부분을 사용합니다. 다만, 더 많은 파라미터와 더 많은/다양한 학습 데이터 사용하였고, 학습을 더 많이 했다는 것이 주요한 차이가 됩니다.

추론 상황들

Fine-Tuning (FT) has been the most common approach in recent years, and involves updating the weights of a pre-trained model by training on a supervised dataset specific to the desired task. Typically thousands to hundreds of thousands of labeled examples are used. The main advantage of fine-tuning is strong performance on many benchmarks. The main disadvantages are the need for a new large dataset for every task, the potential for poor generalization out-of-distribution [MPL19], and the potential to exploit spurious features of the training data [GSL+18, NK19], potentially resulting in an unfair comparison with human performance. In this work we do not fine-tune GPT-3 because our focus is on task-agnostic performance, but GPT-3 can be fine-tuned in principle and this is a promising direction for future work.

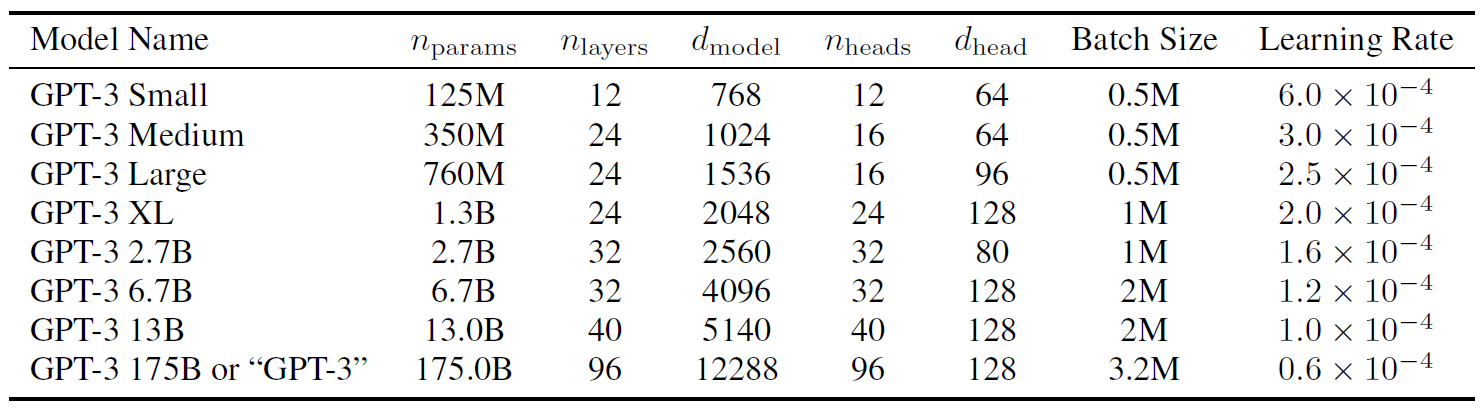
Few-Shot (FS) is the term we will use in this work to refer to the setting where the model is given a few demonstrations of the task at inference time as conditioning [RWC+19], but no weight updates are allowed. As shown in Figure 2.1, for a typical dataset an example has a context and a desired completion (for example an English sentence and the French translation), and few-shot works by giving K examples of context and completion, and then one final example of context, with the model expected to provide the completion. We typically set K in the range of 10 to 100 as this is how many examples can fit in the model’s context window (nctx = 2048). The main advantages of few-shot are a major reduction in the need for task-specific data and reduced potential to learn an overly narrow distribution from a large but narrow fine-tuning dataset. The main disadvantage is that results from this method have so far been much worse than state-of-the-art fine-tuned models. Also, a small amount of task specific data is still required. As indicated by the name, few-shot learning as described here for language models is related to few-shot learning as used in other contexts in ML [HYC01, VBL+16] – both involve learning based on a broad distribution of tasks (in this case implicit in the pre-training data) and then rapidly adapting to a new task.



• One-Shot (1S) is the same as few-shot except that only one demonstration is allowed, in addition to a natural language description of the task, as shown in Figure 2.1. The reason to distinguish one-shot from few-shot and zero-shot (below) is that it most closely matches the way in which some tasks are communicated to humans. For example, when asking humans to generate a dataset on a human worker service (for example Mechanical Turk), it is common to give one demonstration of the task. By contrast it is sometimes difficult to communicate the content or format of a task if no examples are given.

Zero-Shot (0S) is the same as one-shot except that no demonstrations are allowed, and the model is only given a natural language instruction describing the task. This method provides maximum convenience, potential for robustness, and avoidance of spurious correlations (unless they occur very broadly across the large corpus of pre-training data), but is also the most challenging setting. In some cases it may even be difficult for humans to understand the format of the task without prior examples, so this setting is in some cases “unfairly hard”. For example, if someone is asked to “make a table of world records for the 200m dash”, this request can be ambiguous, as it may not be clear exactly what format the table should have or what should be included (and even with careful clarification, understanding precisely what is desired can be difficult). Nevertheless, for at least some settings zero-shot is closest to how humans perform tasks – for example, in the translation example in Figure 2.1, a human would likely know what to do from just the text instruction.

모형 구조



dhead is the dimension of each

attention head. All models use a context window of nctx = 2048 tokens.

