

Natural Language Processing Introduction to Transformers

Fall, 2024 (2nd week)

School of AI Convergence

Jangmin Oh

Transformers

Self-Attention 메커니즘.

: 문장 내의 단어가 다른 단어들과 어떻게 연관되는지를 계산하는 방식.

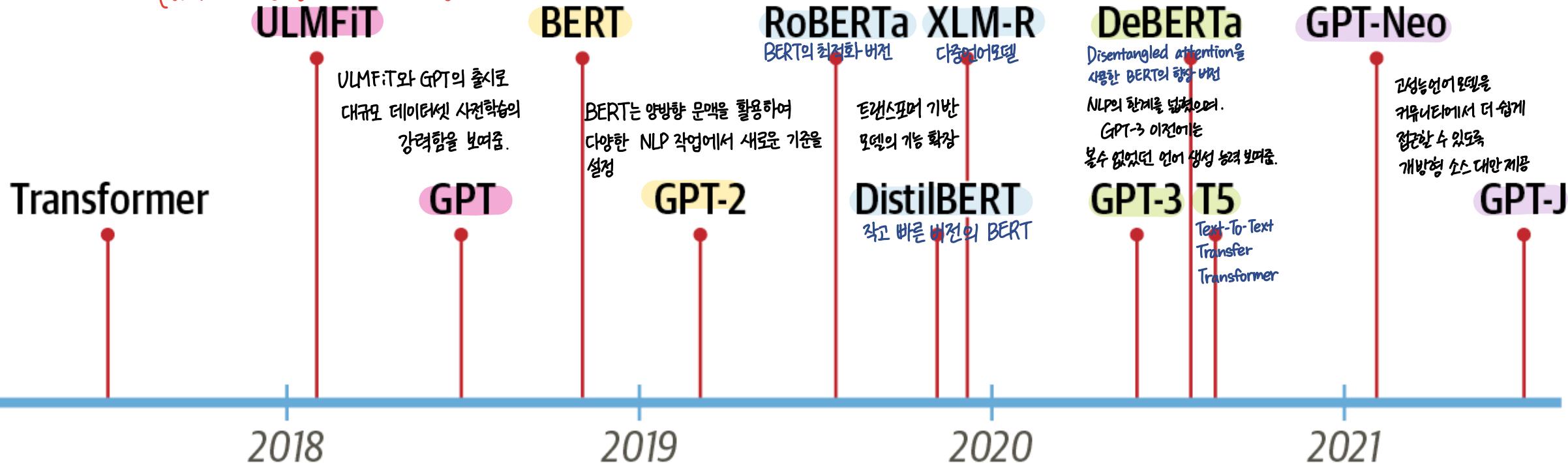
자연어 처리 분야에서 중요한 전환점

- Ashish Vaswani et. al., Attention Is All You Need, 2017 (Google)

(RNN)

트랜스포머 모델의 도입으로, 전통적인 순환 신경망 대신 셀프 어텐션 메커니즘을 사용함으로써 더 효율적이고 확장 가능한 언어 모델 학습이 가능해짐.

(Universal Language Model Fine-tuning).



Transformers

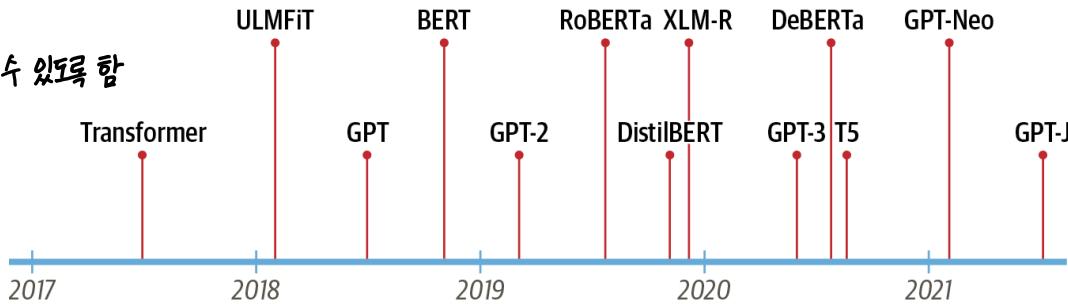
Self Attention 메커니즘을 활용하여

문장 내 단어 간의 종속성을 더 효과적으로 포착할 수 있도록 함

전통적인 순환신경망(RNN)보다 훨씬 더 우수한 성능 보여줌.

Transformer architecture

→ 트랜스포머 모델



모델 아니고
전이학습 이용하는 기법

- A. Vaswani et al., Attention is All You Need, 2017

• A novel architecture: Surpasses Recurrent Neural Networks (RNNs) in terms of machine translation quality and training costs

ULMFit

트랜스포머는 번역 품질을 향상시켰을 뿐만 아니라, 병렬로 데이터를 처리할 수 있는 능력 덕분에 학습 비용도 절감할 수 있었음.

- J. Howard et al., Universal Language Model Fine-Tuning for Text Classification, 2018

• Created language models from diverse corpora, then fine-tuned on small amounts of labeled data to achieve state-of-the-art performance

- Became a catalyst for subsequent developments such as GPT, BERT, etc.

