Generative AI
: GPT-1, GPT-2

MILab Undergraduate student, Kim Taehyeon

2023. 08. 10

목차

1. GPT 모델

- 생성형 AI 란?

- 모델의 발전

2. GPT-1 모델

3. GPT-2 모델

4. 마무리

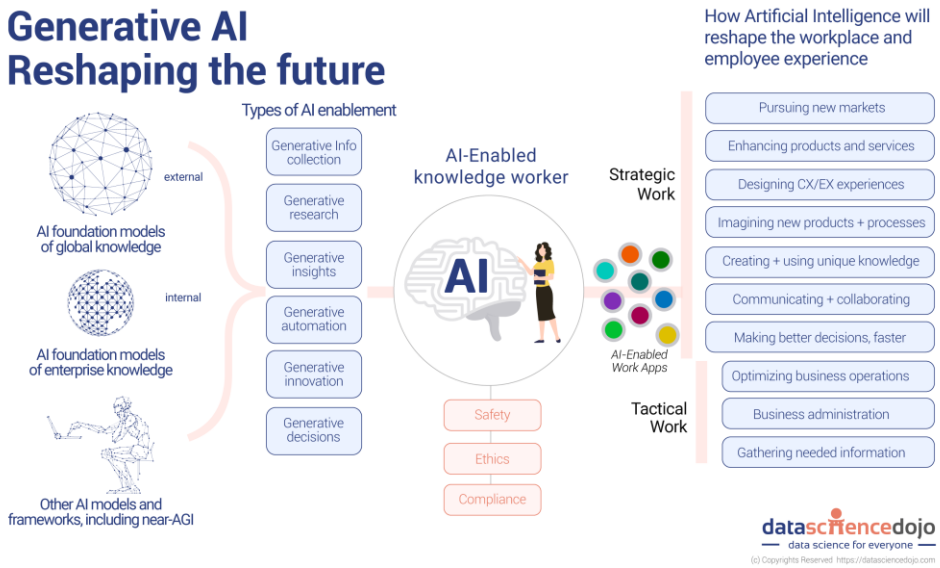
+) Reference

1. GPT 모델

1. 생성형 AI 란

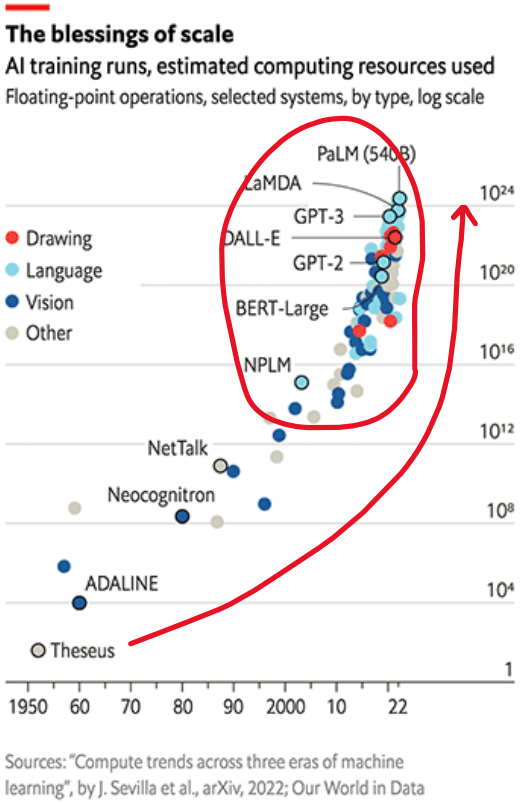
생성형 AI (Generative AI) 는 비정형 데이터와 딥러닝 모델을 사용하여 사용자 입력을 기반으로 콘텐츠를 생성하는 인공지능이다.

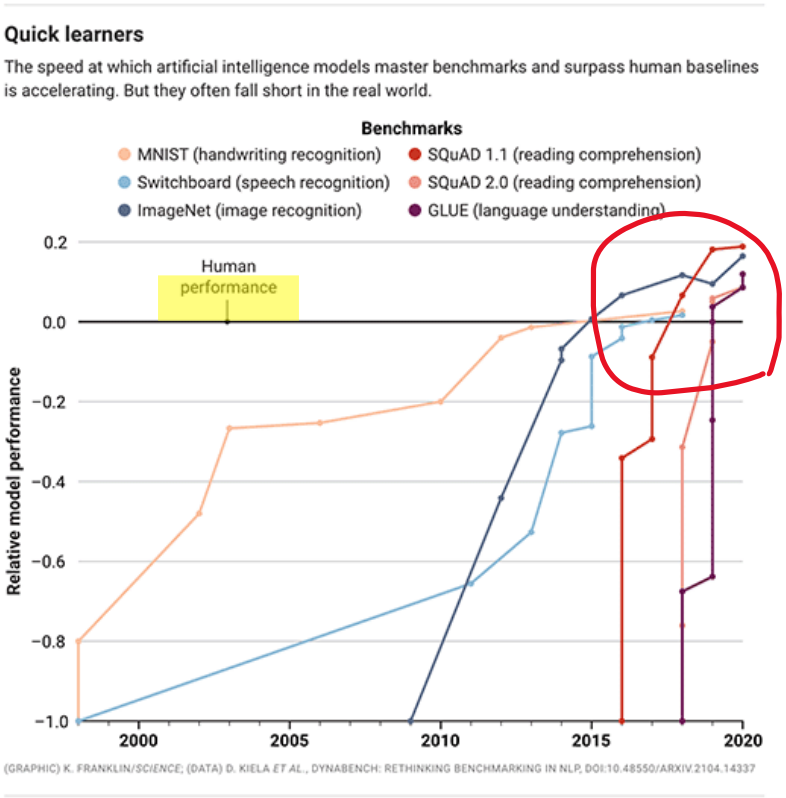
예를 들어, **ChatGPT** 에 질문을 입력하면 간단하지만 합리적이고 상세한 답변을 제공해주고, 후속 질문에 대하여 대화 초기의 내용을 기반으로 답변이 가능하다.