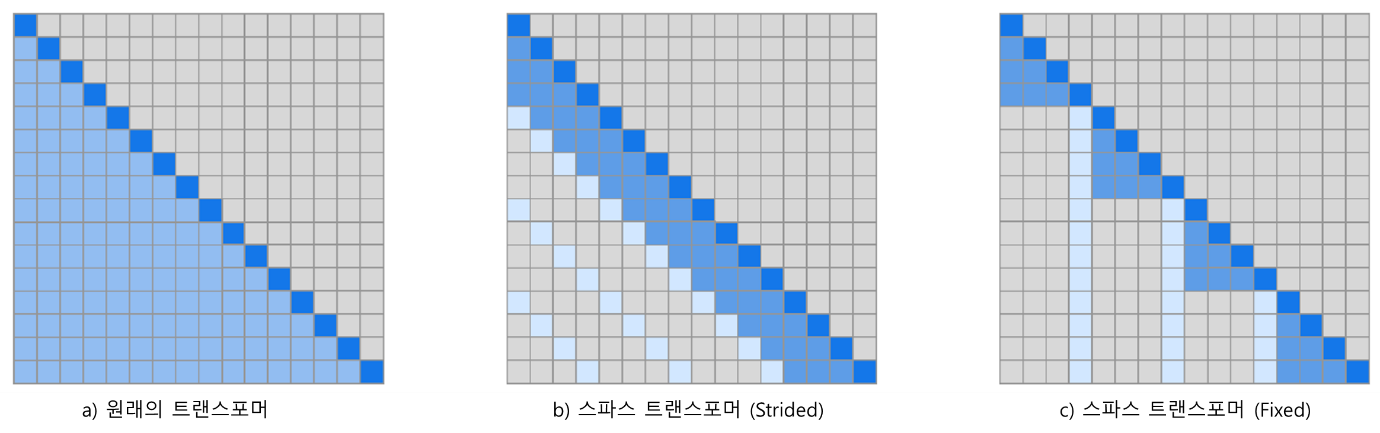
GPT-2와 유사

하지만, 원래의 어텐션 방법 (이를 dense attention이라고 표현합니다)과 sparse attention 방법을 번갈아 가면서 사용합니다. sparse attention은 attention 작업을 수행할 때 모든 다른 토큰들을 사용하는 것이 아니라 일부의 토큰들만 사용하는 것을 의미합니다.

해당 논문에서 사용한 모형의 버전은 그림 XX와 같습니다. 제일 큰 모형 즉, GPT-3 175B를 GPT-3라고 표현하였습니다.

## 참고 시작 ##

Sparse Transformer[[5]](#footnote-5)



두 가지 방식 사용

1) Strided 방법

두 가지 방식으로 attention 작업을 수행하고자 하는 토큰들 선택

{t,t + 1,...,i} for t = max(0,i − l)

여기서 l은 스트라이드를 의미하며, 일반적으로 의 값으로 지정됩니다. 은 토큰의 수를 나타냅니다. 그림 XX에서 입니다. 따라서 해당 그림 가 됩니다. 그리고 는 어텐션 작업의 대상이 되는 토큰의 인덱스를 의미합니다. 그림 XX에서 진한 파란색 ()의 토큰을 나타냅니다. 예를 들어, 여섯 번째 토큰에 대한 어텐션을 수행한다고 가정하는 경우, 가 됩니다. 따라서 이 됩니다. 따라서 어텐션 작업에 사용되는 토큰의 인덱스는 가 됩니다.

두 번째 방법으로는 아래와 같이 어텐션을 수행하고자 하는 다른 토큰들의 인덱스가 결정됩니다.

j : (i−j) mod l = 0 (where j <= i)

즉, 를 로 나눈 나머지의 값이 0인 의 인덱스를 갖는 토큰들이 되는 것입니다. 예를 들어, 인 경우에, 위식을 만족하는 는 1과 5가 됩니다. 위 그림에서 연한 하늘색()으로 표현되었습니다.

2) Fixed 방법

첫번째 방법

여기서 는 floor division을 의미합니다. floor division은 나눈 결과 이하의 정수 중에서 가장 큰 정수를 의미합니다. 예를 들어 5/4는 1.25가 됩니다. 이숫자의 floor는 1이 됩니다. =1이 되는 것입니다.

예를 들어, I = 3이라고 하는 경우, 이 되기 때문에 위 식을 만족하는 는 0, 1, 2, 3이 됩니다.

두번째 {j : j mod l ∈ {t, t + 1,...,l}, where t = l − c and c is a hyperparameter.

예를 들어 c=1 이라고 하겠습니다. 그러현 경우, t=4-1=3이 됩니다. 따라서, j를 4로 나눈 나머지가 {3, 4}에 속하는 j가 됩니다. 예를 들어, i=10이라고 하는 경우, j <= i이면서 위 식을 만족하는 j는 3과 7이 됩니다. 이는 i>=7인 토큰에 대해 공통적으로 적용됩니다. 위 그림에서 옅은 하늘색 ()에 해당하는 토큰들이 됩니다.

## 참고 끝 ##

학습 데이터

GPT-3에서는 기본적으로 CommonCrawl 데이터셋[[6]](#footnote-6)을 사용합니다. 학습 당시 해당 데이터셋에는 1조개의 단어들이 포함되어 있으며, 이는 가장 큰 GPT-3 모형을 같은 입력 시퀀스 데이터를 두 번 이상 사용하지 않고도 학습시킬 수 있는 정도의 데이터양이라고 합니다. 하지만, 원본 Common Crawl 데이터셋의 경우, 질적인 문제가 있어 해당 데이터셋을 그대로 사용하지 않고 질을 높이기 위한 추가적인 작업을 수행한 데이터셋을 최종 학습 데이터로 이용합니다.

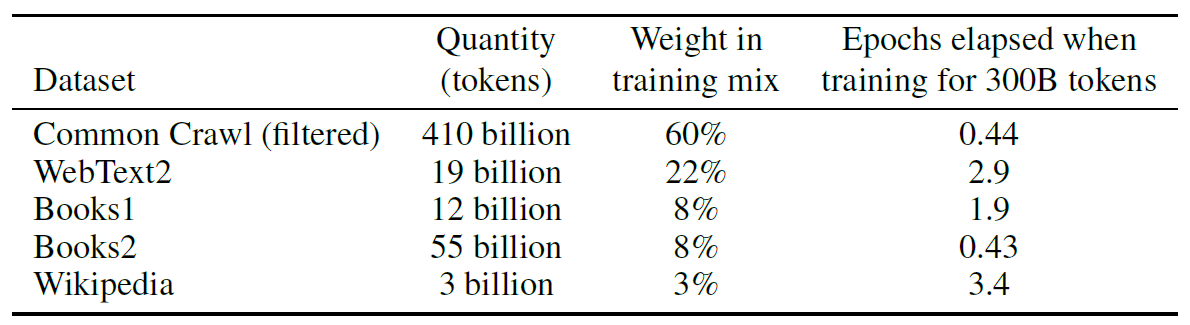
1) 양질의 참조 데이터와의 유사도를 기반으로 원본 데이터를 필터링 합니다.