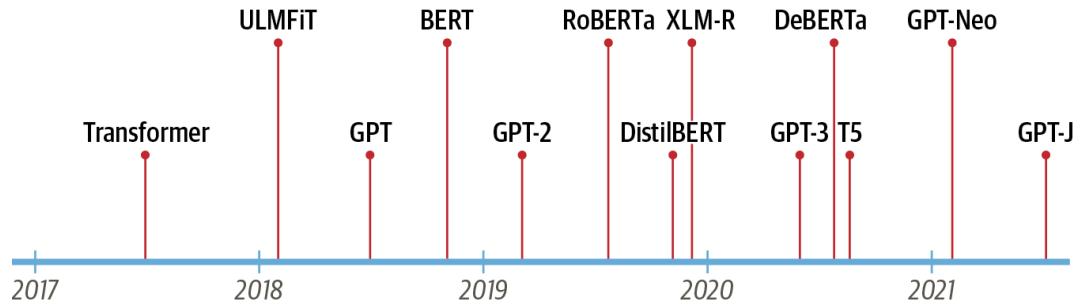
트랜스포머 모델의 이러한 혁신과 ULMFiT 같은 기법의 개발은 GPT와 BERT 같은 강력한 모델들의 개발로

이어지는 촉매 역할을 함.

(NLP에서 전이학습의 중요성 강조
한 작업이나 도메인에서
얻은 지식을 다른 작업에 적용하여
모델의 효율성과 효과성을 크게 향상
시킬 수 있음을 시사)

Transformers

GPT는 트랜스포머 아키텍처를 활용하여 생성적 사전학습이라는 새로운 훈련 접근 방식을 사용



- GPT
 - A. Radford et al., Improving Language Understanding by Generative Pre-Training, 2018 대규모 텍스트 데이터 코퍼스를 비지도 방식으로 학습시켜 모델이 특정 작업에 대해 미세 조정되기 전에 언어의 기본 구조를 학습할 수 있도록 함. GPT의 성공은 트랜스포머와 비지도 학습 기법을 결합하는 것이 얼마나 강력한지 보여주었음.
- BERT (*Bidirectional Encoder Representations from Transformers*)
 - J. Delvin et al., BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding, 2018 BERT는 양방향 접근 방식을 활용하여 단어 주변의 양쪽 방향에서 문맥을 고려할 수 있게 함. 언어의 의미와 뉘앙스를 이해하는데 중요한 강점 제공.
- Combined Transformer architecture with unsupervised learning to train language models GPT와 BERT 모두 트랜스포머 아키텍처의 강점과 비지도 사전 학습의 이점 결합
- Surpassed state-of-the-art performance on most NLP benchmarks

Encoder-Decoder Framework

트랜스포머가 등장하기 전, 자연어 처리의 최신 모델은 LSTM과 같은 순환신경망(RNN) 아키텍처를 기반으로 함.

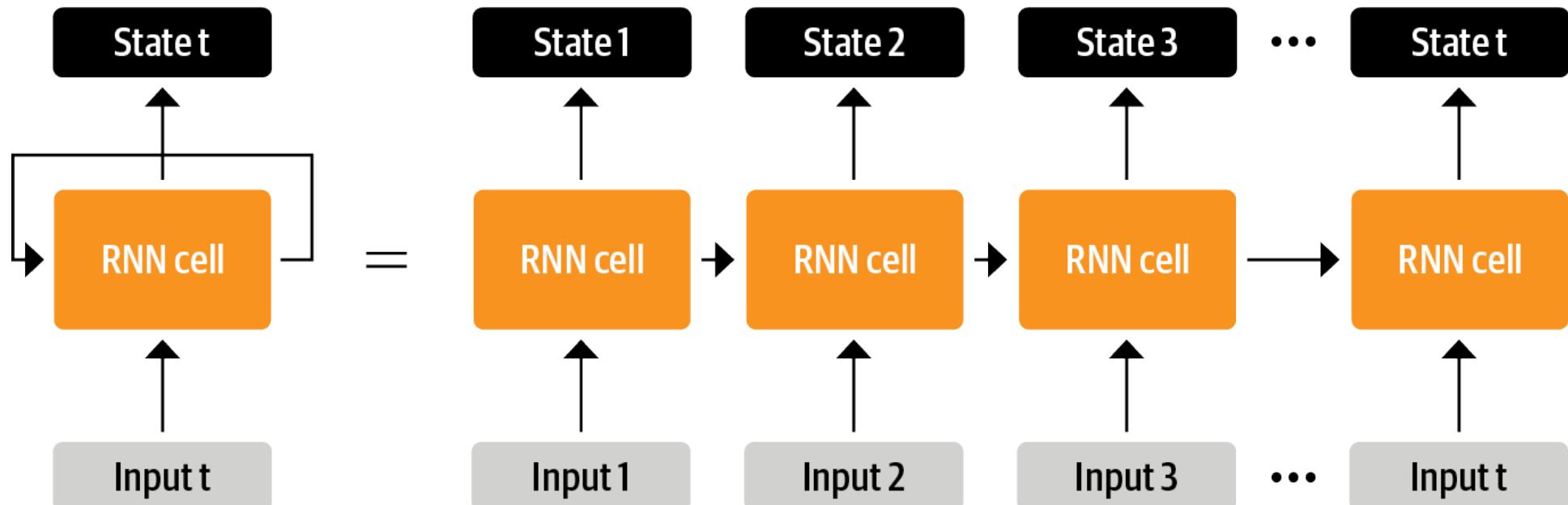
• State-of-the-Art in NLP Before Transformers: RNN Architectures

Like LSTM

각 단계의 상태 정보가 시퀀스의 다음 단계로 전달됨. RNN 기반 모델은 이전 단계의 정보를 추적하여 다음 단계에 대한 예측을 내리는 데 사용

- The state information is passed to the next step
- Tracking information across steps

이러한 순차적 처리를 통해 데이터의 시간적 종속성을 효과적으로 포착.



Encoder-Decoder Framework

인코더-디코더 프레임워크는 RNN에서 중요한 역할.

이 프레임워크에서 인코더의 역할은 입력 시퀀스를 처리하고 이를 고정 크기의 컨텍스트 벡터로 요약하는 것. 이 컨텍스트 벡터는 종종 인코더의 마지막 은닉 상태로 표현되며.

