트랜스포머를 활용한 자연어 처리

MS AI 스쿨

Infinyx

01

트랜스포머 소개

1. 트랜스포머 소개
2. 인코더 – 디코더 프레임워크 소개
3. 어텐션 메커니즘
4. NLP의 전이 학습
5. 허깅페이스 트랜스포머
6. 트랜스포머 애플리케이션 둘러보기
7. 허깅페이스 생태계

Infinyx
THINK LIFE SYNC AI

개발 공부를 잘하는 방법 !!!!

1. 처음 배울 때 ---------- >>>> 기본 쌓기
2. 기본이 쌓이면 --------- >>>> 지식 강화 (다지기)
3. 주기적으로 ---------- >>>> 역량 강화

https://pytorch.org/tutorials/ https://docs.python.org/3.8/

https://pytorch.org/docs/stable/index.html

프로그래밍 언어, 프레임워크, 라이브러리를 배울 때 공식 사이트가 제일 정확하다
>>> 철학, 의도, 사용법

질문이 생길 때마다 첫번째로 해야 할 일은 공식 문서에서 답을 구할 수 있는지 알아 본다
>>> 재확인

개발 공부를 잘하는 방법 !!!!

제일 처음 공식문서로 기본기를 쌓고 실력과 자신감이 향상 되면서 공식문서로 재확인 주요 업데이트 내용을 공식문서에서 빠르게 얻어냄

>>>>>>>>> 제일 빨리 앞서갈 수 있음
>>>>>>>>> 지식과 이해도를 향상할 수 있음

개발자의 제일 좋은 습관
공식 문서를 읽고
공식문서에서 찾고
공식문서를 참조

온라인 강의를 배우는 경우도 동일 !!!

- git nation의 리스트 참조: 컨퍼런스나 서밋에 초대, 참여나 유지보수 등 문서를 꾸준히 작성하는 분, 특정한 토픽에 꾸준히 나오는 분의 블로그를 봐서 신뢰할수 있다면 팔로잉. 리액트 공부나 유닛테스트에 관심이 있다면 Ken.C dodds분의 블로그 참조, 리액트 쿼리를 배우고 있다면 공식문서와 같이 Dominik dorfmeister분의 블로그를 읽어보기. 공식 문서+명성있는 지식인들의 블로그를 같이 보면서 공부.

2. 기본지식을 제대로 익혀라 !!! 파이썬 문법을 정확하게 파악 !!

3. 메모하기 : 내가 필요한 지식들을 정리하기

개발 공부를 잘하는 방법 !!!!

- git nation의 리스트 참조: 컨퍼런스나 서밋에 초대, 참여나 유지보수 등 문서를 꾸준히 작성하는 분, 특정한 토픽에 꾸준히 나오는 분의 블로그를 봐서 신뢰할수 있다면 팔로잉. 리액트 공부나 유닛테스트에 관심이 있다면 Ken.C dodds분의 블로그 참조,

리액트 쿼리를 배우고 있다면 공식문서와 같이 Dominik dorfmeister분의 블로그를 읽어보기. 공식 문서+명성있는 지식인들의 블로그를 같이 보면서 공부.

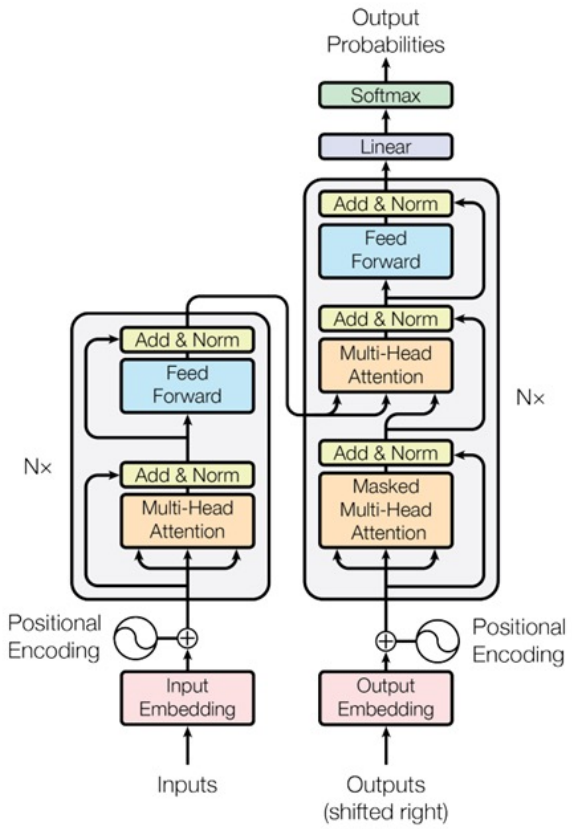
트랜스포머 소개

자연어 처리(Natural Language Processing – NLP)는 인공지능 모델을 활용하여 사람의 언어를 모방하는 모든 분야를 말합니다. AI 분야에서 큰 한 축을 담당하는 만큼 많은 사람들이 연구하고 있으며, 새로운 연구자 또한 계속해서 나오고 있습니다. 그리고 지금 단계에서, 딥러닝 초급자를 넘어 자연어 처리를 공부하고자 하는 모든 사람들에게 피할 수 없는 관문 같은 키워드가 있습니다.
바로 트랜스포머 입니다.

트랜스포머는 2017년 구글이 발표한 논문인 Attention is all you need에서 나온 모델로 기존의 seq2seq의 구조인 인코더-디코더를 따르면서도 어텐션으로만 구현만 모델입니다. 본 모델은 RNN을 사용하지 않고 인코더-디코더 구조를 설계했음에도 성능이 RNN보다 우수하다고 합니다.

트랜스포머는 강력했으며, 언어적 특징을 가장 잘 고려한 모델이라고도 볼 수 있습니다. 트랜스포머가 낳은 수많은 자식들이 현재 자연어 처리 분야를 지배하고 있는 만큼, 자연어 처리를 공부하는 사람에게 트랜스포머는 하나하나 꼭꼭 씹어서 소화해야만 하는 모델이라고 볼 수 있습니다.

트랜스포머 소개



논문의 제목을 그대로 해석해 보면 어떤 가요? Attention이야말로 모든 것이라고 말하고 있습니다. 트랜스포머가 자연어 분야를 평정한 데에는 바로 이 Attention을 적극적으로 기용한 데에 있습니다. 자연어 모델을 다룰 때에는 트랜스포머 못지않게 이 Attention에 대해 자주 논할 수 밖에 없습니다. 그 중에서도 트랜스포머의 핵심은 self-attention에 있습니다.

Attention이 의미하는 것은 '강조' 입니다. 일반적으로 어떤 말을 통해 의미를 전달하고자 할 때 핵심은 사실 한 두 문장, 혹은 한 두 단어에 압축 되어 있는 경우가 많습니다. 여러 단어들이 함께 나열되어 하나의 글을 이루어도, 각각의 중요도가 서로 다르다는 것은 구태여 언급하지 않아도 모두가 알고 있습니다. 트랜스포머는 이러한 언어적 특징을 중점적으로 다루었습니다. 모델에게 언어를 가르칠 때 문장 내에서 단어 간 중요도에 차이가 있음을 '강하게' 알려준 것입니다. 그냥 attention이 아닌 self-attention인 이유는 스스로가 스스로를 강조하기 때문입니다.