2) 중복이 되는 데이터를 제거합니다.

3) 양질의 다른 데이터셋을 추가적으로 사용합니다.

an expanded version of the WebText dataset, two internet-based books corpora (Books1 and Books2) and English-language Wikipedia.

Note that during training, datasets are not sampled in proportion to their size, but rather datasets we view as higher-quality are sampled more frequently, such that CommonCrawl and Books2 datasets are sampled less than once during training, but the other datasets are sampled 2-3 times.



예를 들어 300B의 토큰을 학습하는 과정에서 Wikipedia 데이터셋은 3.4번 사용된 반면, CommonCrawl 데이터셋은 0.44번 밖에 사용되지 않았습니다.

많은 양의 인터넷 데이터를 학습 데이터로 사용하는 경우 발생할 수 있는 문제 => 다운스트림 작업의 오염, 평가 데이터의 일부가 학습 데이터에 포함될 수 있는 가능성 존재 => 이러한 문제를 해결하기 위해 되도록 중복이 되는 데이터 제거

평가

퓨샷 상황

특정 다운스트림 태스크의 학습 데이터에서 K의 예제를 랜덤하게 추출하여 입력 데이터의 일부로 사용합니다. K의 값은 0 부터 모형의 문맥 윈도우가 허용하는 범위 내에서의 최대값을 취할 수 있습니다. the model’s context window, which is nctx = 2048 일반적으로 10개에서 100개의 예를 사용했습니다.

그리고 어떠한 다운스트림 작업에 대해서는 위의 예제와 더불어 프롬프트를 사용하기도 했습니다. 프롬프트는 그림 XX에서 볼 수 있듯이 입력 데이터의 일부로 사용되는 풀고자 하는 문제에 대한 텍스트라고 생각할 수 있습니다. 예를 들어, 한글을 영어로 번역하고자 하는 경우, 번역하고자 하는 한글 텍스트가 프롬프트가 될 수 있는 것입니다. 예, 금요일을 영어로 번역하고자 하는 경우의 프롬프트의 예: ‘금요일 =>’

On tasks that involve choosing one correct completion from several options (multiple choice), we provide K examples of context plus correct completion, followed by one example of context only, and compare the LM likelihood of each completion.

여러가지 작업들에 대해 모형의 성능을 파악

주어진 텍스트를 바탕으로 다음을 예측하여 생성하는 언어모델 성능 평가에서 GPT-3가

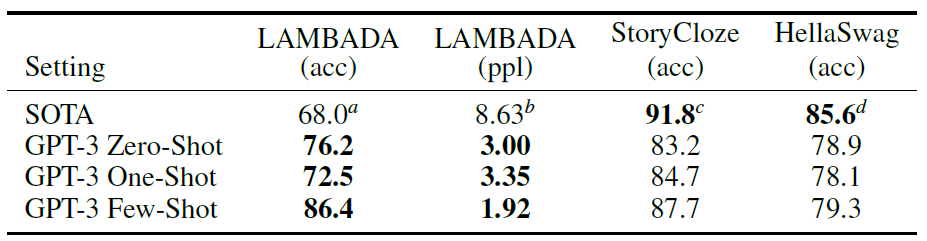
SOTA를 달성했고 (Lambada 등), 광범위한 지식에 대해 문맥 없이 closed book으로 답변을

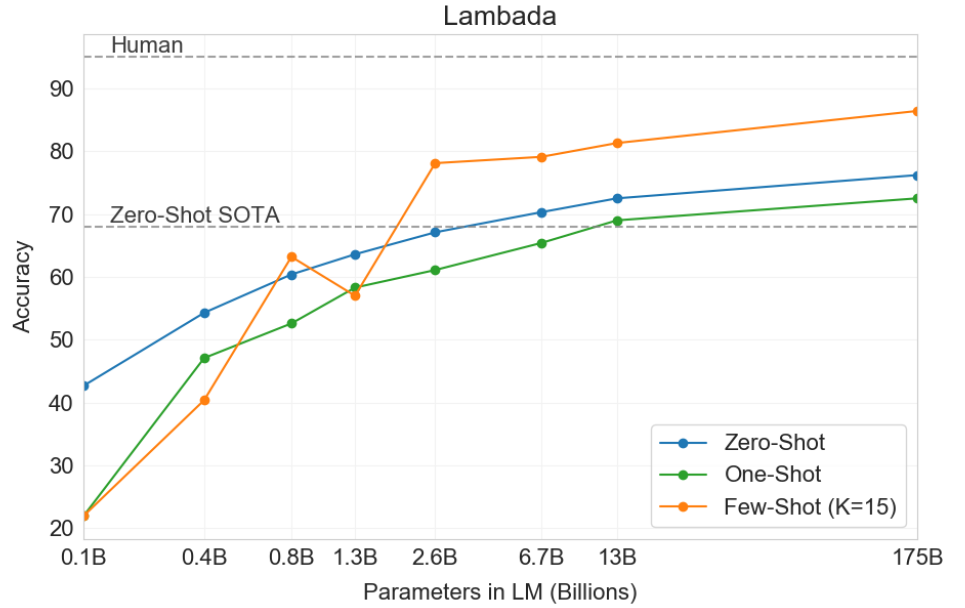
생성하는 task(TriviaQA)에서도 기존의 미세조정 SOTA 모델을 제치고 SOTA를 달성하였습니다.

물리학적인 이해와 세상에 대한 전반적인 이해를 묻는 상식추론(PhysicalQA)에서도 SOTA를

달성했고, 자유형식 대화에서 독해력을 평가하는 기계독해(CoQA)에서는 인간에 가까운 성능을

보였습니다.