- The Role of RNNs in Machine Translation Systems

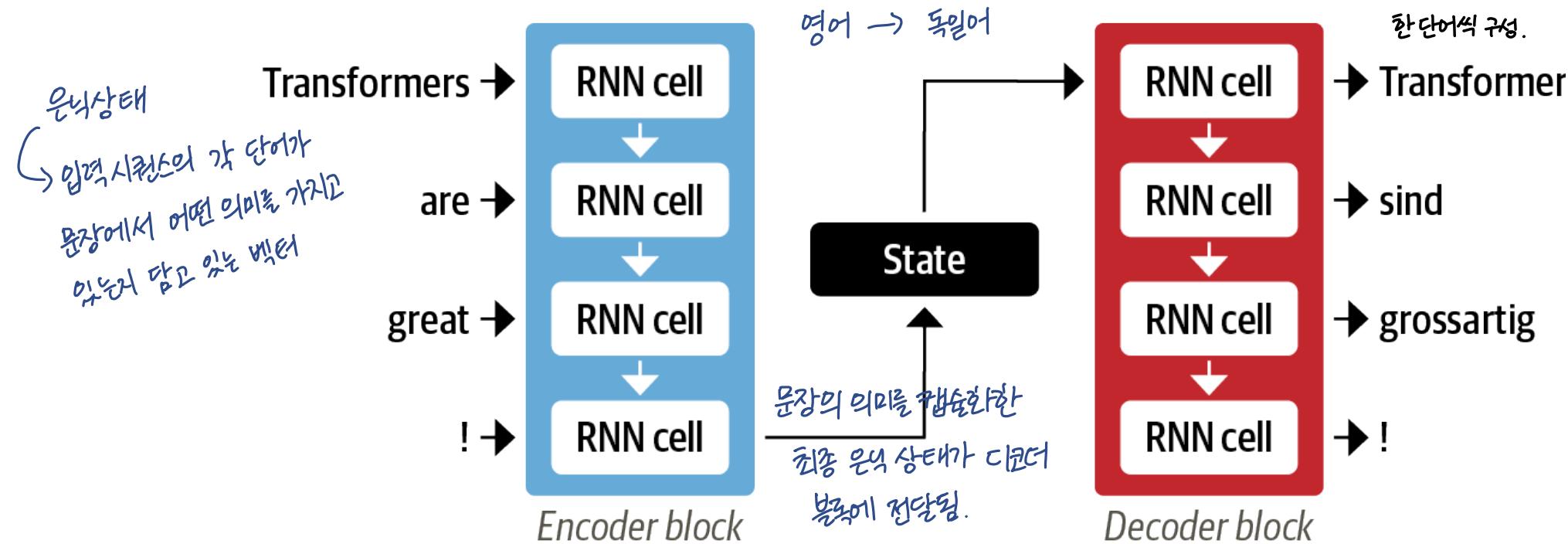
- Encoding and Decoding

- Last hidden state: a summary of the input sequence
- is used as input to the decoder

입력 시퀀스의 본질을 효과적으로 캡처하여
디코더가 사용할 수 있는 형태로 압축.



인코딩 과정이 완료되면 디코더가 그 역할 이어받음.
디코더는 인코더의 마지막 은닉 상태로부터 전달된 컨텍스트
벡터를 입력으로 사용하여 출력 시퀀스 생성.
디코더는 이 정보를 순차적으로 처리하여 번역된 문장을
한 단어씩 구성.



Encoder-Decoder Framework

RNN 기반 인코더-디코더 프레임워크는 효과적이지만, 정보 병목 현상과 같은 약점 있음. 인코더의 최종 은닉 상태는 입력 시퀀스의 모든 정보를 요약해야 하는데,

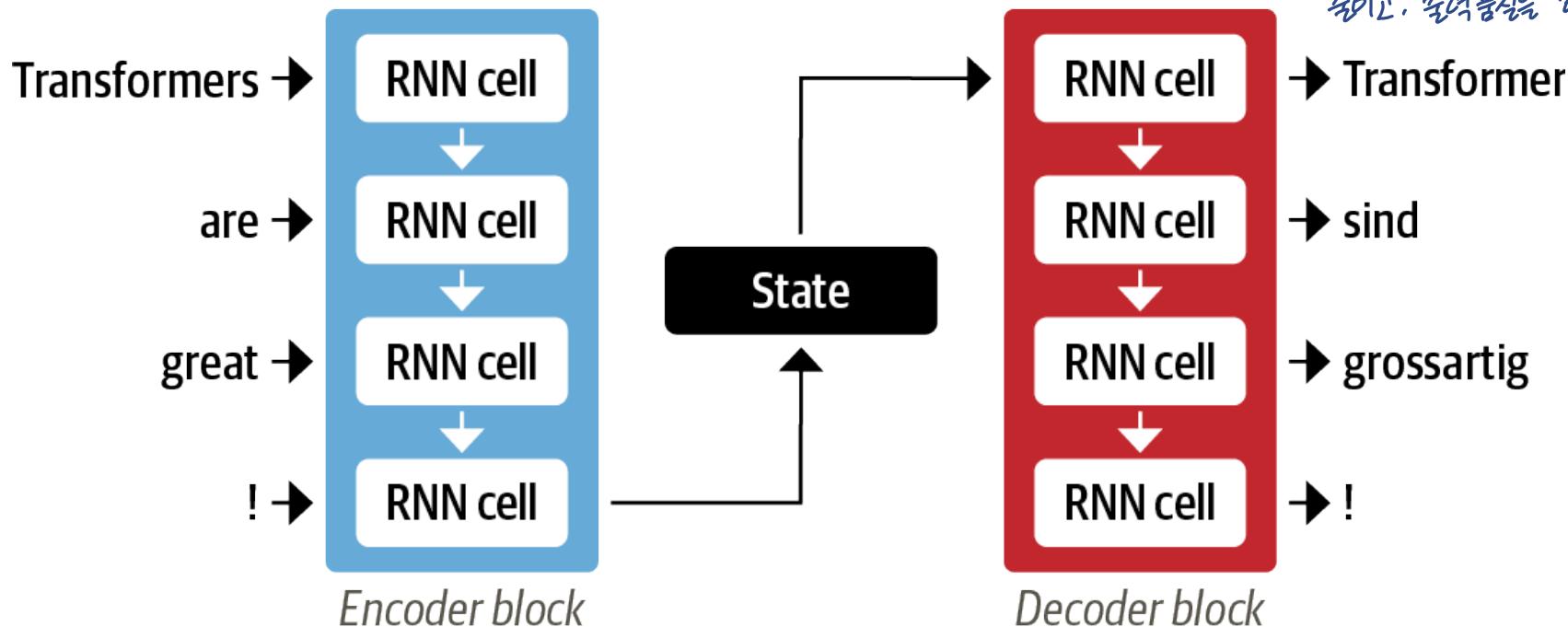
• RNN Weakness #1: Information Bottleneck