1. GPT 모델

2. 모델의 발전



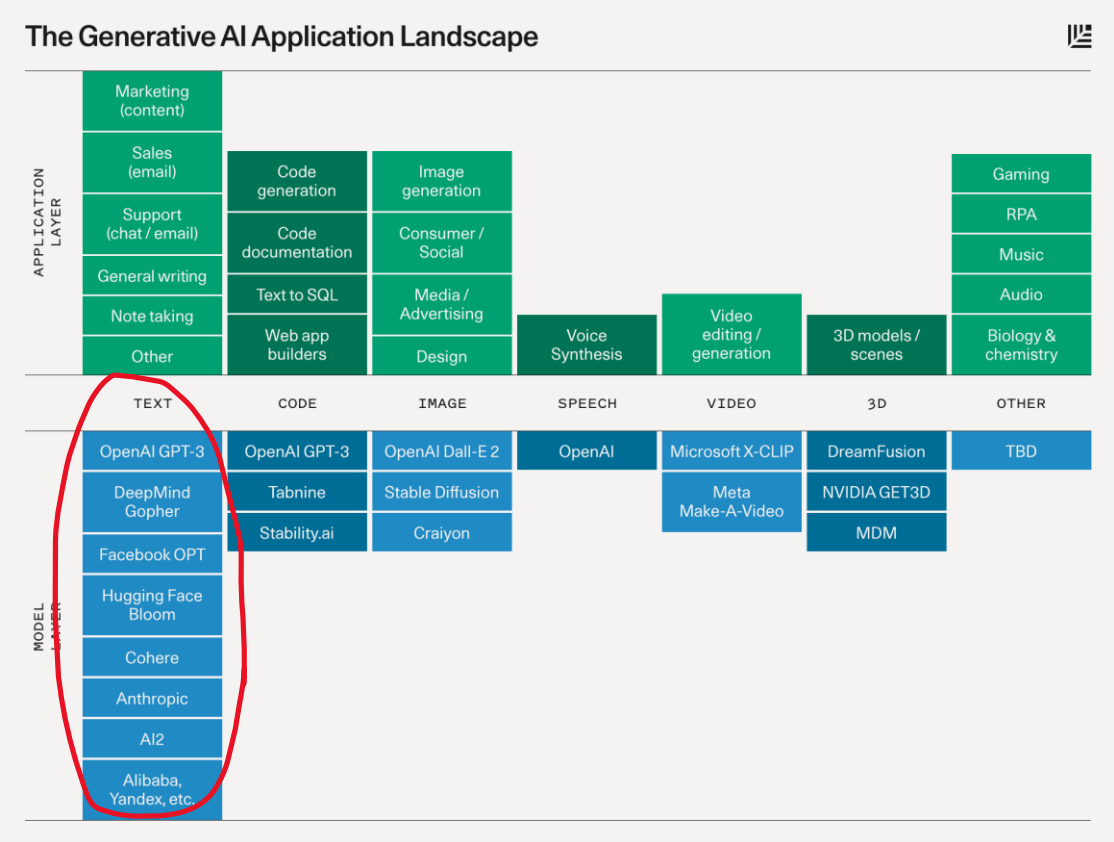


현재 방대한 정보의 양을 바탕으로 대부분 모델의 성능이 비약적으로 발전함.

이러한 상황이 생성형 AI 의 급발전이 가능하게 된 초석이 되었음.

1. GPT 모델

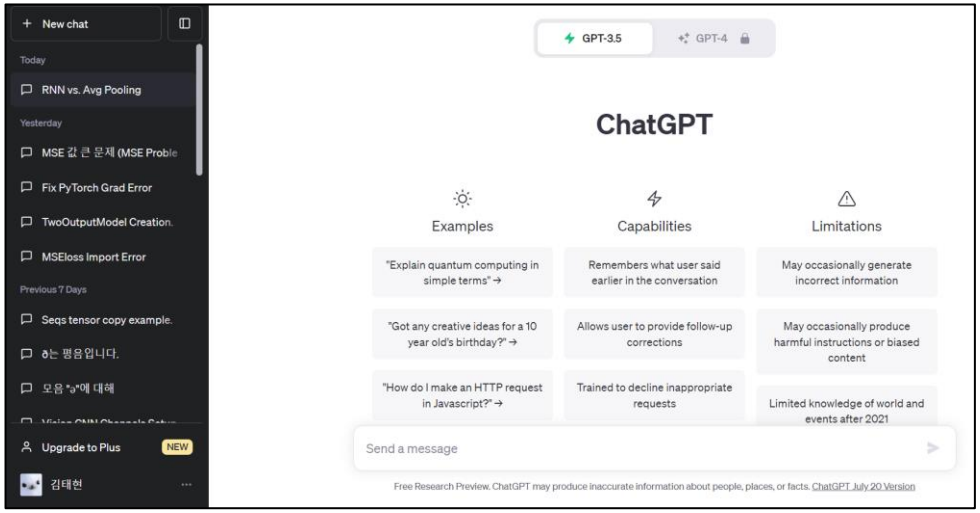
2. 모델의 발전

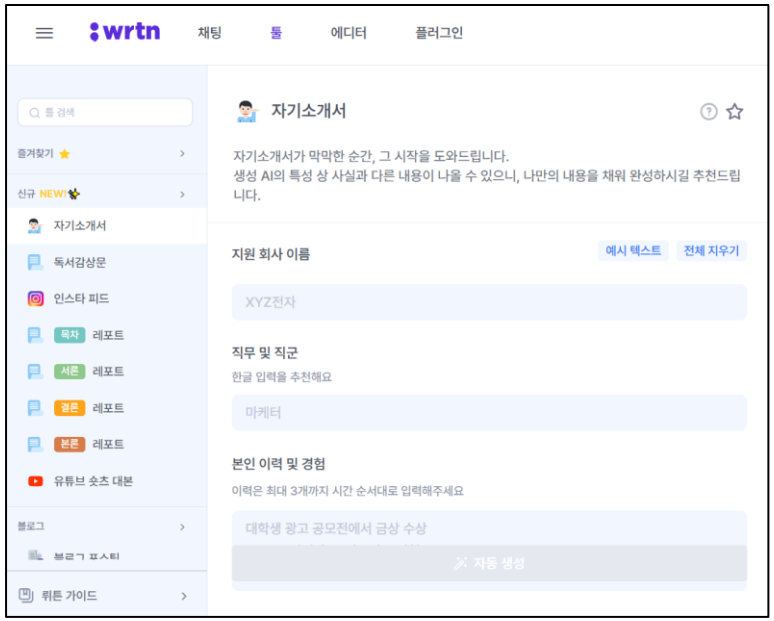


텍스트 관련 생성형 AI 의 발전이 많이 이루어졌음.

1. GPT 모델

2. 모델의 발전





사람들의 관심을 끌면서 생성형 AI 의 기대감은 한창 높아짐.

인간의 일자리를 대체할 수도 있겠다는 불안감까지 조성할 정도였음.