The HellaSwag dataset [ZHB+19] involves picking the best ending to a story or set of instructions. The examples were

adversarially mined to be difficult for language models while remaining easy for humans (who achieve 95.6% accuracy).

GPT-3 achieves 78.1% accuracy in the one-shot setting and 79.3% accuracy in the few-shot setting, outperforming the

75.4% accuracy of a fine-tuned 1.5B parameter language model [ZHR+19] but still a fair amount lower than the overall

SOTA of 85.6% achieved by the fine-tuned multi-task model ALUM.

3.1.4 StoryCloze

We next evaluate GPT-3 on the StoryCloze 2016 dataset [MCH+16], which involves selecting the correct ending

sentence for five-sentence long stories. Here GPT-3 achieves 83.2% in the zero-shot setting and 87.7% in the few-shot

setting (with K = 70). This is still 4.1% lower than the fine- tuned SOTA using a BERT based model [LDL19] but

improves over previous zero-shot results by roughly 10%.

하지만, 번역(Multi-BLEU)과 자연어추론(ANLI Round3)에서는 좋지 않은 성능을 보였습니다.

번역은 특히 ‘영어→타언어’로의 번역 성능이 좋지 못했고, 자연어추론은 few-shot setting을

제외하면 찍기 수준의 성능밖에 발휘하지 못하는 한계가 있었습니다.

Limitations

논문 보고 다시 작성할 것

## 20.4 InstructGPT

InstructGPT was released on 27 January, 2022. Using GPT-3 as its base model, GPT-3.5 models use the same pre-training datasets as GPT-3, with additional fine-tuning.

InstructGPT의 가장 큰 특징 혹은 이전 버전의 GPT 모형들과 갖는 가장 큰 차이는 학습 과정에서사람의 피드백을 사용했다는 것입니다. 이는 기존 모형들이 alignment가 잘 안된다는 문제를 해결하기 위해서 입니다. 즉, 결과물이 사람들의 기대나 예상 혹은 의도와 다르다는 문제를 해결하기 위한 것입니다.

주요 작동 과정은 세 단계로 구분됩니다. InstructGPT는 GPT 3.5 series를 사전 학습 모형으로 사용했습니다. 그 중에서 text-davinci-03 모형을 사용했을 것으로 예상됩니다.

1) Supervised Fine-Tuning

양질의 데이터셋을 구축하기 위해서 40명의 labeler 사용

프롬프트에 대해서 사람들이 답변 작성

프롬프트의 경우, 일부는 사람들이 직접 생성하였고, 일부는 OpenAI 에서 제공하는 GPT-3 기반의 API에 입력된 프롬프츠를 사용했습니다.

데이터의 다양성을 증가시키기 위해 레이블러별로 200개의 프롬프트와 그에 대한 답변 데이터를 사용

이렇게해서 12K – 15K 정도의 정답 데이터를 구축

이를 이용해서 사전 학습된 모형에 대해 미세 조정 진행

Due to the limited amount of data for this step, the SFT model obtained after this process is likely to output text which is still (probabilistically) not very user-attentive and generally suffers from misalignment, in the sense explained in the above sections. The problem here is that the supervised learning step suffers from high scalability costs. To overcome this problem, instead of asking human labelers to create a much bigger curated dataset, a slow and costly process, the strategy is now to have the labelers rank different outputs of the SFT model to create a reward model –let’s explain this in more detail in the following section.

2) 보상 모형 (reward model) 사용

선택되어진 일부의 프롬프트에 대해서 1)의 단계에서 반환된 4 – 9 개의 결과 사용

이를 각 레이블러들이 best에서 worst의 순으로 순위를 매김

=> 이것이 새로운 정답 데이터셋, where rankings are the labels. 이 데이테섯의 크기는 1단계에서 사용된 데이터셋 보다 10배 정도 큼

그리고 이러한 정답 데이터는 the reward model을 학습하는데 사용됨

데이터 입력의 형태

- 입력 데이터: series of prompts & responses

- 출력 결과: 예상 리워드 값 (or ranking)

This new data is used to train a reward model (RM). This model takes as input a few of the SFT model outputs and ranks them in order of preference.

이를 이용하여 학습?

보상 모형의 goal is to learn an objective function (the reward model) directly from the data. The purpose of this function is to give a score to the SFT model outputs, proportional to how desirable these outputs are for humans.

3) Reinforcement Learning 모형

단계2를 이용해서 학습된 policy를 이용해서 입력된 프롬프트에 대해서 응답 출력

그리고 출력된 응답에 따라서 reward 값이 결정 (혹은 계산됨)

모델은 이 값을 이용해서 policy update

Proximal Policy Optimization

Policy 업데이트를 위해 PPO 방법 사용

이는 on-policy 방법 => it learns from and updates the current policy directly => (agent의 행동에따라) 지속적으로 policy를 update한다는 것을 의미

PPO uses a value function to estimate the expected return of a given state or action. The value function is used to compute the advantage function, which represents the difference between the expected return and the current return. The advantage function is then used to update the policy by comparing the action taken by the current policy to the action that would have been taken by the previous policy. This allows PPO to make more informed updates to the policy based on the estimated value of the actions being taken.

In this step, the PPO model is initialized from the SFT model, and the value function is initialized from the reward model. The environment is a bandit environment which presents a random prompt and expects a response to the prompt. Given the prompt and response, it produces a reward (determined by the reward model) and the episode ends. A per-token KL penalty is added from the SFT model at each token to mitigate over optimization of the reward model.

From the paper

언어 모형의 크기를 크게 하는 것이 무조건적으로 사용자의 의도에 맞는 결과를 보장하는 것은 아니다. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not aligned with their users.

본 논문에서는 사람의 피드백을 이용한 미세 조정 방법을 사용해서 사용자의 의도에 맞는 결과를 반환하는 언어 모형을 제안하고자 한다.