이는 정보 손실을 초래할 수 있음.

특히 입력 시퀀스가 길거나, 복잡하면, 중요한 정보가 최종 은닉 상태에 모두 담기기 어려움. 출력 품질 ↓

- The final hidden state of the encoder should summarize all the information.
- Inevitable information loss

이러한 한계 때문에 셀프 어텐션 메커니즘을 사용하는 트랜스포머와 같은 대체 아키텍처가 개발됨. 트랜스포머는 입력의 다양한 부분에 동시에 주의를 기울여 정보 병목을 줄이고, 출력 품질을 향상시킴.

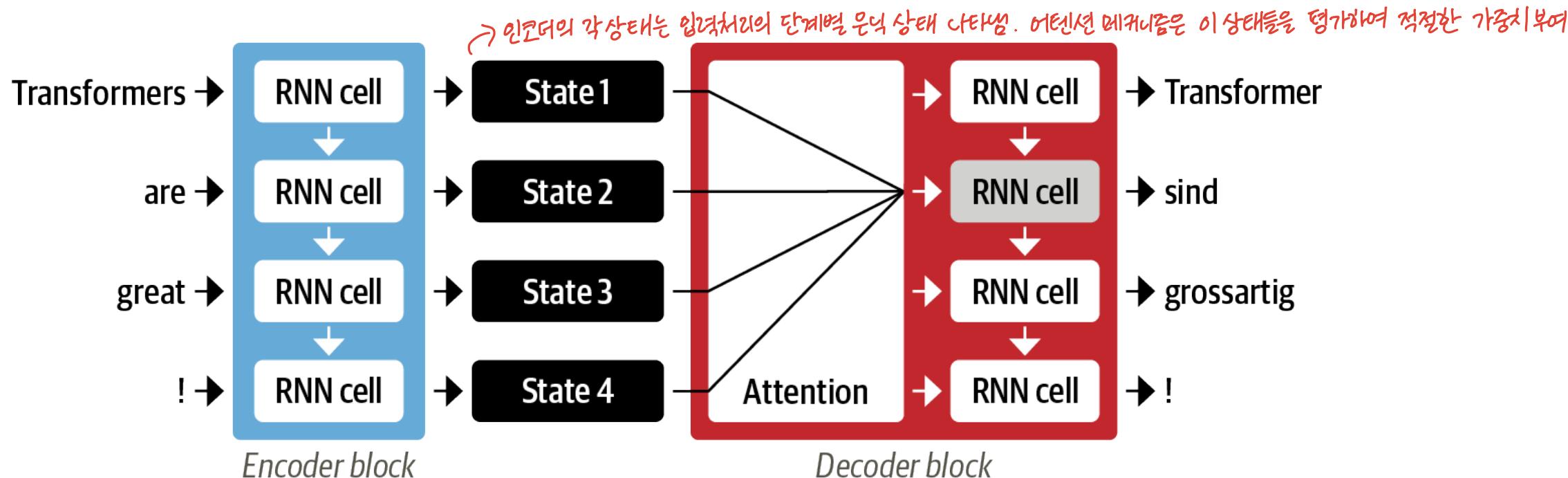


Attention Mechanism

기존의 마지막 은닉 상태에 의존하는 대신, 어텐션 메커니즘은 인코더의 모든 은닉 상태를 활용하여 모델이 출력의 각 단어를 생성할 때 입력의 다양한 부분을 고려할 수 있게 함.

- Uses hidden states from all encoder steps
- Assigns weights (attention) to encoder outputs

어텐션 메커니즘은 각 은닉 상태에 가중치를 할당해 디코딩 단계에서 입력 시퀀스의 중요한 부분에 집중할 수 있도록 도움.



이 방식은 더 많은 문맥을 유지하고, 정보 손실을 줄여, 보다 정확하고 일관된 결과 제공.

2024. 9. 4. 어텐션 메커니즘은 NLP에서 중요한 발전을 이룬다. 트랜스포머와 같은 모델이 다양한 작업에서 뛰어난 성과를 내는데 기여.
AI융합학부

Attention Mechanism

기계 번역의 맥락에서 어텐션 메커니즘은 소스 문장과 대상 문장의 단어 간의 정렬을 학습함으로써 중요한 역할 함.

소스 문장의 하든 스테이트에 어텐션 가중치를 할당하여 모델이 번역된 문장의 각 단어를 생성할 때 가장 관련성이 높은 부분에 집중할 수 있도록 함으로써 달성.