https://arxiv.org/pdf/1706.03762.pdf

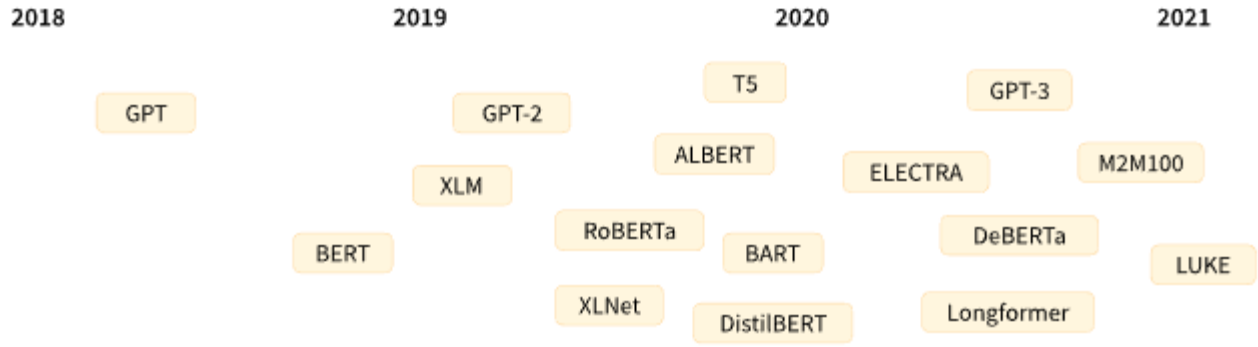
트랜스포머 소개

이해를 돕기 위해 예시를 한번 들어보겠습니다.
‘The man cross the river that is large with a small boat’

위 문장을 살펴 볼까요? 남자가 강을 건너는 주체이고, 그 강은 큰 강이며, 남자가 탄 보트는 작다는 것을 즉시 파악할 수 있습니다. 하지만 학습되지 않은 언어모델은 boat라는 단어와 small이라는 단어가 무슨 관계인지, large라는 단어와 river라는 단어가 무슨 관계인지, cross와 man은 무슨 관계인지 전혀 알지 못합니다. self-attention은 이런 이해를 할 수 있게 도와줍니다. 각 단어 입장에서 어떤 부분을 강조해야 할지 알려주는 것입니다.

트랜스포머 모델에서 파악해야 할 것은 self-attention만이 아닙니다. 나열하기도 어려울 만큼 다양한 기능이 한데 어우러져 트랜스포머라는 거대한 구조를 이루고 있습니다.

트랜스포머 소개



Transformer 아키텍처는 2017년 6월에 소개되었습니다.
- 원래 이 아키텍처 연구의 초점은 기계 변역이었습니다. 최초 모델이 소개된 후 다음과 같은 몇 가지 강력하고 우수한 모델이 추가적으로 도입되었습니다.

2018년도 6월 GPT 최초의 사전 학습된 모델, 다양한 NLP 작업에 대한 미세 조정에 사용되고 있음

2018년도 10월 BERT 또 다른 대규모 사전 학습된 모델, 이 모델은 특히 고수준의 문장 요약을 제공하도록 설계되었습니다.

2019년도 2월 GPT-2 윤리적인 문제로 인해 즉시 공개되지 않은, 기존 GPT 보다 규모가 더 크고 성능이 향상된 GPT 버전

2019년도 10월 DistillBERT 속도가 60% 더 빠르고 메모리 소비는 40% 줄였지만 여전히 BERT 성능의 97%를 유지하는 증류된 BERT

2019년도 10월 BART 및 T5, 원래 Transformer 모델과 동일한 아키텍처를 사용하는 두 개의 대규모 사전 학습모델

2020년도 5월 GPT-3 미세조정 없이도 다양한 작업을 훌륭하게 수행할 수 있는 GPT-2의 더 큰 버전

트랜스포머 소개

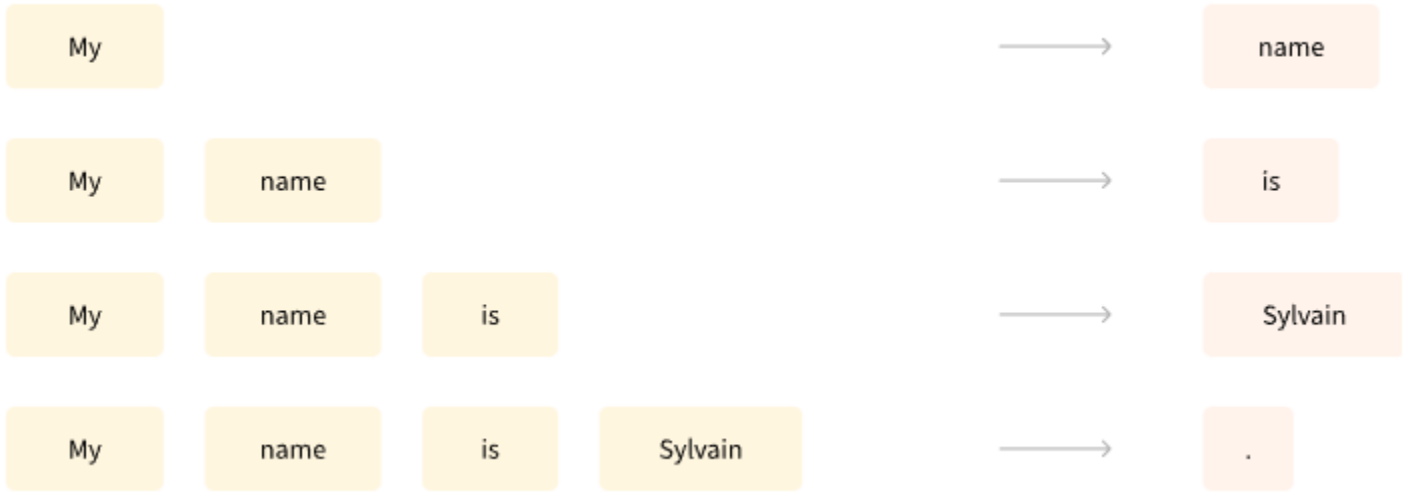
Transformers는 언어 모델입니다.

위에서 언급한 모든 Transformer 모델(GPT, BERT, BART, T5 등)은 *언어 모델(language model)* 로 학습되었습니다. 이는 그 모델들이 자가 지도(self-supervised) 학습 방식으로 많은 양의 원시 텍스트에 대해 학습되었음을 의미합니다. 자가 지도 학습(self-supervised learning)은 목적 함수(objectives)가 모델의 입력에서 자동으로 계산되는 학습 유형입니다. 즉, 사람이 데이터에 레이블을 지정할 필요가 없습니다!
이러한 유형의 모델은 학습된 언어에 대한 통계적인 이해를 가능하게 하지만 실제 태스크에는 그다지 유용하지 않습니다. 이런 이유때문에 일반적으로 사전 학습된 모델은 *전이 학습(transfer learning)* 이라는 프로세스를 거치게 됩니다. 이 프로세스 동안 모델은 주어진 작업에 대해 감독(supervised) 방식 즉, 사람이 주석으로 추가한 레이블을 사용함으로써 미세 조정(fine-tuning)됩니다.

트랜스포머 소개

Transformers는 언어 모델입니다.