미세 조정을 위해서는 사전학습된 GPT-3을 사용하였습니다.

We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback.

모델명: InstructGPT

We call the resulting models InstructGPT. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters.

평가

사용자의 의도에 맞는 결과를 제대로 반환하는지를 평가하기 위해 평가 항목으로 Truthfulness 등을 사용

1. Introduction

GPT-3는 몇 개의 예제와 프롬프트만 있어도 언어 모형은 다양한 자연어 처리 문제를 수행할 수 있다는 것을 보여주었습니다. 하지만, 여전히 문제가 있습니다. 사실을 조작한다던지, 편향된 혹은 악의적인 텍스트를 생성한다는지, 또는 사용자의 지시를 따르지 않는 것 등이 그러한 문제의 예입니다.

이러한 이유 중 하나는 인터넷에 존재하는 데이터를 이용해서 학습되는 언어 모형의 목적이 “사용자의 지시를 안전하고 도움이 될 수 있게 수행하라’는 것과 차이가 있기 때문입니다. 즉, 언어 모형의 목적은 사용자의 지시 또는 의도와 misaligned 되어 있습니다.

해당 논문에서는 사용자의 의도와 일치하는 방향으로 행동하는 언어 모형을 제안하고자 합니다. 여기서의 사용자의 의도는 사용자의 지시와 같은 명시적인 의도만을 의미하는 것이 아니라 믿을 수 있고, 편향되거나 악의적이지 않으며 해롭지 않은 결과와 같은 암묵적인 의도도 포함됩니다.

이를 위해서 해당 논문에서는 GPT-2, GPT-3와 달리 미세 조정 방법에 집중합니다. 특히, 사람들의 피드백을 이용한 강화 학습 방법을 사용하여 미세 조정을 수행합니다.

이 방법은 사용자 선호를 미세 조정을 위한 보상 신호 (reward signal)로 사용합니다.

학습 방법

InstructGPT는 기본적으로 거대 사전학습 언어 모형을 시작점으로 사용합니다. 해당 논문에서는 GPT-3를 이러한 사전학습 모형으로 사용했습니다.

그리고 세 단계를 거쳐 미세 조정을 수행합니다.

Step 1: Collect demonstration data, and train a supervised policy.

Step 2: Collect comparison data, and train a reward model.

Step 3: Optimize a policy against the reward model using PPO.

Steps 2 and 3 can be iterated continuously; more comparison data is collected on the current best policy, which is used to train a new RM and then a new policy. In practice, most of our comparison data comes from our supervised policies, with some coming from our PPO policies.

Data

대부분의 프롬프트 데이터는 OpenAI API에 제출되었던 프롬프트들을 이용해서 준비

We heuristically deduplicate prompts by checking for prompts that share a long common

prefix, and we limit the number of prompts to 200 per user ID. We also create our train, validation,

and test splits based on user ID, so that the validation and test sets contain no data from users whose

data is in the training set. To avoid the models learning potentially sensitive customer details, we

filter all prompts in the training split for personally identifiable information (PII).

뿐만 아니라, 레이블러들이 직접 프롬프트를 생산하기도 했음

레이블러들이 생성한 프롬프트는 크게 세 가지 종류

• Plain: We simply ask the labelers to come up with an arbitrary task, while ensuring the

tasks had sufficient diversity.

• Few-shot: We ask the labelers to come up with an instruction, and multiple query/response

pairs for that instruction.

• User-based: We had a number of use-cases stated in waitlist applications to the OpenAI

API. We asked labelers to come up with prompts corresponding to these use cases.

이러한 프롬프트를 이용해서 세 개의 데이터셋을 구축 => From these prompts, we produce three different datasets used in our fine-tuning procedure: (1) our

SFT dataset, with labeler demonstrations used to train our SFT models, (2) our RM dataset, with

labeler rankings of model outputs used to train our RMs, and (3) our PPO dataset, without any human

labels, which are used as inputs for RLHF fine-tuning. The SFT dataset contains about 13k training

prompts (from the API and labeler-written), the RM dataset has 33k training prompts (from the API

and labeler-written), and the PPO dataset has 31k training prompts (only from the API). More details

on dataset sizes are provided in Table 6.

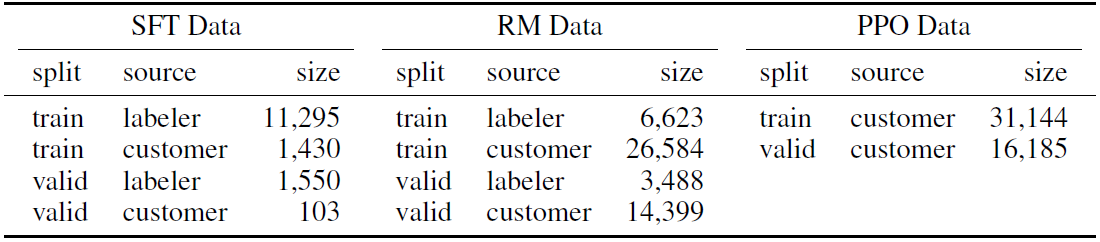
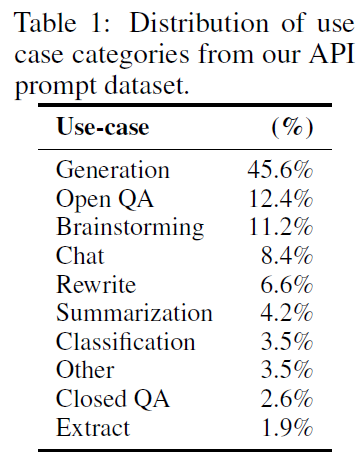


Table 6: Dataset sizes, in terms of number of prompts.



Most of the use-cases have are generative, rather than classification or QA.