- Attention Mechanism in Machine Translation

- Learns alignment between words in source and target sentences.
- Example: Visualization of attention in English-French translation.
- Bright areas indicate strong attention (e.g., "zone" aligns with "Area").

- RNN Weakness 2: Sequential Operations

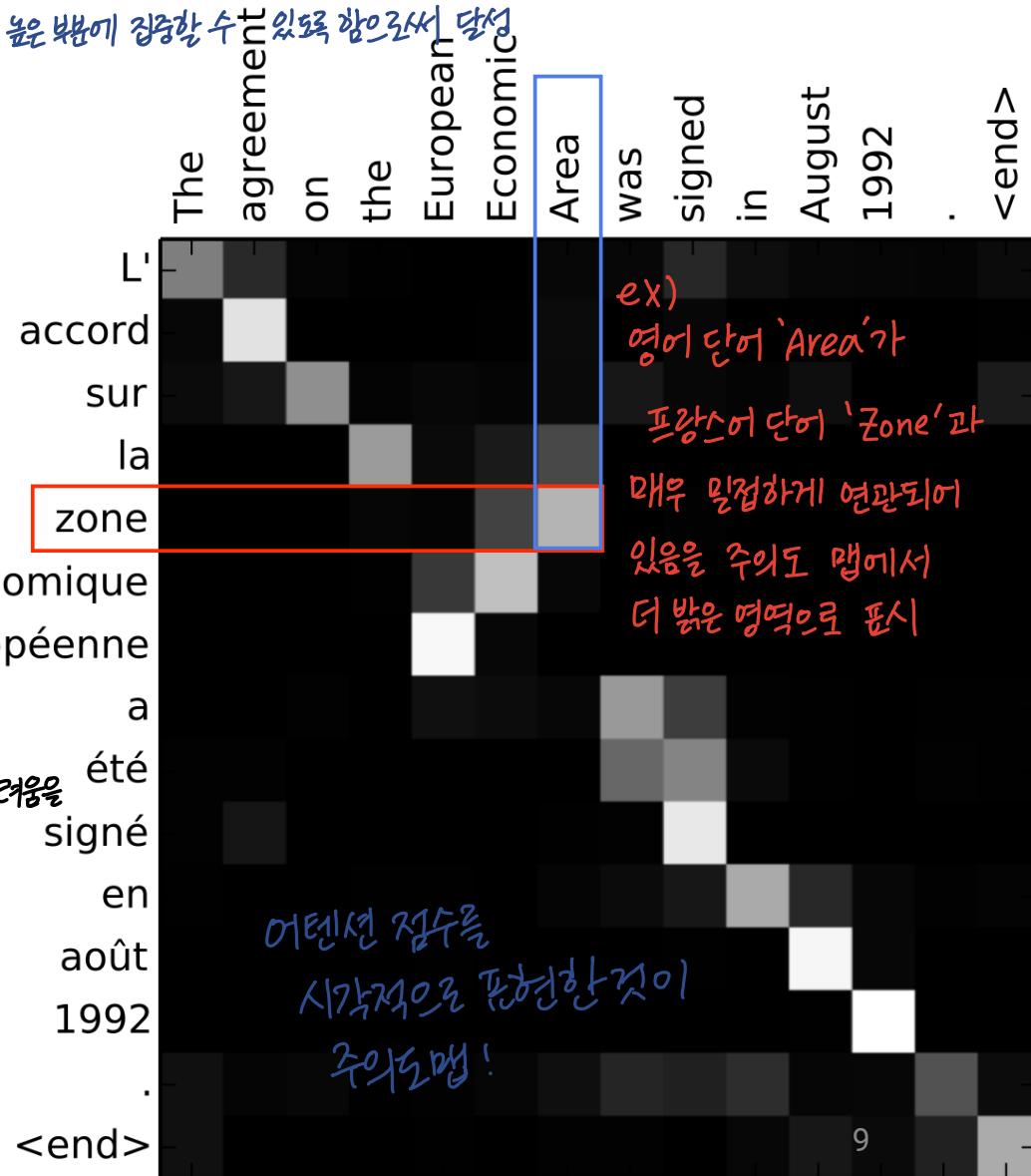
겪는다는 점.

- Difficulty in parallelizing computations over the entire sequence.

RNN의 각 단계는 이전 단계에 의존하기 때문에 단계를 병렬로 실행하여 처리 속도를 높이기가 어려움. 이러한 한계는 특히 광범위한 계산 리소스가 필요한 긴 시퀀스나 대규모 데이터 세트를 처리할 때 RNN의 효율성을 떨어뜨림.