1. GPT 모델

2. 모델의 발전 (Generative Pre-trained Transformer)

구분	GPT-1	GPT-2	GPT-3	GPT-4
발표 년도	2018년	2019년	2020년	2023년
파라미터 수	1.17억 개	15억 개	1.75조 개	10조 개 이상
학습 데이터	영어 위키피디아와 뉴스 기사	인터넷에 있는 거의 모든 웹페이지	인터넷에 있는 거의 모든 웹페이지	멀티 모달 데이터
특징	대규모 언어 모델의 초석	GPT-1 보다 더 많은 파라미터와 데이터를 사용	GPT-2 보다 더 많은 파라미터와 데이터를 사용	다양한 자연어 처리 테스크에 적용 가능

OpenAPI 의 GPT 모델 발전 역사는 다음과 같음.

우선, GPT-1, GPT-2 에 대해 알아볼 것임.

목차

1. GPT 모델

- 생성형 AI 란?

- 모델의 발전

2. GPT-1 모델

3. GPT-2 모델

4. 마무리

+) Reference

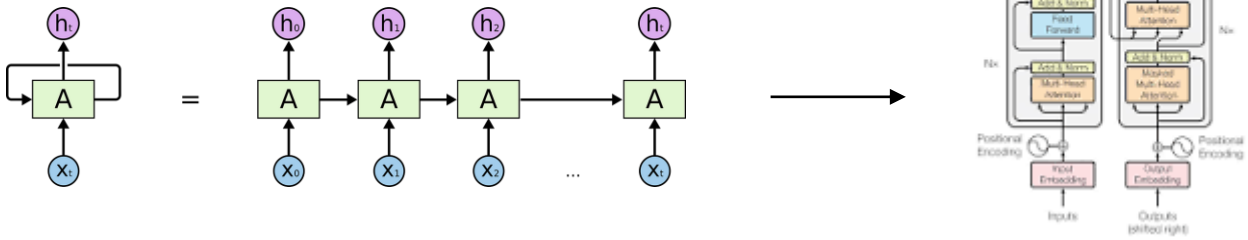
2. GPT-1 모델

1. 모델의 발전

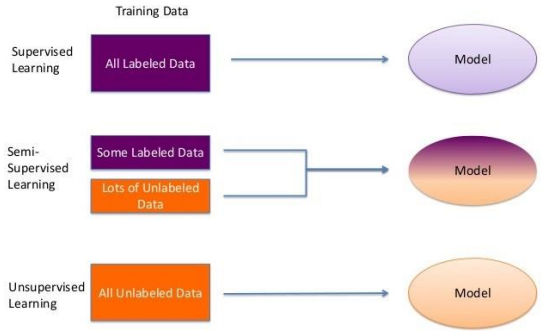
구분	GPT-1	GPT-2	GPT-3	GPT-4
발표 년도	2018년	2019년	2020년	2023년
파라미터 수	1.17억 개	15억 개	1.75조 개	10조 개 이상
학습 데이터	영어 위키피디아와 뉴스 기사	인터넷에 있는 거의 모든 웹페이지	인터넷에 있는 거의 모든 웹페이지	멀티 모달 데이터
특징	대규모 언어 모델의 초석	GPT-1 보다 더 많은 파라미터와 데이터를 사용	GPT-2 보다 더 많은 파라미터와 데이터를 사용	다양한 자연어 처리 테스크에 적용 가능

2. GPT-1 모델

2. 핵심 Point



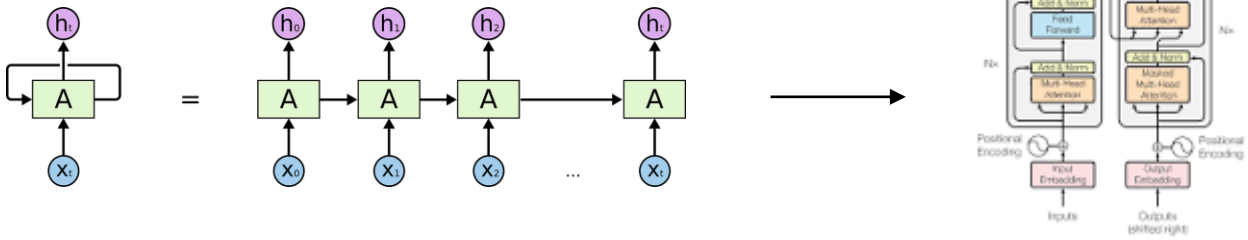
- 기존의 RNN 방식에서 **Transformer 모델** 선정



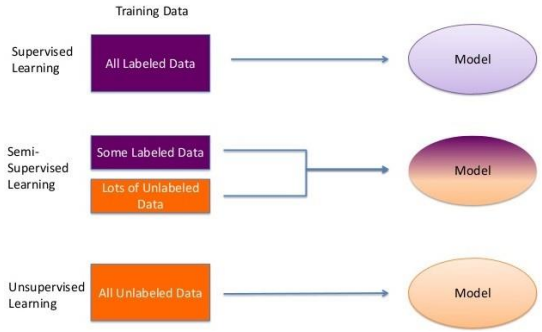
- **Unsupervised pre-training 과 Supervised fine-tuning 을 합친 모델**