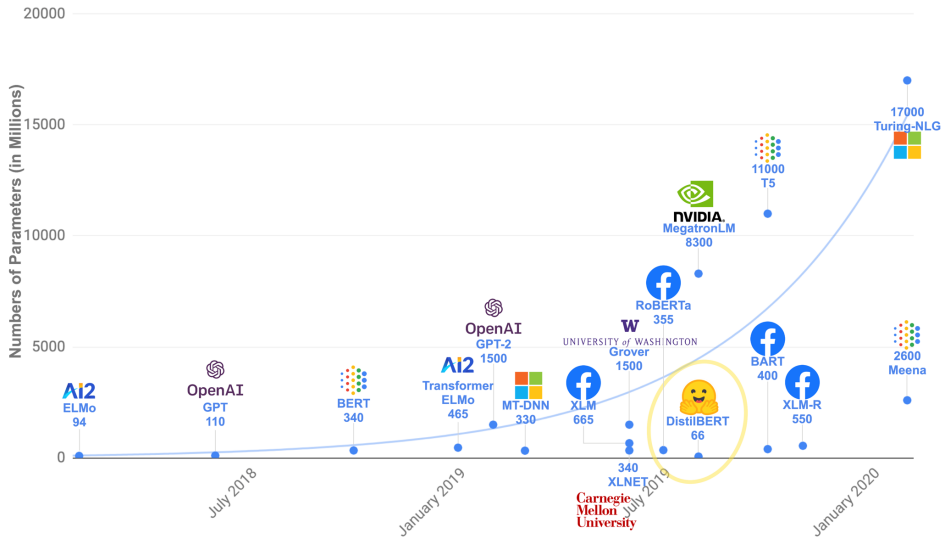
예를 들어, n개의 이전 단어를 읽은 문장에서 다음 단어를 예측한다고 합시다. 여기서 출력할 예측 값은 과거 및 현재 입력 값에 의존하지만 미래 입력 값에는 의존하지 않기 때문에 우리는 이것을 *causal language modeling(인과적 언어 모델)* 이라고 합니다.



트랜스포머 소개

Transformers는 규모가 큰 모델입니다.

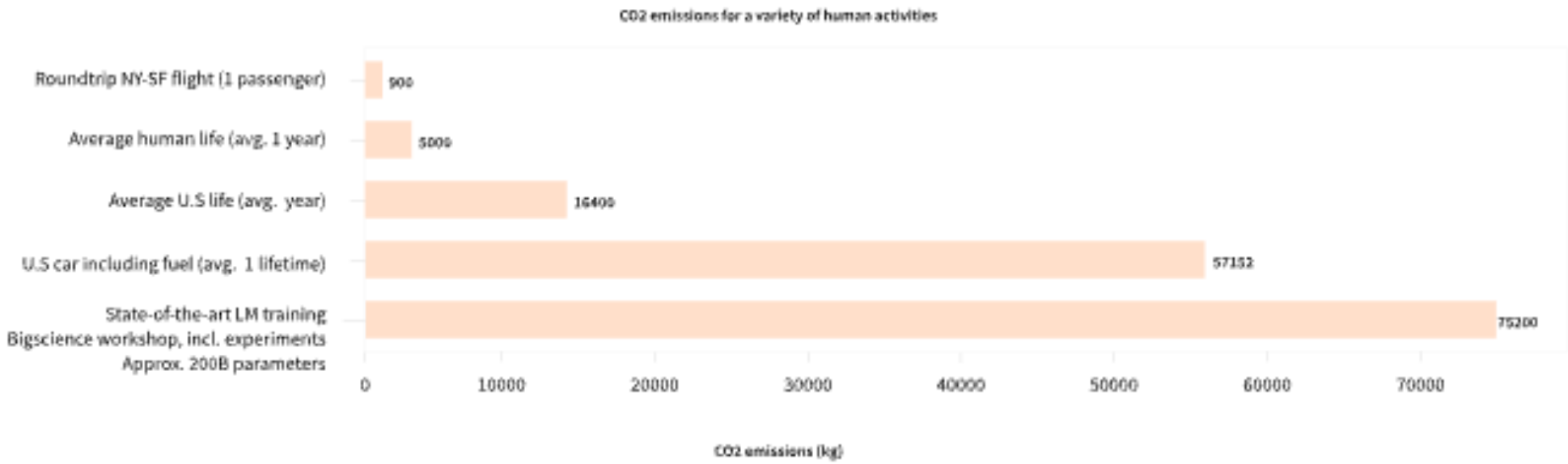
몇 가지 예외 모델(예: DistilBERT)을 제외하고, 더 나은 성능을 달성하기 위한 일반적인 전략은 모델의 크기와 사전 훈련된 데이터의 양을 늘리는 것입니다.



불행히도 사전 학습 모델(pretrained model), 특히 큰 모델을 학습하려면 많은 양의 데이터가 필요합니다. 이는 시간과 컴퓨팅 리소스 면에서 고비용을 초래합니다.

심지어 이러한 대규모 모델 학습은 다음 그래프에서 볼 수 있듯이 환경적인 문제(environmental impact)를 야기하기도 합니다.

트랜스포머 소개



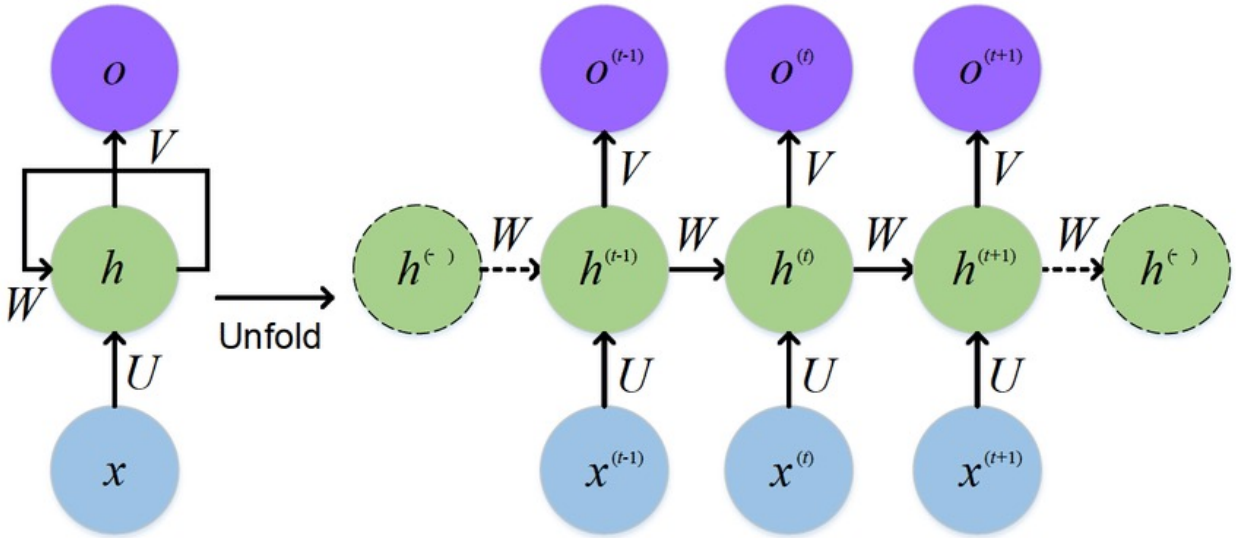
위 그래프의 맨 아래에서 사전 학습의 환경적 영향을 줄이기 위해 의식적으로 노력하고 있는 한 연구팀에 의해서 진행되는 프로젝트에서 대규모의 사전 학습을 실행할 때 배출되는 이산화탄소의 양을 보여주고 있습니다. 최적의 하이퍼 파라미터(hyperparameter)를 얻기 위해 많은 학습 시도를 실행해야 하는 경우 탄소 발자국(carbon footprint)은 훨씬 더 높을 것입니다.

이것이 언어 모델을 공유해야만 하는 가장 중요한 이유입니다. 학습된 가중치(weights)를 공유하고 이미 학습된 가중치를 기반으로 미세 조정(fine-tuning)하여 모델을 만들면 커뮤니티의 전체 컴퓨팅 비용과 탄소 발자국(carbon footprint)을 줄일 수 있습니다.