→ source sentence : 번역을 하기 전의 원문
→ target sentence : 번역된 문장

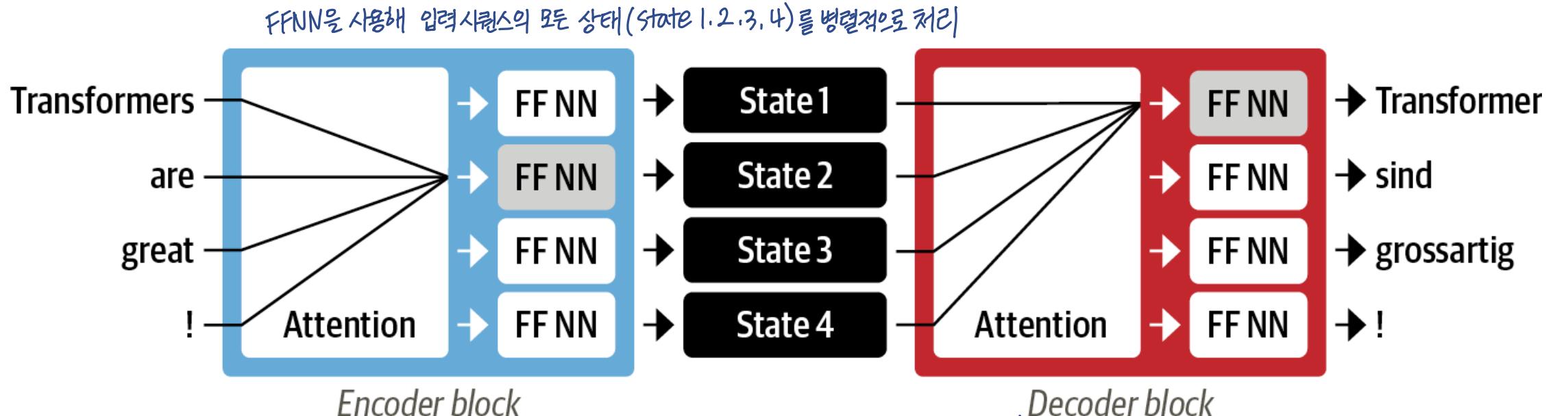
ex) 영어문장 : The cat sits on the mat.
한국어번역: 고양이가 매트 위에 앉아 있다.
이때 The cat은 고양이 sits는 앉아있다, on the mat는 매트 위에와 대응된다.
어떤 단어들이 서로 매칭되어 번역 과정에서 연결되는지를 학습함.



Self-Attention Mechanism

셀프 어텐션 메커니즘은 RNN의 핵심인 순환구조의 필요성을 없앰. 셀프 어텐션은 정보를 순차적으로 처리하는 대신 입력 시퀀스의 모든 위치를 동시에 고려할 수 있게 해줌. 이는 동일한 층에 있는 모든 은닉 상태들에 대해 어텐션 점수를 계산함으로써 이루어지며, 시퀀스 내의 다른 단어들과의 관계에서 각 단어의 중요성을 평가할 수 있게 함.

- Removes need for recurrent structure.
- Operates attention over all states in the same layer.
- Uses Feed Forward Neural Network (FF NN).
- Enables parallel processing: significant innovation in NLP.



셀프 어텐션은 이러한 어텐션 가중치가 적용된 표현을 추가로 처리하기 위해 주로 피드 포워드 신경망(FF NN)을 사용.

2024. 9. 4. 셀프 어텐션 주요 장점 중 하나는 병렬 처리를 지원할 수 있다. → 통합학부 이는 더 빠른 학습 시간과 더 긴 시퀀스를 효과적으로 처리할 수 있는 능력을 의미.

Transfer Learning in NLP

전이 학습은 컴퓨터 비전 분야에서의 성공을 바탕으로 자연어 처리에서 중요한 방법론이 됨. 전통적인 지도 학습에서는 각 도메인마다 모델을 별도로 학습해야 하며, 각 작업마다 많은 양의 레이블된 데이터가 필요. 반면 전이 학습은 모델이 먼저 크고 정보가 풍부한 도메인에서 사전 학습된 후, 적은 양의 레이블된 데이터로 특정 작업에 대해 미세 조정될 수 있게 합니다.

• Transfer Learning in Vision

- Traditional supervised learning: separate training for each domain.
- Transfer learning: model trained on a rich domain, then fine-tuned on a new domain.
- Example: ImageNet pretraining for specific flower classification.
- Benefits: Often achieves higher accuracy than supervised learning with the same amount of labeled data.

ex) 컴퓨터 비전에서 ImageNet으로 학습된 모델은 수백만 개의 이미지로부터 다양한 특징을 학습.

그런 다음 이러한 모델은 특정 꽃의 품종을 분류하는 것과 같은 작업에 특화되도록 파인튜닝될 수 있음.

이 방식은 시간과 자원을 절약할 뿐만 아니라, 동일한 양의 레이블된 데이터로 처음부터 모델을 학습하는 것보다 더 높은 정확도를 달성하는 경우가 多

Transfer Learning in NLP

전이 학습은 전통적인 지도 학습과 대비되는 NLP에서 강력한 접근 방식.

지도학습에서는 각 특정 도메인에 대해 모델을 독립적으로 학습시키며, 각 작업마다 많은 양의 레이블된 데이터 필요. But 전이학습은 정보가 풍부한 도메인에서 사전 학습된 모델을 최소한의 추가 학습으로

- **Supervised Learning:**

- Separate training for each domain.

- **Transfer Learning:**

- Pre-trained on a rich domain.
- Fine-tune the model body for new domain.
- Head actively tuned, body slightly tuned.

- **Example: ImageNet**

- Pretraining on millions of images to model general features.
- Fine-tuning for specific tasks (e.g., flower species classification).