2. GPT-1 모델

2. 핵심 Point



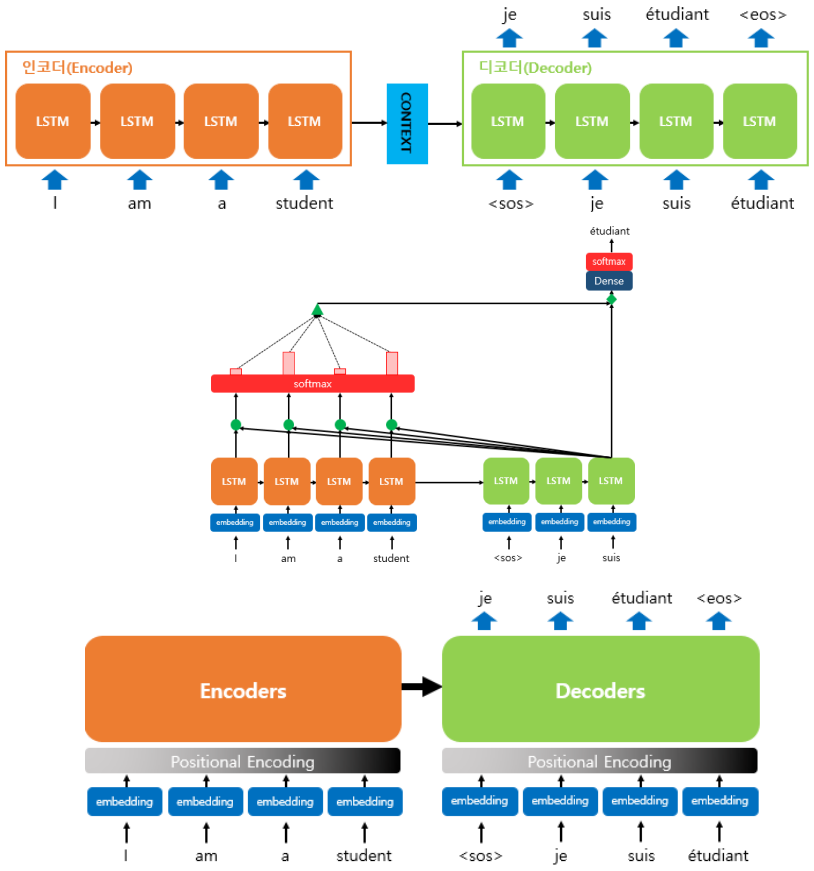
- 기존의 RNN 방식에서 **Transformer 모델** 선정



- **Unsupervised pre-training 과 Supervised fine-tuning 을 합친 모델**

2. GPT-1 모델

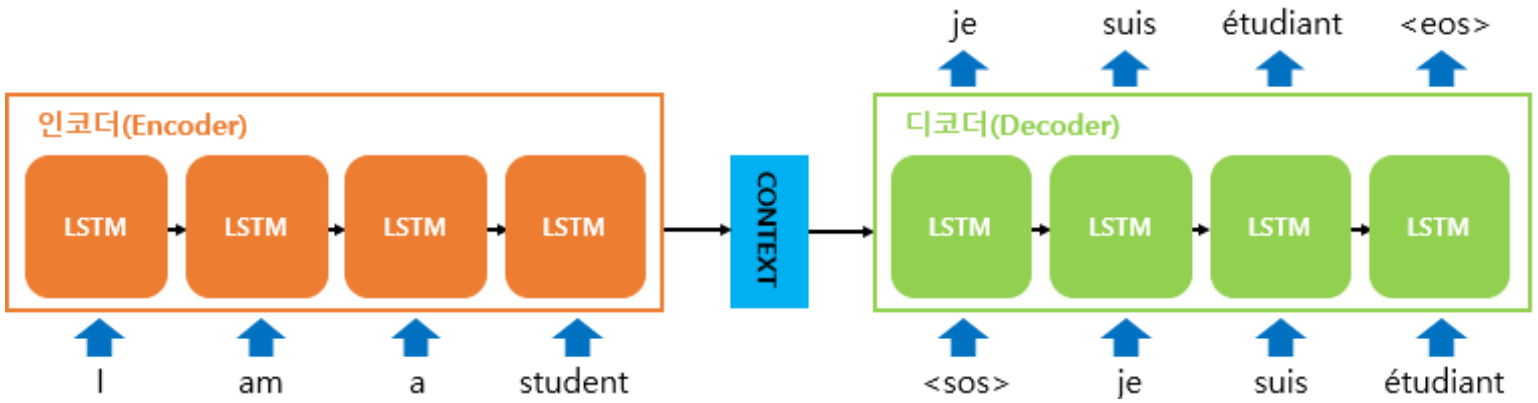
2. 핵심 Point - 기존의 RNN 방식에서 **Transformer 모델** 선정



Seq2seq → attention → Transformer

2. GPT-1 모델

2. 핵심 Point - 기존의 RNN 방식에서 **Transformer 모델** 선정



Seq2seq

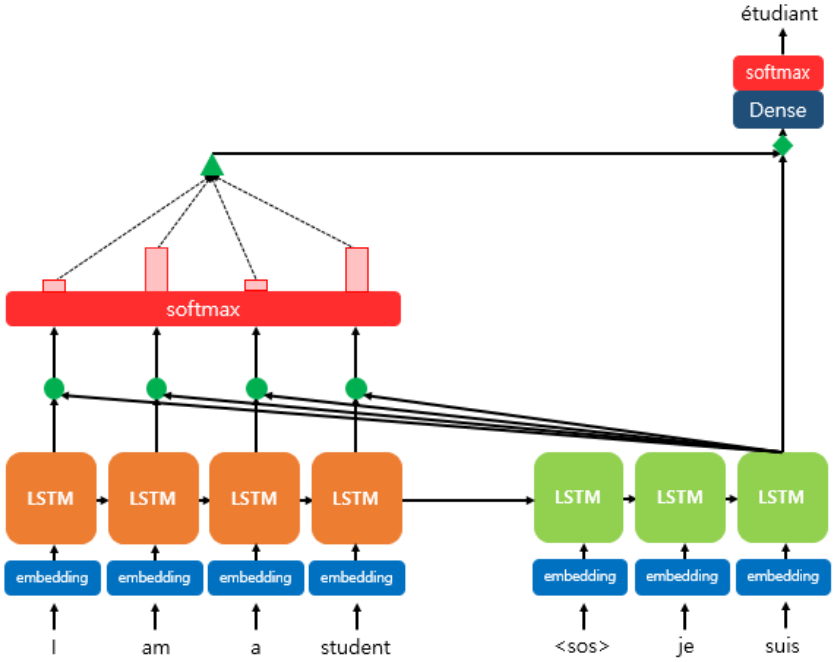
- 기존의 RNN 방식에서 벗어나 Encoder 와 Decoder 의 구조를 적용