인코더 – 디코더 프레임워크 소개

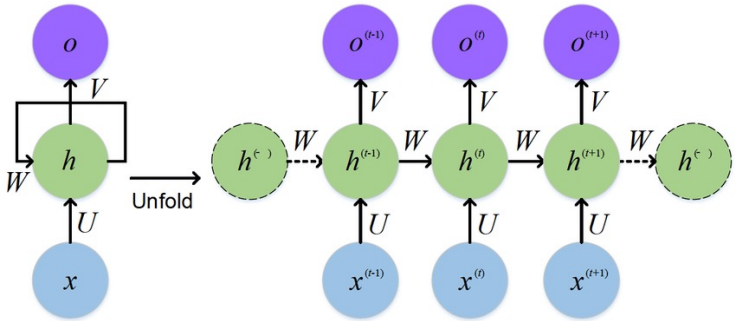
트랜스포머가 등장하기 전, NLP에서는 LSTM 같은 순환 신경망 구조가 최고 수준의 성능을 달성했습니다.
순환신경망 구조에는 정보를 한 스탭에서 다음 스탭으로 전파하도록 네트워크 피드백 루프가 포함

이런 구조는 텍스트와 같은 순차 데이터를 모델링하는 데 이상적입니다.



RNN은 이런 식으로 이전 스텝의 정보를 추적하고 이를 사용해 예측을 만듭니다.

트랜스포머 소개



이런 구조는 NLP 작업, 음성 처리, 시계열 작업에 널리 사용됐고 지금도 사용되고 있습니다.

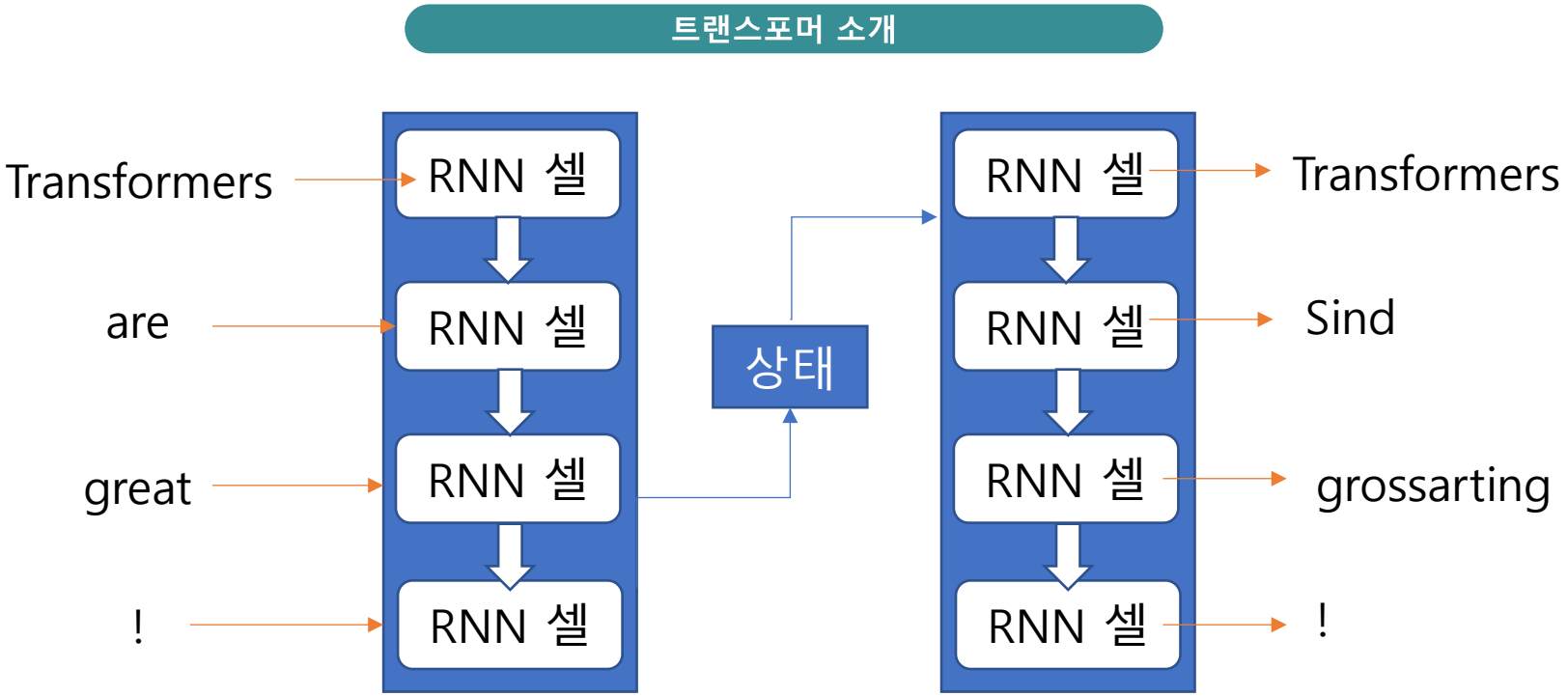
참고 블로그 : https://oreil.ly/Q55o0

RNN은 단어 스퀀스를 한 언어에서 다른 언어로 매핑하는 기계 번역 시스템을 개발할 때 중요한 역할

이러한 종류의 작업은 대개 인코더-디코더 또는 시퀀스-투-시퀀스 구조로 처리하며 입력과 출력이 임의의 길이를 가진 시퀀스일 때 잘 맞습니다.

- 참고 논문 : https://arxiv.org/abs/1409.3215 (2014)

인코더는 입력 시퀀스의 정보를 마지막 은닉 상태라고도 부르는 수치 표현으로 인코딩합니다.
그 다음 이 상태가 디코더로 전달되어 출력 시퀀스가 생성됩니다.



한 쌍의 RNN으로 구성된 인코더 디코더 구조 (보통은 그림에 있는 것보다 훨씬 더 많은 순환 층을 사용

위의 그림은 영어문장 "Transformers are great !"을 은닉 상태 백터로 인코딩한 다음, 이를 디코딩해 독일어 문장 "Transformers sind grossarting" 으로 한 쌍의 RNN을 보여줍니다. 입력 단어는 순차적으로 인코더에 주입되고 출력 단어는 위에서 아래 방향으로 한 번에 하나씩 생성됩니다.

허나 문제는 있습니다. 1. 이 구조는 인코더의 마지막 은닉 상태가 정보 병목이 된다는 약점이 있습니다.
디코더는 인코더의 마지막 은닉 상태만을 참조해 출력을 만듦으로 여기에 전체 입력 시퀀스의 의미가 담겨야 합니다. 시퀀스가 긴 경우, 모든 것을 고정된 하나의 표현으로 압축하는 과정에서 시작 부분의 정보가 손실될 가능성이 있어 더욱 취약합니다.

어텐션

디코더가 인코더의 모든 은닉 상태에 접근해 이 병목을 제거합니다. 이런 일반적인 메커니즘을 어텐션 이라고 합니다.

- 논문 https://arxiv.org/abs/1409.0473 (2014)

어텐션은 입력 스퀀스에서 은닉 상태를 만들지 않고 스텝마다 인코더에서 디코더가 참고할 은닉 상태를 출력한다는 주요 개념에 기초합니다. 하지만 모든 상태를 동시에 사용하려면 디코더에 많은 입력이 발생하므로 어떤 상태를 먼저 사용할지 우선순위를 정하는 메커니즘이 필요합니다. 여기서 어텐션이 등장합니다.

디코더가 모든 디코딩 타임스텝마다 인코더의 각 상태에 다른 가중치 또는 어텐션을 할당합니다.