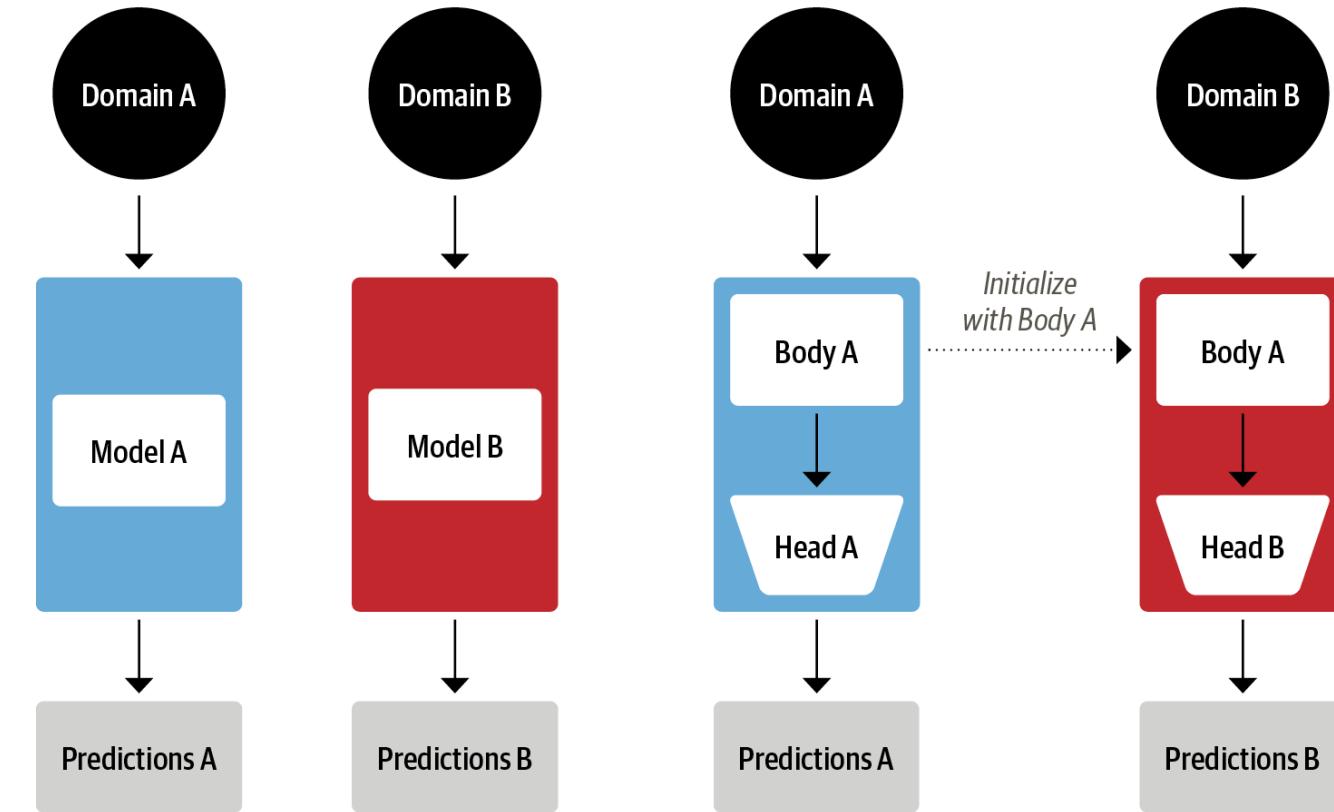
- **Advantages:**

- Higher accuracy with the same amount of labeled data compared to supervised learning.

전이 학습은 동일한 양의 레이블된 데이터로 전통적인 지도 학습에 비해 더 높은 정확도를 달성하는 경우가 多

Training and evaluation on
the same task/domain

새로운 작업이나 도메인에 적용할 수 있게 해줌
Extract knowledge from source task,
and apply to different target task



전이 학습 과정에서, 원래 학습된 일반 지식을 포함하고 있는 모델의 바디를 시작점으로 사용

Supervised learning

그런 다음 이 모델을 새로운 도메인에 맞게 조정하는데, 모델의 헤드(특정 작업 출력을 담당하는 부분)는 적극적으로 튜닝되고, 바디는 약간만 조정됨.

Transfer Learning in NLP

2018년 ULMFiT 모델이 전이 학습을 효과적으로 도입. 전이 학습은 주로 사전 학습, 도메인 적응, 파인튜닝의 3단계로 이루어짐.

• Introduction

전이 학습 순서 : ① 사전 학습 → ② 도메인 적응 → ③ 파인튜닝

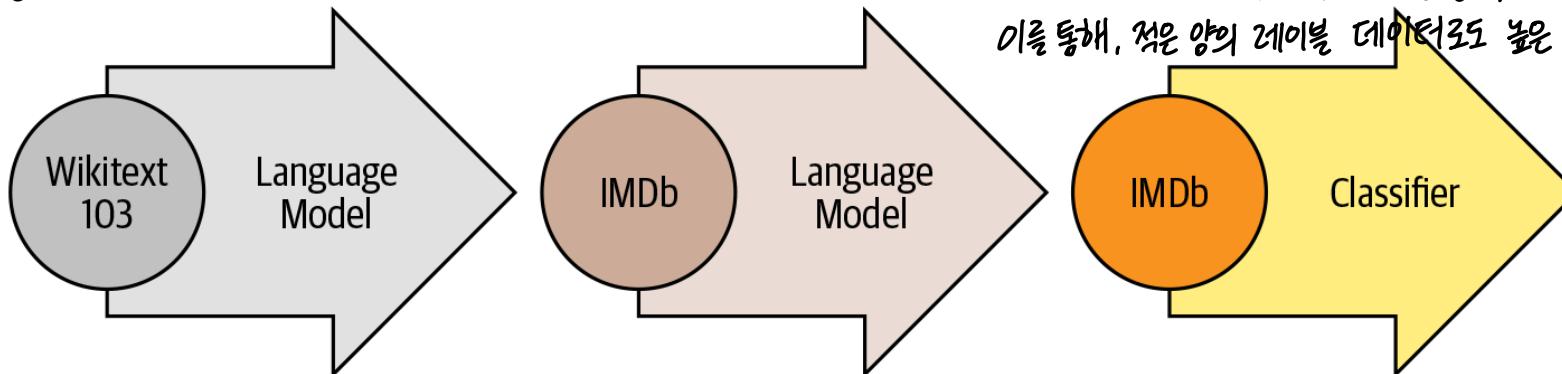
- ULMFiT (2018) introduced the concept of transfer learning in NLP.

• Three steps of Transfer Learning ① 사전 학습에서는 언어 모델이 대규모 공개 데이터셋으로 다음 단어를 예측하도록 학습해 일반적인 언어 패턴을 학습합니다.