3.3 Tasks

Our training tasks are from two sources: (1) a dataset of prompts written by our labelers and (2) a

dataset of prompts submitted to early InstructGPT models on our API (see Table 6). These prompts

are very diverse and include generation, question answering, dialog, summarization, extractions, and

other natural language tasks (see Table 1). Our dataset is over 96% English, however in Section 4.3

we also probe our model’s ability to respond to instructions in other languages and complete coding

tasks. 96%가 영어 데이터

휴먼 코더들을 이용한 데이터 생성

In each case, we ask our labelers to do their best to infer the intent of the user who wrote the prompt, and ask them to skip inputs where the task is very unclear. Moreover, our labelers also take into account the implicit intentions such as truthfulness of the response, and potentially harmful outputs such as biased or toxic language, guided by the instructions we provide them (see Appendix B) and their best judgment.

3.4 Human data collection

레이블러 채용

To produce our demonstration and comparison data, and to conduct our main evaluations, we hired

a team of about 40 contractors on Upwork and through ScaleAI. Compared to earlier work that

collects human preference data on the task of summarization (Ziegler et al., 2019; Stiennon et al.,

2020; Wu et al., 2021), our inputs span a much broader range of tasks, and can occasionally include

controversial and sensitive topics. Our aim was to select a group of labelers who were sensitive to the

preferences of different demographic groups, and who were good at identifying outputs that were

potentially harmful. Thus, we conducted a screening test designed to measure labeler performance

on these axes. We selected labelers who performed well on this test; for more information about our

selection procedure and labeler demographics, see Appendix B.1.

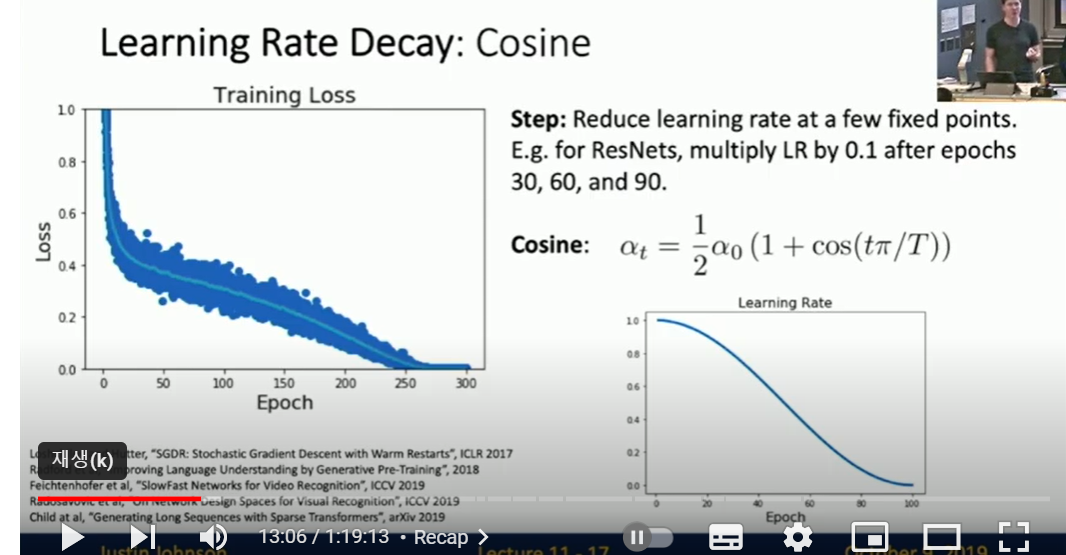
3.5 Models

GPT-3를 사전학습 모형으로 사용

세 가지 다른 미세 조정 방법을 이용해서 학습 진행

1) Supervised fine-tuning (SFT).

We fine-tune GPT-3 on our labeler demonstrations using supervised learning. We trained for 16 epochs, using a cosine learning rate decay, and residual dropout of 0.2.



<https://www.youtube.com/watch?v=WUazOtlti0g>

2) Reward modeling (RM). Starting from the SFT model with the final unembedding layer removed,

we trained a model to take in a prompt and response, and output a scalar reward.

In order to speed up comparison collection, we present labelers with anywhere between K = 4 and

K = 9 responses to rank. This produces comparisons for each prompt shown to a labeler.