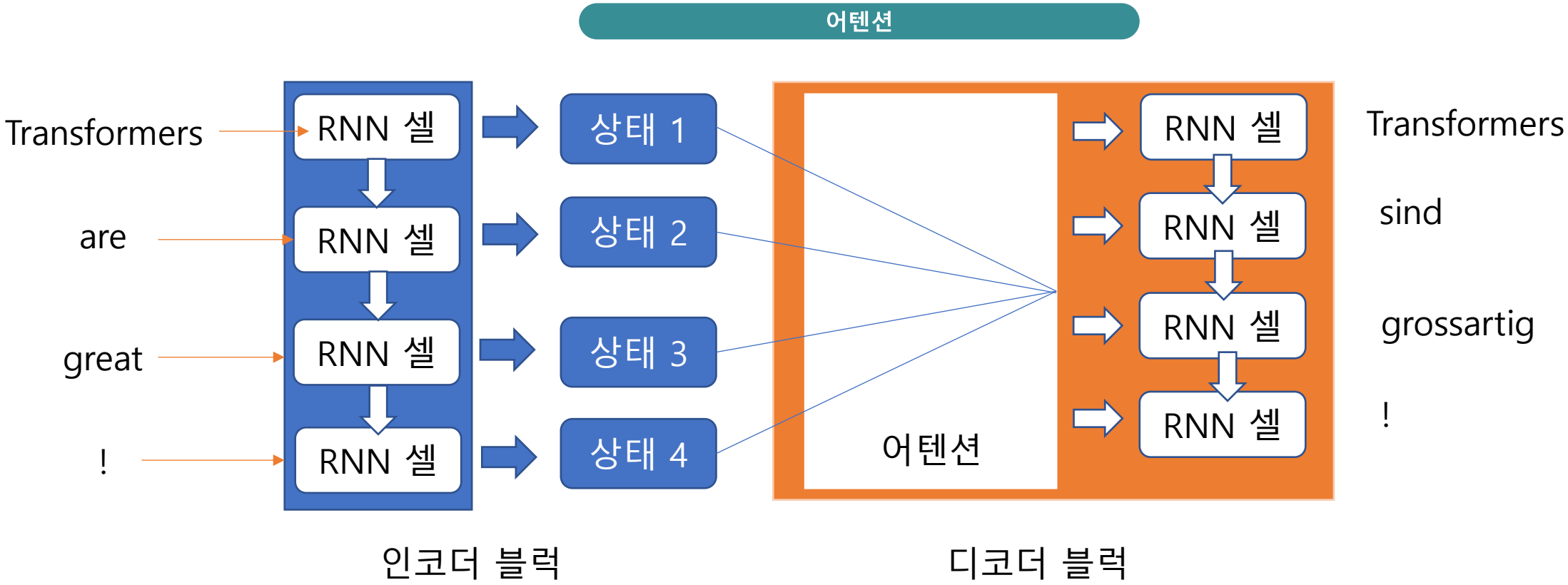


그림)에서 어텐션은 출력 시퀀스에 있는 두번째 토큰을 예측하는 역할 수행

어텐션 기반 모델은 타임스탭마다 가장 많이 관련된 입력된 토큰에 초점을 맞추므로 변역 문장에 있는 단어와 원 문장에 있는 단어의 복잡한 정렬 문제를 학습합니다.

어텐션 함수

어텐션 메커니즘을 언급하기 전에 컴퓨터공학의 많은 분야에서 사용되는 Key-Value로 구성되는 자료형에 대해서 잠깐 언급하겠습니다. 파이썬에도 Key-Value로 구성되는 자료형인 딕셔너리(Dict) 자료형이 존재합니다. 파이썬의 딕셔너리 자료형은 키(Key)와 값(Value)이라는 두 개의 쌍으로 구성되는데, 키를 통해서 맵핑된 값을 찾아낼 수 있다는 특징을 갖고있습니다.