  - Encoder 와 Decoder 사이에 Context Vector 를 추가하였음

    - <sos> 부터 마지막 <eos> 까지 순차적으로 매핑함

2. GPT-1 모델

2. 핵심 Point - 기존의 RNN 방식에서 **Transformer 모델** 선정

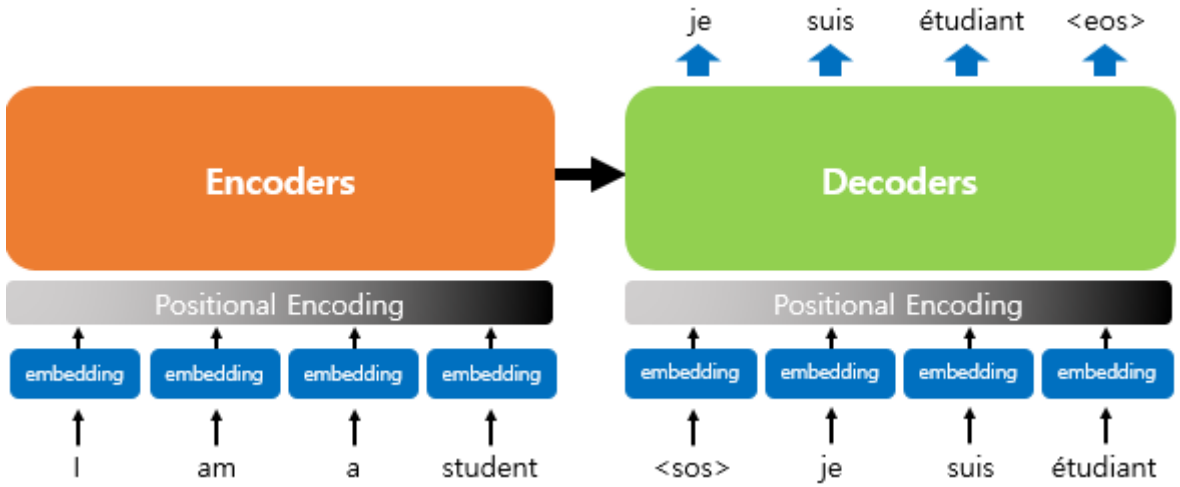


attention

- 이전의 seq2seq 구조에서도 엄청 긴 문장에 대해서는 성능이 안 좋았음
- 이전 lstm 의 은닉층과 attention value를 바탕으로 softmax 함수를 적용
  - 전체 문장에서 매핑하는 단어의 중요도를 파악할 수 있었음

2. GPT-1 모델

2. 핵심 Point - 기존의 RNN 방식에서 **Transformer 모델** 선정



Transformer

- 시간의 위치 정보를 가진 LSTM 레이어를 삭제하였음

- 그 대신 위치정보를 대신해주는 Positional Encoding 기법을 적용

- 이전의 보았던 Encoder 과 Decoder 를 여러 겹 쌓는 방식으로 구현

2. GPT-1 모델

2. 핵심 Point - 기존의 RNN 방식에서 **Transformer 모델** 선정

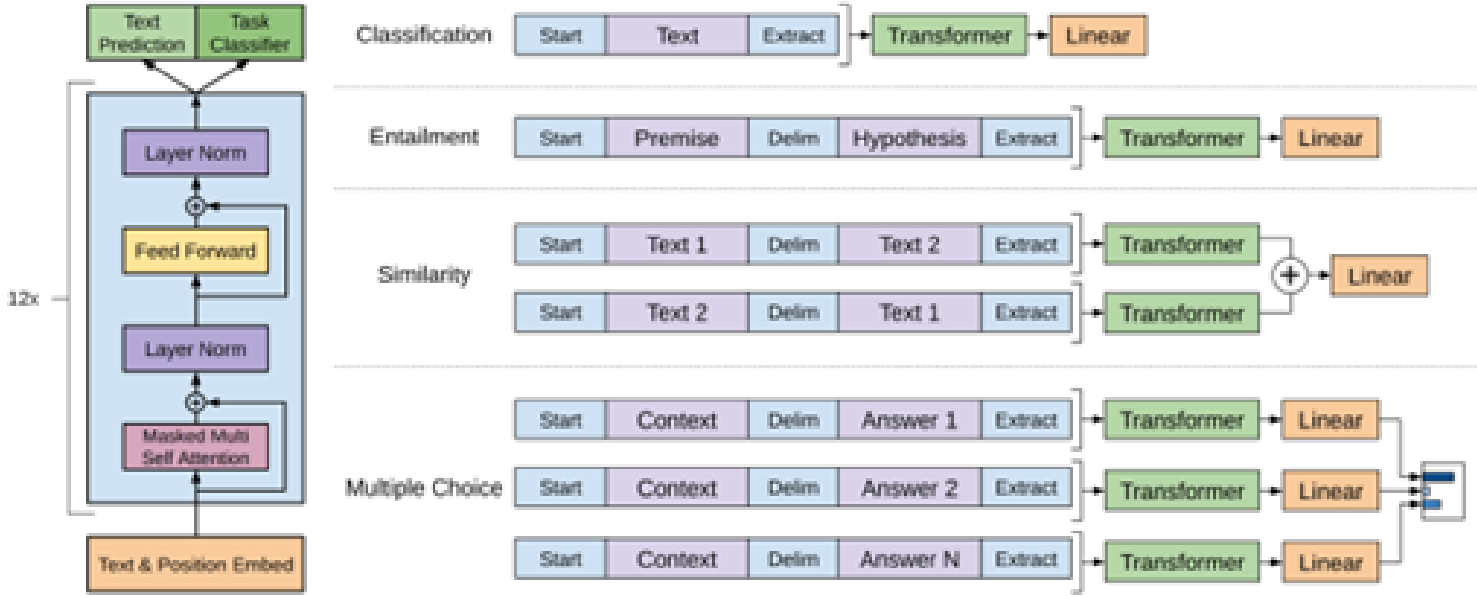
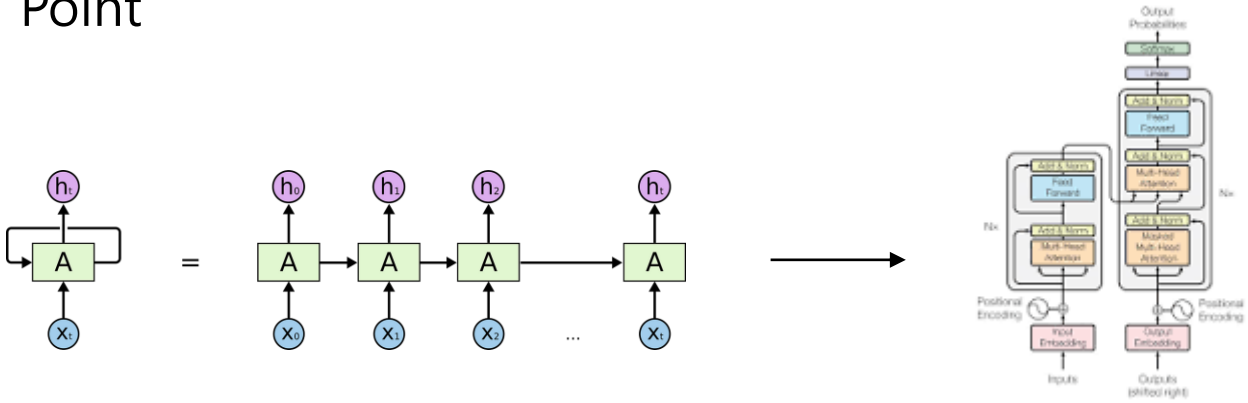


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

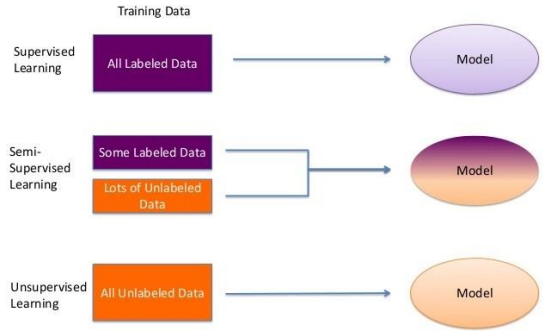
최종적으로, task-specific input transformer 를 적용함.