하나의 프롬프트에 대해서 두 개의 response 존재 => 사람들이 순위를 매김

이를 이용해서 RM 모형이 각 패어에 대해서 reward scalar 값을 예측

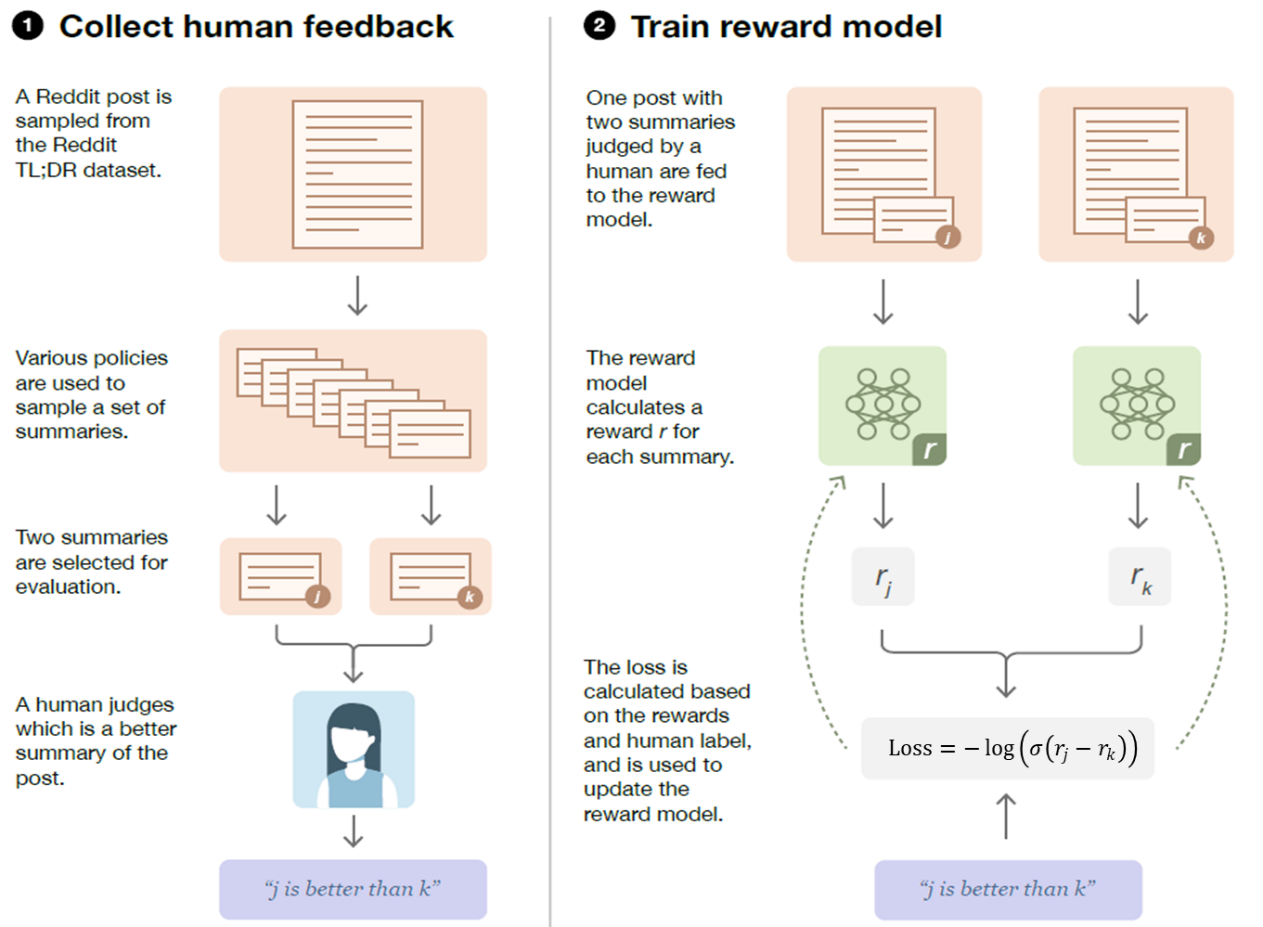
순위가 높은 경우 (즉, 사람에게 더 선호되는 response의 경우) 더 큰 값이 예측되어야 함

이러한 과정은 아래와 같이 표현될 수 있음

이러한 페어에 대한 비용함수는 아래와 같이 표현됩니다.

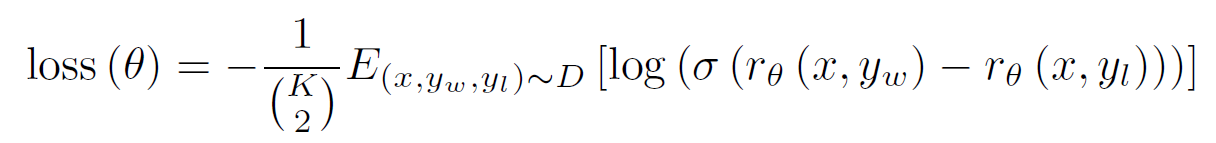
사람의 관점에서는 응답 j가 응답 k 보다 더 우수합니다. 는 보상 모형이 예측하는 응답 j에 대한 보상 점수가 됩니다.

보상 모형이 예측을 잘 한다면 의 값이 보다 크게 나와야 합니다. 위 식에서 는 시그모이드 함수입니다. 시그모이드 함수는 0과 1 사이의 값을 반환합니다. 만약 모형이 예측을 잘못해서 가 된다면, 는 음수가 될 것이고, 이러한 경우 시그모이드 함수는 0에 가까운 값을 출력합니다. 그리고 그에 대한 로그값은 더 작아지게 됩니다. 따라서 비용의 값이 커지게 됩니다. 반대로 예측을 잘할수록 비용의 값이 작아지게 됩니다.



의 각 페어를 하나의 관측치로 간주하게 되면 과적합이 발생하기 때문에 해당 논문에서는 에 해당하는 전체 페어를 하나의 관측치로 간주하고 한 번에 학습하는 방법을 사용하였습니다.

보상 모형의 비용함수는 아래와 같습니다.



는 입력 프롬프트 와 그에 대한 응답 에 대해 보상 모형이 출력하는 보상 점수가 됩니다. 그리고 는 보상 모형 (즉, 신경망 모형)이 갖는 파라미터가 됩니다. 그리고 와 는 프롬프트 x에 대한 두 개의 응답을 나타내며, 둘 중 레이블러에게 더 좋은 응답이라고 평가된 것이 입니다. 즉, 는 winner를 은 loser를 뜻한다고 생각할 수 있습니다. 보상 모형은 위의 비용함수를 최소화하는 방향으로 파라미터의 값을 학습하게 됩니다.

3) Reinforcement learning (RL). Once again following Stiennon et al. (2020), we fine-tuned the SFT model on our environment using PPO (Schulman et al., 2017).

우리는 이전 단계에서 학습된 보상 모형을 이용해서 사람의 기준에서 질이 좋은 답변을 생산하는 policy (즉, 신경망 모형)을 학습하고자 합니다.

이를 RL을 이용해서 수행합니다.

보상 모형이 출력하는 결과 (즉, 리워드 점수)를 전체 요약 (summary) 혹은 전체 응답에 대한 리워드라고 간주합니다. 우리는 이러한 리워드를 PPO를 이용해서 최대화하고자 합니다. 즉, 최대화하는 방향으로 policy의 파라미터를 업데이트합니다.

각 time step은 하나의 BPE 토큰입니다. 각 타임 스텝에서 policy를 이용해서 토큰을 출력합니다 (이것이 해당 타임 스텝에서의 행동이 되는 것입니다). 그리고 이러한 행동의 결과로 전체 응답이 생성됩니다 (이것을 하나의 에피소드라고 간주할 수 있습니다). 그리고 이러한 응답에 대해 보상 모형이 반환하는 보상 점수를 응답 (여러 개의 토큰으로 구성된)에 대한 보상으로 간주하고, 우리는 이 보상을 최대화하는 방향으로 policy를 업데이트하고자 하는 것입니다.