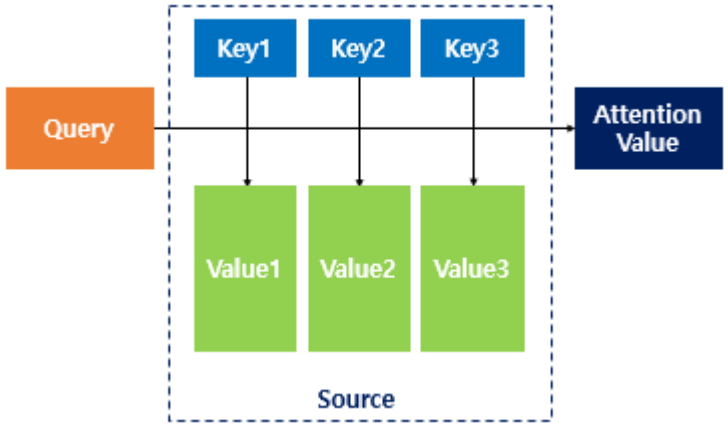
```
# 파이썬의 딕셔너리 자료형을 선언 # 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

위의 자료형에서 2017은 키에 해당되며, Transformer는 2017의 키와 맵핑되는 값에 해당됩니다.
그와 마찬가지로 2018은 키에 해당되며, BERT는 2018이라는 키와 맵핑되는 값에 해당됩니다.

어텐션

어텐션 함수

Key-Value 자료형에 대한 이해를 가지고 어텐션 함수에 대해서 설명해보겠습니다.



어텐션을 함수로 표현하면 주로 다음과 같이 표현됩니다.

Attention(Q, K, V) = Attention Value

Q = Query : t 시점의 디코더 셀에서의 은닉 상태
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들
V = Values : 모든 시점의 인코더 셀의 은닉 상태들

어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구합니다. 그리고 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해줍니다. 그리고 유사도가 반영된 '값(Value)'을 모두 더해서 리턴합니다.

어텐션

어텐션으로 번역이 한결 좋아졌지만, 인코더와 디코더에 사용하는 순환 모델의 단점은 여전히 존재합니다. 태생적으로 계산이 순차적으로 수행되며 입력 시퀀스 전체에 걸쳐 병렬화할 수 없습니다.

트랜스포머는 모델링 패러다임을 바꿨습니다. 순환을 모두 없애고 셀프 어텐션 이라는 특별한 형태의 어텐션에 전적으로 의지하게 된 것 입니다.

기본적으로 신경망의 같은 층에 있는 모든 상태에 대해서 어텐션을 작동시키는 방식이란 점만 알아 둡시다.

NLP의 전이 학습

요즘 컴퓨터 비전에서는 전이 학습을 사용해 ResNet 같은 합성곱 신경망을 한 작업에서 훈련한 다음 새로운 작업에 적용하거나 미세 튜닝하는 일이 많습니다. 이를 통해 신경망은 원래 작업에서 학습한 지식을 사용합니다.

구조적으로 볼 때 모델은 바디와 헤드로 나뉩니다. 바디의 가중치는 훈련하는 동안 원래 도메인에서 다양한 특성을 학습하고, 이 가중치를 사용해 새로운 작업을 위한 모델을 초기화합니다.

전통적인 지도학습과 비교하면 전이 학습은 일반적으로 다양한 작업에서 적은 양의 레이블 데이터로 훨씬 효과적으로 훈련하는 높은 품질의 모델을 만듭니다.

NLP의 전이 학습

컴퓨터 비전에서는 먼저 이미지가 수백만 개인 ImageNet 같은 대규모 데이터셋에서 모델을 훈련합니다. 이 과정이 사전 훈련이며, 모델에게 에지 나 색깔 같은 기본적인 이미지의 특징을 가르치는 것이 그 목적입니다.

컴퓨터 비전에서는 전이 학습이 표준이 됐지만, NLP에서 전이 학습과 유사한 사전 훈련 과정이 무엇인지는 수년간 특정하지 못했습니다. 그 결과 NLP 애플리케이션은 성능을 높이기 위해서 레이블링된 대량의 데이터를 사용했지만, 그럼에도 컴퓨터 비전의 성능보다 훨씬 낮았습니다.

2017년과 2018년 몇몇 연구 단체가 NLP에서 전이 학습을 수행하는 새 방식을 제안했습니다. OpenAI 연구원들은 감성 분류 작업에 비지도 사전 훈련에서 추출한 특성을 사용해 높은 성능을 얻으며, 소개(논문 https://arxiv.org/abs/1704.01444) (2017)된 방식입니다. 그 뒤를 이어 범용적인 프레임워크 ULMFiT가 등장했습니다. ULMFiT는 다양한 작업에 사전 훈련된 LSTM 모델을 적용합니다.

NLP의 전이 학습

알아야하는 용어 정리

* 사전 훈련
  - 이전 단어를 바탕으로 다음 단어를 예측하는 것입니다. 이 작업을 언어 모델링이라고 합니다. 편리하게도 이 작업은 레이블링된 데이터가 필요하지 않으며 위키피다아 같은 소스에 있는 풍부한 텍스트를 활용합니다.
  >> 지구상의 대부분 언어에서는 디지털화된 대규모 텍스트 말뭉치를 구하기 어렵기에 이는 영어에 해당하는 말입니다. 이 차이를 극복하는 방법을 찾는 것이 NLP의 한 분야이며 활발한 연구와 활동이 이루어짐

* GPT
  - 트랜스포머 아키텍처의 디코더 부분만 사용하고 ULMFiT 같은 언어 모델링 방법을 사용합니다. GPT 사전훈련한 BookCorpus 논문 : https://arxiv.org/abs/1506.06724 (2015) 데이터셋은 어드벤처, 판타지, 로맨스 등 장르를 망라한 미출판 도서 7,000권으로 구성됩니다.

NLP의 전이 학습

알아야하는 용어 정리

* BERT
 - 트랜스포머 아키텍처의 인코더 부분을 사용하고 마스크드 언어 모델링이라는 특별한 형태의 언어 모델링을 사용합니다. 마스크드 언어 모델링의 목표는 텍스트에서 랜덤하게 마스킹된 단어를 예측하는 것입니다.

예를 들어 I looked at my [MASK] and saw that [MASK] was late 라는 문장에서 모델은 [MASK]로 마스킹된 단어에 대해 가장 가능성이 높은 후보를 예측합니다. BERT는 BookCorpus 와 영어 위키피디아에서 사전 훈련 했습니다.

GPT 와 BERT 다양한 NLP 벤치마크에서 기록을 새롭게 갱신하며 트랜스포머 시대를 열었습니다.

단 문제가 있습니다. 연구실마다 서로 호환되지 않는 프레임워크(파이토치, 텐서플로우)를 사용해 모델 릴리즈 했고, 이런 모델은 NLP 기술자들이 자신의 애플리케이션에 포팅하기란 쉽지 않습니다.

이러한 문제를 해결하기위해서 허깅페이스 트랜스포머 https://github.com/huggingface/transformers 존재

허깅페이스 트랜스포머스

Transformers

build passing | license Apache-2.0 | website online | release v4.25.1 | Contributor Covenant v2.0 adopted | DOI 10.5281/zenodo.7391177

English | 简体中文 | 繁體中文 | 한국어 | Español | 日本語 | हिन्दी

Jax, Pytorch, TensorFlow를 위한 최첨단 자연어처리

🤗 Transformers는 분류, 정보 추출, 질문 답변, 요약, 번역, 문장 생성 등을 100개 이상의 언어로 수행할 수 있는 수천개의 사전학습된 모델을 제공합니다. 우리의 목표는 모두가 최첨단의 NLP 기술을 쉽게 사용하는 것입니다.

🤗 Transformers는 이러한 사전학습 모델을 빠르게 다운로드해 특정 텍스트에 사용하고, 원하는 데이터로 fine-tuning해 커뮤니티나 우리의 모델 허브에 공유할 수 있도록 API를 제공합니다. 또한, 모델 구조를 정의하는 각 파이썬 모듈은 완전히 독립적이여서 연구 실험을 위해 손쉽게 수정할 수 있습니다.

🤗 Transformers는 가장 유명한 3개의 딥러닝 라이브러리를 지원합니다. 이들은 서로 완벽히 연동됩니다 — Jax, PyTorch, TensorFlow. 간단하게 이 라이브러리 중 하나로 모델을 학습하고, 또 다른 라이브러리로 추론을 위해 모델을 불러올 수 있습니다.

허깅페이스 트랜스포머스

트랜스포머스는 다양한 추상화 수준에서 라이브러와 상호작용하도록 계층화된 API를 제공합니다.

이번 시간에는 원시 텍스트를 미세 튜닝된 모델의 예측으로 변환하기 위해 필요한 모든 단계를 추상화 하는 파이프라인을 사용하겠습니다.

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

텍스트가 고객 피드백이라면 아마도 긍정적인 피드백인지 부정적인 피드백인지 알고 싶을 것입니다 이러한 작업을 감정분석 이라고 합니다. 여기서는 트랜스포머를 사용해 예시 텍스트의 감성을 분류하는 방법을 알아보겠습니다.

허깅페이스 트랜스포머스

```
from transformers import pipeline
# pip install transformers

# pipeline() 함수를 호출하면서 관심 작업 이름을 전달해 파이프라인 객체를 생성
classifier = pipeline("text-classification")

# 처음 이 코드를 실행하면 파이프라인이 자동으로 허깅페이스 허브에서 모델 가중치 다운로드 합니다.
# 파이프라인 객체를 다시 만들 때는 가중치가 이미 다운로드됐으므로 캐싱된 버전을 사용한다는 안내 메시지가 나옵니다.
# 기본적으로 txt-classification 파이프라인은 감성 분석을 위해 설계된 모델을 사용하지만, 다중 분류와 다중 레이블 분류도 지원합니다.

# 파이프 라인이 준비됐으니 예측을 만들기 ! 각 파이프라인은 텍스트 문자열(또는 문자열의 리스트)를 입력으로 받고 예측 리스트를 반환 합니다.
# 각 예측은 하나의 파이썬 딕셔너리 형태로 반환
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""

outputs = classifier(text)
for output in outputs :
    print(output["label"], output["score"])

# 모델은 텍스트가 부정적이라고 확신합니다. 감성 분석 작업에서 파이프라인은 POSITIVE 와 NEGATIVE 레이블중 하나를 반환
```

허깅페이스 트랜스포머스

```
# 개체명 인식
# 고객 피드백의 감성을 예측하는 일은 분석의 좋은 출발점이 되지만, 피드백이 특정 제품과 서비스 중 무엇에 대한 것인지 알고 싶을 때가 많습니다.
# NLP에서는 제품, 장소, 사람과 같은 실제 객체를 개체명이라고 하고 이런 개체명을 텍스트에서 추출하는 작업을 개체명 인식이라고 합니다
# 해당 파이프라인을 로드하고 고객 리뷰를 전달해 NER을 적용해보겠습니다.

ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
temp = pd.DataFrame(outputs)
print(temp)
```

	entity_group	score	word	start	end
0	ORG	0.879010	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556570	Mega	208	212
4	PER	0.590256	##tron	212	216
5	ORG	0.669693	Decept	253	259
6	MISC	0.498349	##icons	259	264
7	MISC	0.775362	Megatron	350	358
8	MISC	0.987854	Optimus Prime	367	380
9	PER	0.812096	Bumblebee	502	511

```
"""
이 파이프라인은 모든 개체명을 감지하고 ORG(조직), LOC(위치), PER(사람) 같은 카테고리에 할당했습니다.
이 예에서는 모델 예측에 따라 단어를 그룹화하기 위해 aggregation_strategy 매개변수를 사용했습니다.
점수는 모델이 개체명을 얼마나 확신하는지 나타냅니다.