2. GPT-1 모델

2. 핵심 Point



- 기존의 RNN 방식에서 **Transformer 모델** 선정



- **Unsupervised pre-training 과 Supervised fine-tuning 을 합친 모델**

2. GPT-1 모델

2. 핵심 Point - **Unsupervised pre-training 과 Supervised fine-tuning 을 합친 모델**

3.1 Unsupervised pre-training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

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

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \quad (2)$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

- Unsupervised data 로 pre-training 진행
- Language Model 을 대규모 corpus 에서 학습 진행 -> L1

2. GPT-1 모델

2. 핵심 Point - **Unsupervised pre-training 과 Supervised fine-tuning 을 합친 모델**

3.2 Supervised fine-tuning

After training the model with the objective in Eq. 1, we adapt the parameters to the supervised target task. We assume a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label y. The inputs are passed through our pre-trained model to obtain the final transformer block's activation h_l^m, which is then fed into an added linear output layer with parameters W_y to predict y:

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \tag{3}$$

This gives us the following objective to maximize:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m). \tag{4}$$

We additionally found that including language modeling as an auxiliary objective to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence. This is in line with prior work [50, 43], who also observed improved performance with such an auxiliary objective. Specifically, we optimize the following objective (with weight λ):

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \tag{5}$$

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

- Supervised data 가 입력이 되면 이전에 pre-trained 된 LM 을 바탕으로 fine-tuning task 목적에 맞춰 목표를 추가

목차

1. GPT 모델

- 생성형 AI 란?

- 모델의 발전

2. GPT-1 모델

3. GPT-2 모델

4. 마무리

+) Reference

3. GPT-2 모델

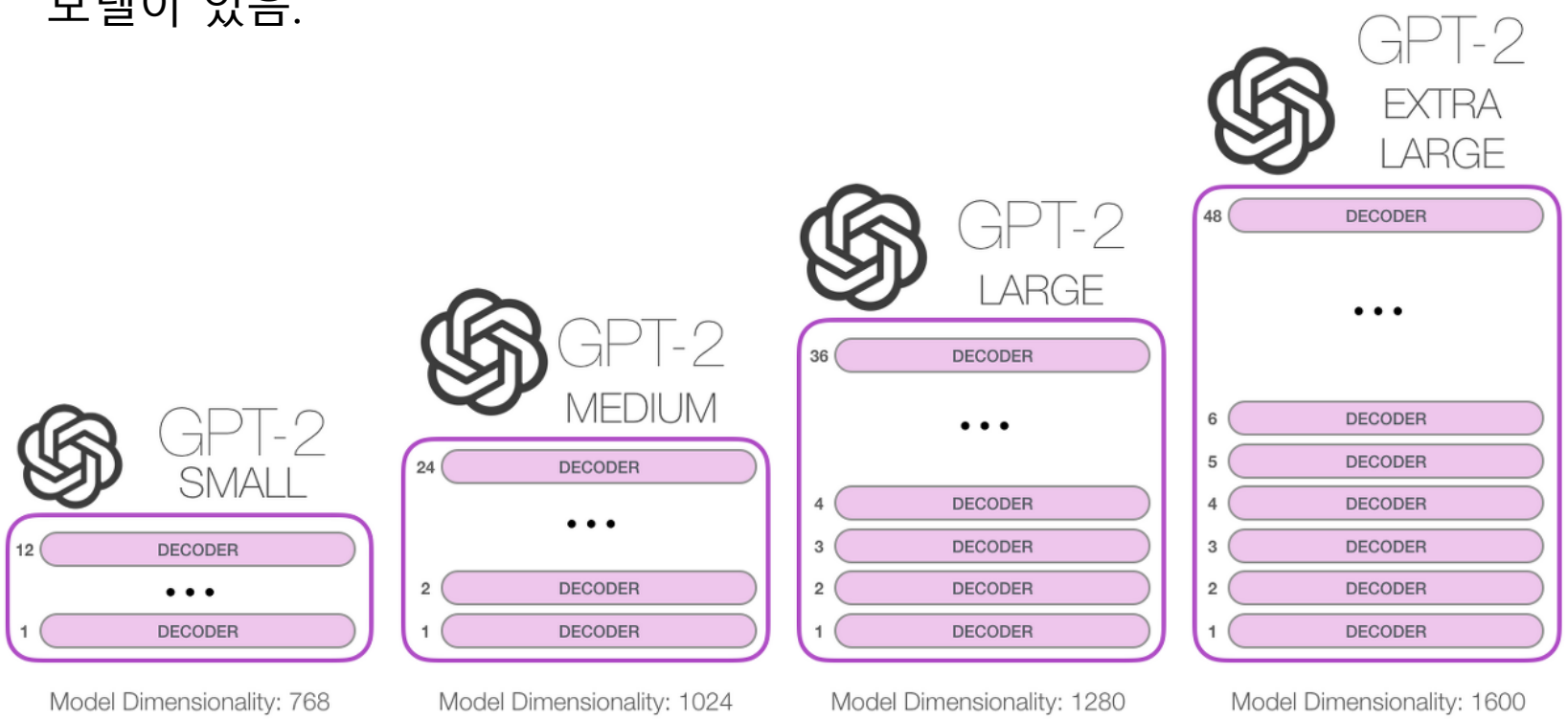
1. 모델의 발전

구분	GPT-1	GPT-2	GPT-3	GPT-4
발표 년도	2018년	2019년	2020년	2023년
파라미터 수	1.17억 개	15억 개	1.75조 개	10조 개 이상
학습 데이터	영어 위키피디아와 뉴스 기사	인터넷에 있는 거의 모든 웹페이지	인터넷에 있는 거의 모든 웹페이지	멀티 모달 데이터
특징	대규모 언어 모델의 초석	GPT-1 보다 더 많은 파라미터와 데이터를 사용	GPT-2 보다 더 많은 파라미터와 데이터를 사용	다양한 자연어 처리 테스크에 적용 가능

3. GPT-2 모델

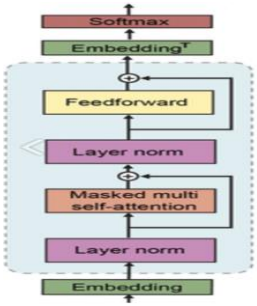
1. 모델의 발전

GPT-2 를 기준으로, SMALL 부터 EXTRA LARGE 까지 다양한 버전의 모델이 있음.