초기 policy를 Reddit TL;DR 데이터셋을 이용해서 학습된 모형으로 합니다 (We initialize our

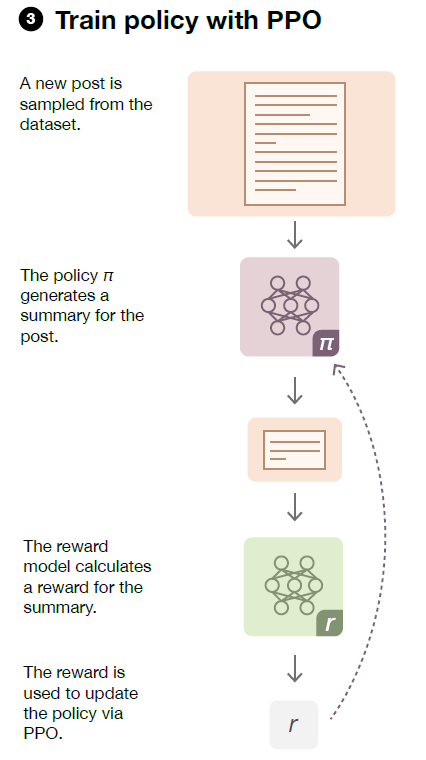
policy to be the model fine-tuned on Reddit TL;DR.)

보상 모형이 계산하는 보상을 그대로 사용하지 않고, KL 기반의 페날티를 추가합니다. KL divergence between 학습된 RL policy with parameter and the original supervised model

그럼 전체 보상은 아래와 같이 표현됩니다.

위 식에서 이 KL divergence가 됩니다.

KL을 사용하게 되면 policy가 더 다양한 값들을 탐색하게 되며, 그리고 보상 모형이 학습 동안 관찰한 결과와 너무 다르지 않은 결과를 해당 policy가 학습하지 않는 효과가 있다고 합니다.

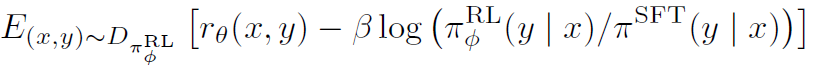


프롬프트와 응답 존재

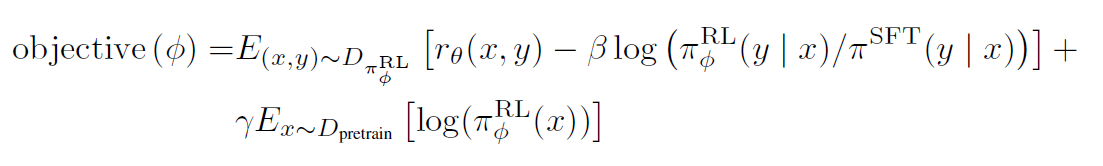
환경은 보상을 생산, 보상은 보상 모형에 의해 결정, 그리고 에피소드가 종료 (즉, 응답이 리턴)

In addition, we add a per-token KL penalty from the SFT model at each token to mitigate overoptimization of the reward model. The value function is initialized from the RM. We call these models “PPO.”

기본 objective는 다음과 같이 표현



하지만, 해당 논문에서는 추가적인 텀을 사용



We call these models “PPO-ptx.”

<https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286>

In the final stage, the model is presented with a random prompt and returns a response. The response is generated using the ‘policy’ that the model has learned in step 2. The policy represents a strategy that the machine has learned to use to achieve its goal; in this case, maximizing its reward. Based on the reward model developed in step 2, a scaler reward value is then determined for the prompt and response pair. The reward then feeds back into the model to evolve the policy.

In 2017, Schulman et al. introduced Proximal Policy Optimization (PPO), the methodology that is used in updating the model’s policy as each response is generated. PPO incorporates a per-token Kullback–Leibler (KL) penalty from the SFT model. The KL divergence measures the similarity of two distribution functions and penalizes extreme distances. In this case, using a KL penalty reduces the distance that the responses can be from the SFT model outputs trained in step 1 to avoid over-optimizing the reward model and deviating too drastically from the human intention dataset.

Steps 2 and 3 of the process can be iterated through repeatedly though in practice this has not been done extensively.

ptx 모형 관련해서 <https://towardsdatascience.com/the-new-version-of-gpt-3-is-much-much-better-53ac95f21cfb>를 참고할 것

## 참고 시작 ##

강화학습

policy는 주어진 상태에 대해 특정 행동을 반환하는 함수라고 생각할 수 있습니다. policy 기반의 방법에서 이러한 함수는 여러 개의 파라미터들로 정의됩니다. 하나의 신경망 모형이 하나의 policy가 될 수 있는 것입니다. InstructGPT에서는 SFT 모형을 policy로 사용하였다고 생각할 수 있습니다. 그리고 이러한 모형이 갖는 파라미터를 행동을 통해 얻을 수 있는 보상을 최대화하는 방향으로 업데이트합니다. 이러한 방법을 policy gradient 방법이라고 합니다.

전체 보상을 maximize하게끔 policy를 update

policy는 확률 분포를 결과로 반환

각 행동들에 대한 확률 분포

언어 모형의 경우, 취할 수 있는 행동 => 전체 토큰

행동 => 전체 토큰 중 하나의 토큰을 선택하는 것

policy는 즉, 모형은 각 토큰을 취할 확률은 반환

1. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. [↑](#footnote-ref-1)
2. Semantic similarity (or paraphrase detection) tasks involve predicting whether two sentences are semantically equivalent or not. [↑](#footnote-ref-2)
3. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9. [↑](#footnote-ref-3)
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901. [↑](#footnote-ref-4)
5. Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*. [↑](#footnote-ref-5)
6. https://commoncrawl.org/the-data/ [↑](#footnote-ref-6)