앞의 표시에서 word 열에 있는 해시 기호(#)가 이상하지 않나요 ? 이는 모델의 토크나이저가 생성한 것입니다.
토크나이저는 단어를 토큰이라는 기본 단위로 분할합니다.
"""
```

허깅페이스 트랜스포머스

```
# 텍스트에 있는 모든 개체명이 잘 추출됐습니다. 하지만 이따금 더 구체적인 질문을 하고 싶을때가 있습니다.
# 이를 위해 질문 답변을 사용하겠습니다.
# 방식은 : 텍스트 구절과 함께 답을 얻고 싶은 질문을 모델에 전달하고, 모델은 답변 텍스트를 반환합니다.
reder = pipeline("question-answering")
question = "What dose the customer want ?"
outputs = reder(question=question, context=text)
temp1 = pd.DataFrame([outputs])
print(temp1)
"""
이 예제의 경우 답변을 텍스트에서 직접 추출하기 때문에 추출적 질문 답변이라 합니다.
이런식으로 고객 피드백에서 관련 정보를 빠르게 추출합니다. 하지만 정확하게 늘어놓은 불평이 산더미처럼 쌓여 이를 전부 읽는 시간이 없다면 어떻게 할까요 ?
>> 요약이라는 방법이 있습니다.
"""
```

```
      score  start  end                    answer
0  0.387483    335  358  an exchange of Megatron
```

허깅페이스 트랜스포머스

```
# 텍스트 요약 : 긴 텍스트를 입력으로 받고 관련 사실이 모두 포함된 간단한 버전을 생성하는 것입니다.
# 모델이 논리적인 텍스트를 생성해야 하므로 이전 문제보다 훨씬 더 복잡한 작업입니다.
# 요약 파이프라인 만들기
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=60, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])

"""
원본 텍스트의 일부가 복사 모델이 문제의 핵심을 감지하고 (원본텍스트의 마지막에 등장하는) "Bumblebee" 불만을 제기한 사람이라는 것을 정확하게 구분했습니다.
max_length, clean_up_tokenization_spaces 키워드 매개변수를 사용 이런 매개변수를 통해 실행 시점에 출력을 조정합니다.

"""
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead. As a lifelong enemy of the Decepticons, I hope you can understand

허깅페이스 트랜스포머스

```
# 번역 : 요약과 마찬가지로, 번역 또한 텍스트를 생성해 출력하는 작업입니다. 번역 파이프라인을 사용해보기
# https://huggingface.co/docs/transformers/main/en/main_classes/pipelines#transformers.TranslationPipeline
# https://huggingface.co/models?pipeline_tag=translation&sort=downloads&search=opus-mt-en-ko
# pip install sentencepiece

from transformers import pipeline
pipe = pipeline("translation", model="Helsinki-NLP/opus-mt-tc-big-en-ko")
print(pipe(text))
```

허깅페이스 트랜스포머스

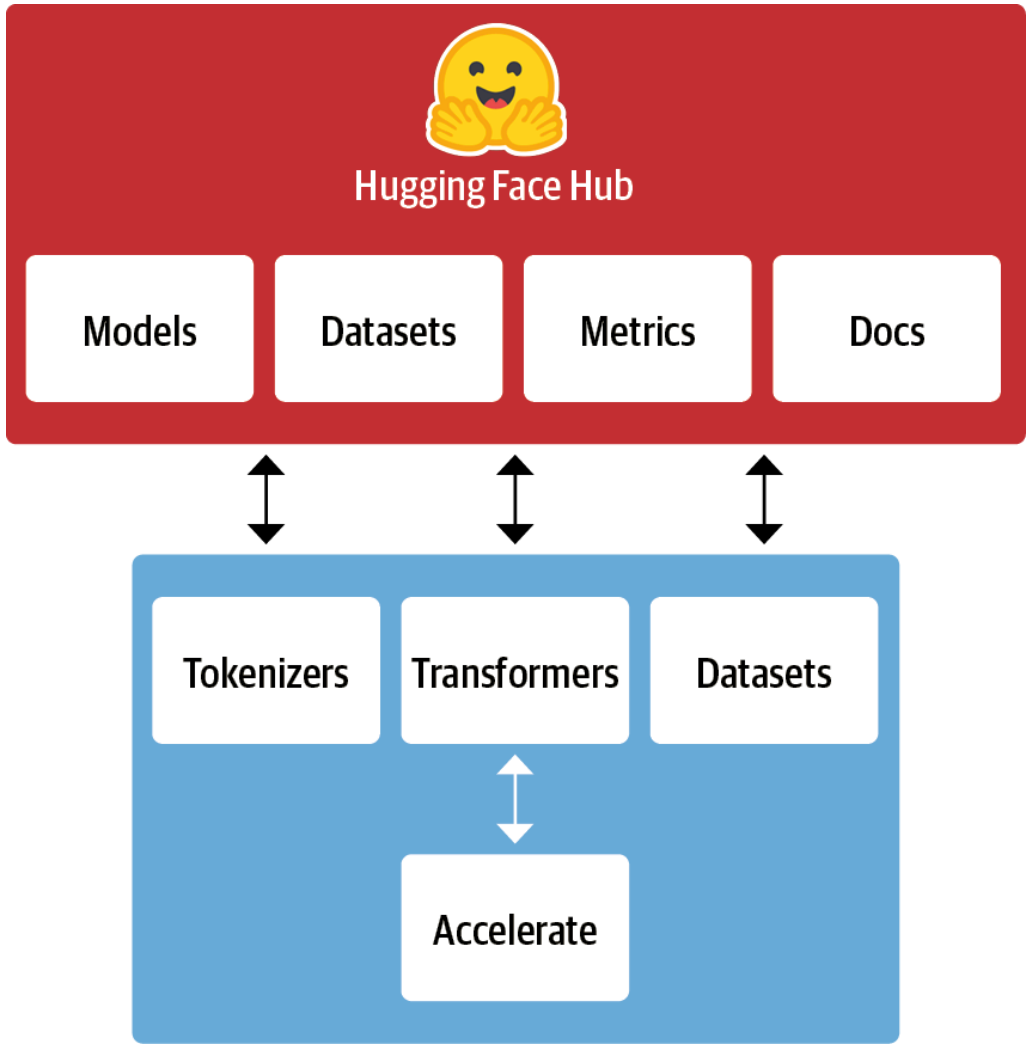
```
"""
텍스트 생성
"""
from transformers import set_seed
set_seed(42) # 동일 결과를 재현하기 위해 지정

generator = pipeline("text-generation")
response = "Dear Bumblebee, I am sorry to hear that your order was mixed up."
prompt = text + "\n\nCustomer service response:\n" + response
outputs = generator(prompt, max_length=200)
print(outputs[0]['generated_text'])

""" 범블비 형은 아직 화가 잔뜩입니다. """
```

```
Customer service response:
Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was
completely mislabeled, which is very common in our online store, but I can
appreciate it because it was my understanding from this site and our customer
service of the previous day that your order was not made correct in our mind and
that we are in a process of resolving this matter. We can assure you that your
order
```