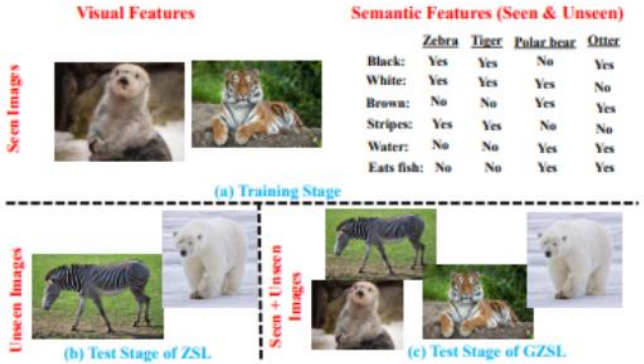


3. GPT-2 모델

2. 핵심 Point



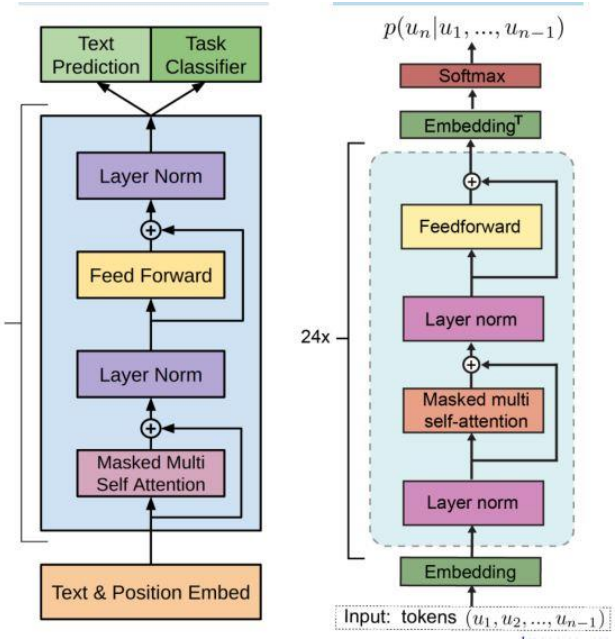
- 구조 변경 (더 큰 대용량 데이터, 배치 사이즈 증가, context size 증가 등)



- **Fine-tuning 없이 unsupervised pre-training 만을 통하여 Zero-Shot 으로 학습 진행**

3. GPT-2 모델

2. 핵심 Point - 구조 변경 (더 큰 대용량 데이터, 배치 사이즈 증가, context size 증가 등)



	GPT-1	GPT-2
Parameters	117 Million	1.5 Billion
Decoder Layers	12	48
Context Token Size	512	1024
Hidden Layer	768	1600
Batch Size	64	512

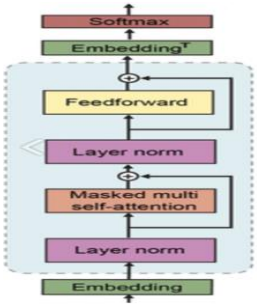
파라미터 수가 약 13배 증가

Layer normalization 위치가 다음과 같이 바뀜.

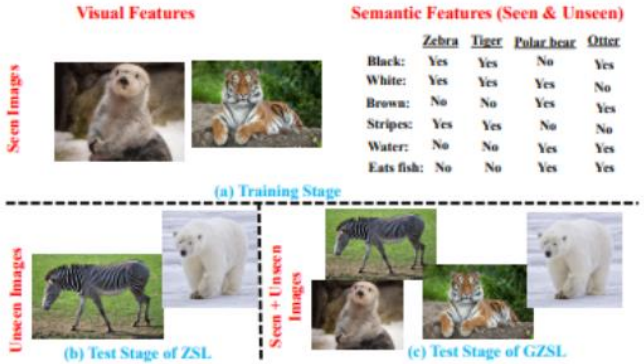
배치 사이즈 늘림 (64 -> 512)

3. GPT-2 모델

2. 핵심 Point



- 구조 변경 (더 큰 대용량 데이터, 배치 사이즈 증가, context size 증가 등)



- **Fine-tuning 없이 unsupervised pre-training 만을 통하여 Zero-Shot 으로 학습 진행**

3. GPT-2 모델

2. 핵심 Point - **Fine-tuning 없이 unsupervised pre-training 만을 통하여 Zero-Shot 으로 학습 진행**

2. Approach

At the core of our approach is language modeling. Language modeling is usually framed as unsupervised distribution estimation from a set of examples $(x_1, x_2, ..., x_n)$ each composed of variable length sequences of symbols $(s_1, s_2, ..., s_n)$. Since language has a natural sequential ordering, it is common to factorize the joint probabilities over symbols as the product of conditional probabilities (Jelinek & Mercer, 1980) (Bengio et al., 2003):

$$p(x) = \prod_{i=1}^{n} p(s_n|s_1, ..., s_{n-1}) \quad (1)$$

This approach allows for tractable sampling from and estimation of $p(x)$ as well as any conditionals of the form $p(s_{n-k}, ..., s_n|s_1, ..., s_{n-k-1})$. In recent years, there have been significant improvements in the expressiveness of models that can compute these conditional probabilities, such as self-attention architectures like the Transformer (Vaswani et al., 2017).

- Unsupervised data 로 pre-training 진행
- Language Model 을 대규모 corpus 에서 학습 진행 -> L1 -> **동일 진행**

3. GPT-2 모델

2. 핵심 Point - **Fine-tuning 없이 unsupervised pre-training 만을 통하여 Zero-Shot 으로 학습 진행**

- 그렇다면, Zero-Shot Learning 이란?

간단하게 설명하자면, 모델이 학습 과정에서 배우지 않은 작업을 수행하는 것임.

이게 가능한 이유는 수많은 비지도 학습에서 얻은 확률 분포 때문임.

3. GPT-2 모델

2. 핵심 Point - **Fine-tuning 없이 unsupervised pre-training 만을 통하여 Zero-Shot 으로 학습 진행**

Fine-tuning 없이 Zero-Shot 으로 학습하고자 하였기 때문에,
더 다양한 데이터로도 학습이 가능하였음.