허깅페이스 생태계

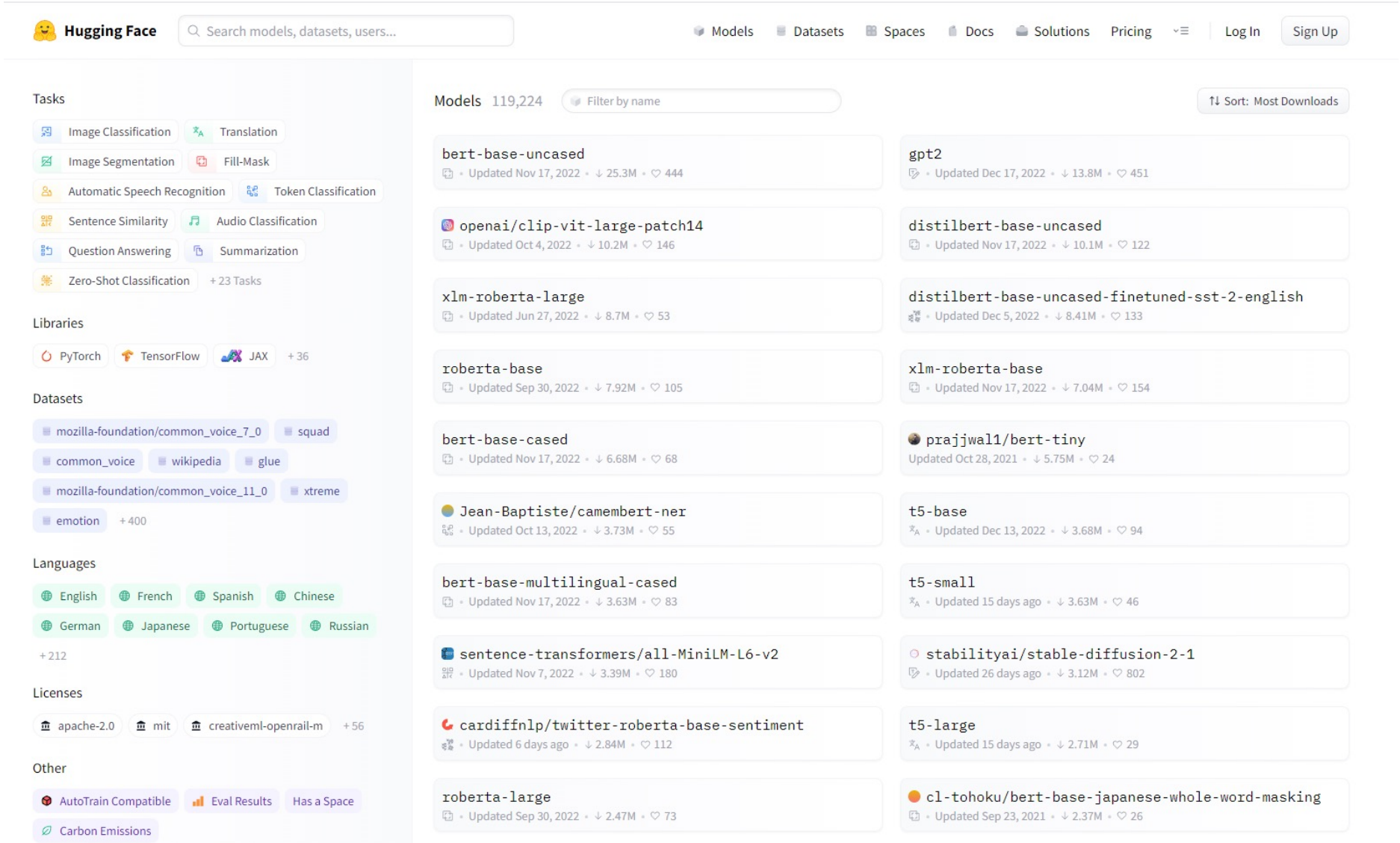


트랜스포머는 NLP와 머신러닝 프로젝트의 속도를 높이는 다양한 라이브러리와 도구를 갖춘 생태계로 빠르게 성정했습니다.

허킹페이스 생태계는 크게 라이브러리와 허브로 구성

라이브러리는 코드를 제공하고 허브는 사전 훈련된 모델 가중치, 데이터셋, 평가 지표를 위한 스크립트 등을 제공합니다.

허깅페이스 생태계


Hugging Face
Search models, datasets, users...
Models
Datasets
Spaces
Docs
Solutions
Pricing
Log In
Sign Up
Tasks
Image Classification
Translation
Image Segmentation
Fill-Mask
Automatic Speech Recognition
Token Classification
Sentence Similarity
Audio Classification
Question Answering
Summarization
Zero-Shot Classification
+ 23 Tasks
Libraries
PyTorch
TensorFlow
JAX
+ 36
Datasets
mozilla-foundation/common_voice_7_0
squad
common_voice
wikipedia
glue
mozilla-foundation/common_voice_11_0
xtreme
emotion
+ 400
Languages
English
French
Spanish
Chinese
German
Japanese
Portuguese
Russian
+ 212
Licenses
apache-2.0
mit
creativeml-openrail-m
+ 56
Other
AutoTrain Compatible
Eval Results
Has a Space
Carbon Emissions
Models 119,224
Filter by name
Sort: Most Downloads
bert-base-uncased
Updated Nov 17, 2022 • ↓ 25.3M • ♡ 444
gpt2
Updated Dec 17, 2022 • ↓ 13.8M • ♡ 451
openai/clip-vit-large-patch14
Updated Oct 4, 2022 • ↓ 10.2M • ♡ 146
distilbert-base-uncased
Updated Nov 17, 2022 • ↓ 10.1M • ♡ 122
xlm-roberta-large
Updated Jun 27, 2022 • ↓ 8.7M • ♡ 53
distilbert-base-uncased-finetuned-sst-2-english
Updated Dec 5, 2022 • ↓ 8.41M • ♡ 133
roberta-base
Updated Sep 30, 2022 • ↓ 7.92M • ♡ 105
xlm-roberta-base
Updated Nov 17, 2022 • ↓ 7.04M • ♡ 154
bert-base-cased
Updated Nov 17, 2022 • ↓ 6.68M • ♡ 68
prajjwal1/bert-tiny
Updated Oct 28, 2021 • ↓ 5.75M • ♡ 24
Jean-Baptiste/camembert-ner
Updated Oct 13, 2022 • ↓ 3.73M • ♡ 55
t5-base
Updated Dec 13, 2022 • ↓ 3.68M • ♡ 94
bert-base-multilingual-cased
Updated Nov 17, 2022 • ↓ 3.63M • ♡ 83
t5-small
Updated 15 days ago • ↓ 3.63M • ♡ 46
sentence-transformers/all-MiniLM-L6-v2
Updated Nov 7, 2022 • ↓ 3.39M • ♡ 180
stabilityai/stable-diffusion-2-1
Updated 26 days ago • ↓ 3.12M • ♡ 802
cardiffnlp/twitter-roberta-base-sentiment
Updated 6 days ago • ↓ 2.84M • ♡ 112
t5-large
Updated 15 days ago • ↓ 2.71M • ♡ 29
roberta-large
Updated Sep 30, 2022 • ↓ 2.47M • ♡ 73
cl-tohoku/bert-base-japanese-whole-word-masking
Updated Sep 23, 2021 • ↓ 2.37M • ♡ 26

트랜스포머의 주요 도전 과제

언어
- NLP 연구에 사용된 언어는 거의 영어입니다. 그 외 언어를 위한 모델도 있지만 데이터가 거의 없거나 소량이어서 사전 훈련된 모델을 찾기가 어렵습니다.

데이터 가용성
- 모델에 필요한 레이블링된 훈련 데이터의 양은 전이 학습을 사용하면 크게 줄지만 사람이 작업을 수행하는데 필요한 양에 비하면 여전히 많습니다.

긴 문서 처리하기
- 셀프 어텐션은 텍스트 길이가 문단 정도 될 때 잘 작동합니다. 하지만 문서와 같이 긴 텍스트에 사용하려면 비용이 많이 듭니다.

불투명성
- 트랜스포머는 다른 딥러닝 모델처럼 대부분 투명하지 않습니다. 모델이 그렇게 예측한 이유를 설명하기 힘듭니다.

편향
- 트랜스포머 모델은 주로 인터넷의 텍스트 데이터를 사용해 사전 훈련을 합니다. 따라서 이런 데이터에 있는 편향이 모델에 고스란히 전이됩니다. 인종차별, 성차별 또는 더 나쁜 편향이 있는지 확인 불가


감사합니다.