하나의 도메인이 아닌 Web scraping dataset 으로 다양한 도메인 데이터 사용

3. GPT-2

3. 결론

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

- Zero shot 으로 8 개 중 7 개에서 SOTA 달성하였음

- 독해에 대해서는 supervised learning model 과 비슷한 성능을 내지만 요약 등의 Task 에서는 성능이 제대로 나오지 못하여 실제 사용에 대한 한계가 있음.

4. 마무리

1. 마무리

구분	GPT-1	GPT-2	GPT-3	GPT-4
발표 년도	2018년	2019년	2020년	2023년
파라미터 수	1.17억 개	15억 개	1.75조 개	10조 개 이상
학습 데이터	영어 위키피디아와 뉴스 기사	인터넷에 있는 거의 모든 웹페이지	인터넷에 있는 거의 모든 웹페이지	멀티 모달 데이터
특징	대규모 언어 모델의 초석	GPT-1 보다 더 많은 파라미터와 데이터를 사용	GPT-2 보다 더 많은 파라미터와 데이터를 사용	다양한 자연어 처리 테스크에 적용 가능

이후, GPT-3 에서부터는 Zero-Shot 이 아닌 Few-Shot 을 사용하였음.

생성형 AI 기술의 품질이 계속 좋아지면서 각광 받기 시작하였음.

Thank you!

MILab Undergraduate student, Kim Taehyeon

2023. 08. 10

Reference

- http://cloudinsight.net/ai/%EA%B1%B0%EB%8C%80-%EB%AA%A8%EB%8D%B8%EC%9D%98-%EB%B0%9C%EC%A0%84%EA%B3%BC-zero-shot%EC%9D%98-%EC%9D%98%EB%AF%B8/
- https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf
- https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
- https://www.thedatahunt.com/trend-insight/generative-ai
- https://medium.com/ai-networkkr/gpt-%EB%AA%A8%EB%8D%B8%EC%9D%98-%EB%B0%9C%EC%A0%84-%EA%B3%BC%EC%A0%95-%EA%B7%B8%EB%A6%AC%EA%B3%A0-%ED%95%9C%EA%B3%84-81cea353200c
- https://lsjsj92.tistory.com/617, https://lsjsj92.tistory.com/620
- https://360digitmg.com/blog/types-of-gpt-in-artificial-intelligence
- https://blog.naver.com/allbareunkr/222852267626

+) 추가

RNN 구조에서는 순서대로 들어오는 입력이 자연스럽게 모델에 들어왔으나, 어텐션 연산에서는 순서 정보가 고려되지 않습니다. 그렇기에 트랜스포머 모델에서는 모델에 입력되는 입력 임베딩(Input Embedding)에 **Positional Encoding**이라 불리는 입력 임베딩과 같은 차원의 위치 정보를 담고 있는 벡터를 더해줍니다.

입력 임베딩의 차원이 d라 할 때, p번째 단어의 **Positional Encoding**은 다음 식을 사용하여 계산됩니다. (단, $i = 0, 1, ..., (d-1)$)

$$f(p) = \begin{cases} \sin\left(\frac{p}{10000^{i/d}}\right) & (i = 2k) \\ \cos\left(\frac{p}{10000^{(i-1)/d}}\right) & (i = 2k+1) \end{cases}$$

예를 들어 `I love you but not love him`이라는 문장이라면 앞의 `love`와 뒤의 `love`는 일반적인 임베딩만을 거쳤을 때 동일한 값을 가지게 됩니다(이는 일반적인 Word2Vec과 같은 임베딩의 문제이기도 합니다). 하지만 Positional Encoding 이라는 주기함수에 의한 위치에 따른 다른 임베딩을 거치면 같은 단어일지라도 문장에서 쓰인 위치에 따라 다른 임베딩 값을 갖게 되어 해당 단어의 위치 정보가 성공적으로 모델에 전달됩니다.

+) 추가

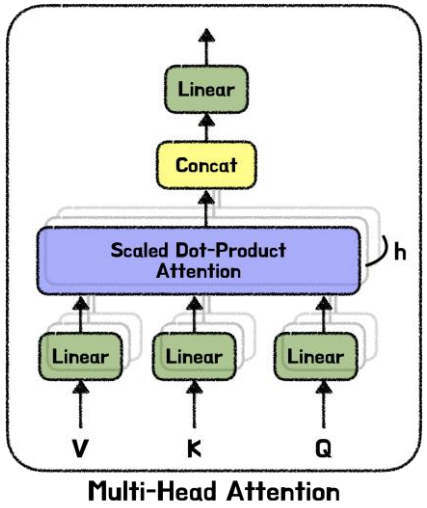
2. Query, Key, Value !

어텐션의 목표는 value를 통해 가중치 합계를 계산하는 것이고 각 Value의 가중치는 주어진 Query와 Key가 얼마나 유사한가에 따라 결정된다. 즉 어텐션은 전체 시퀀스를 동일하게 고려하지 않고 query, key, value를 통해 입력 시퀀스의 각기 다른 부분에 집중하게 된다.

더 자세히 설명하자면 query, key, value는 아래와 같다.

- Query
  - 입력 시퀀스에서 관련된 부분을 찾으려고 하는 정보 벡터(소스)
  - Machine Translation을 예로 들면, 디코더의 현재 상태
  - 관계성, 즉 연관된 정도를 표현하는 가중치를 계산하는데 사용
- Key
  - 관계의 연관도를 결정하기 위해 query와 비교하는데 사용되는 벡터(타겟)
  - Machine Translation을 예로 들면, 소스 문장의 인코더 표현
  - 관계성, 즉 연관된 정도를 표현하는 가중치를 계산하는데 사용
- Value
  - 특정 key에 해당하는 입력 시퀀스의 정보로 가중치를 구하는데 사용되는 벡터(밸류)
  - Machine Translation을 예로 들면, 소스 문장의 인코더 표현
  - 관계성을 표현하는 가중치 합이 최종 출력을 계산하는데 사용

일종의 파이썬 딕셔너리라고 생각하면 이해하기가 쉽다. 파이썬 딕셔너리의 경우 찾고자하는 것이 key값과 동일해야만 value를 가져올 수 있다. 하지만 어텐션의 Q, K, V 구조에서는 query와 key가 적당히 유사하면 유사한 만큼 value를 가져올 수 있다. 이때 그 유사도를 계산하는 것이 벡터곱이고, 해당 곱의 양만큼 value의 양을 가져와야 해서 또다시 곱을 하는 건데, 한줄로 설명하기에는 복잡하다. 그래서 아래에서 자세히 설명하려고 한다.



Multi-Head Attention