- Pretraining (Language Modeling): Predict the next word based on previous ones (using accessible text datasets like Wikipedia).
② 도메인 적응 단계에서는 특정 도메인의 데이터로 모델을 추가 학습해, 해당 도메인의 특유한 언어 특성과 용어를 더 잘 이해하도록 합니다.
- Domain Adaptation: Perform language modeling on domain-specific corpora if sufficiently large.
- Fine-Tuning: Apply a separate classification head for specific tasks (e.g., movie review sentiment classification).

③ 파인튜닝 단계에서는 분류 헤드를 추가하여 모델을 특정 작업에 맞게 조정. 예를 들어, 영화 리뷰의 감정을 긍정적 또는 부정적으로 분류하는데 사용됩니다.

이를 통해, 적은 양의 레이블 데이터로도 높은 성능을 발휘할 수 있습니다.



최신 딥러닝 연구 성과를 활용할 수 있도록 설계된 라이브러리로 사용 간편

Hugging Face transformers

- Utilizing results from recent deep learning research:

- Model architecture coding (PyTorch, TensorFlow).
- Download pre-trained model weights. → 학습과정 가속화 위해 사전 훈련된 모델 가중치 다운로드
- Preprocess input -> Feed to model -> Postprocess output.
- Define dataloader, loss function, optimizer. → 데이터로더(데이터 공급처리), 손실함수(모델 오류측정), 옵티마이저(모델 가중치 업데이트)는 효과적인 모델 학습과 파인튜닝을 위해 필수적으로 정의되어야 하는 요소들

허깅페이스 사용할 때에는 입력 데이터를 전처리하고 모델에 전달.

모델이 입력을 처리한 후 출력을 후처리하여 해석 or 추가 응용에 적합하게 만듬.

- Hugging Face:

- Provides standardized interfaces for various transformer models.
- Offers tools and code for model application (easy switch between PyTorch and TensorFlow).
- Supplies model heads suited to different task types.

다양한 자연어 처리 작업을 유연하게 처리할 수 있는 기능 제공

사용자가 다양한
아키텍처와 작업을
실험하기 쉬움.

Transformer Application Preview

- Feedback on an online order

트랜스포머 모델이 온라인 주문과 같은 실제 시나리오에서 어떻게 응용될 수 있는지에 대한 예시

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

text는 아마존에 잘못된 주문 처리에 대해 제기된 브이. 교환 요청.

Transformer Application: Text Classification

※전체 내용 : 텍스트 분류

- Feedback on an online order 터깅레이스 트랜스포머스에서 모델 예측을 위한 가장 간단한 방법은 파이프라인 사용하는 것.

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

- Pipeline: an abstraction for making predictions from fine-tuned models

```
1 from transformers import pipeline
2
3 classifier = pipeline("text-classification")
```

단순히 pipeline("text-classification")이라고 코딩하여

classifier라는 인스턴스 생성하였습니다.

이 인스턴스는 그 자체로 텍스트를 입력으로 받아 예측 수행 가능.

2024. 9. 4.

AI융합학부

```
1 import pandas as pd
2
3 outputs = classifier(text)
4 pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

→ 고객의 피드백이 부정으로 예측되었으며, 확률은 0.90

Transformer Application: NER

두번째 응용: 개체명 인식 (NER) → 이는 문장의 각 단어 또는 토큰을 미리 지정된 레이블中 하나로 분류

• Named Entity Recognition (NER)

```
1 ner_tagger = pipeline("ner", aggregation_strategy="simple")
2 outputs = ner_tagger(text)
3 pd.DataFrame(outputs)
```

	entity_group	score	word	start	end
0	ORG	0.879011	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556571	Mega	208	212
4	PER	0.590256	##tron	212	216
5	ORG	0.669692	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

Score은 엔티티 레이블의 확률이고, start와 end는 문장내 오프셋

파이프라인 생성 시 태스크는 'ner'로 지정.

추가로 aggregation_strategy는 simple로 지정.

↳ 연속된 단어가 동일 레이블일 경우, 그 전부를 해당 레이블로 카운팅하도록 함.

ex) optimus prime은 MISC로 분류된 연속된 단어로, 하나의 개체로 생각

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

Transformer Application: QA

: 질문 & 답변.

- Question and Answering (QA)

```
1 reader = pipeline("question-answering")
2 question = "What does the customer want?"
3 outputs = reader(question=question, context=text)
4 pd.DataFrame([outputs])
```

	score	start	end	answer
0	0.631292	335	358	an exchange of Megatron

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

Transformer Application: Summarization

- Summarization

```
1 summarizer = pipeline("summarization")
2 outputs = summarizer(text, max_length=60, clean_up_tokenization_spaces=True)
3 print(outputs[0]['summary_text'])
```

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

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

Transformer Application: Translation

- translation

```
1 translator = pipeline("translation_en_to_de",
2                         model="Helsinki-NLP/opus-mt-en-de")
3 outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)
4 print(outputs[0]['translation_text'])
```

Sehr geehrter Amazon, letzte Woche habe ich eine Optimus Prime Action Figur aus Ihrem Online-Shop in Deutschland bestellt. Leider, als ich das Paket öffnete, entdeckte ich zu meinem Entsetzen, dass ich stattdessen eine Action Figur von Megatron geschickt worden war! Als lebenslanger Feind der Decepticons, Ich hoffe, Sie können mein Dilemma verstehen. Um das Problem zu lösen, Ich fordere einen Austausch von Megatron für die Optimus Prime Figur habe ich bestellt. Anbei sind Kopien meiner Aufzeichnungen über diesen Kauf. Ich erwarte, bald von Ihnen zu hören. Aufrichtig, Bumblebee.

a model identifier

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

Transformer Application: Text Generation

- generation

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

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

Hugging Face Ecosystem

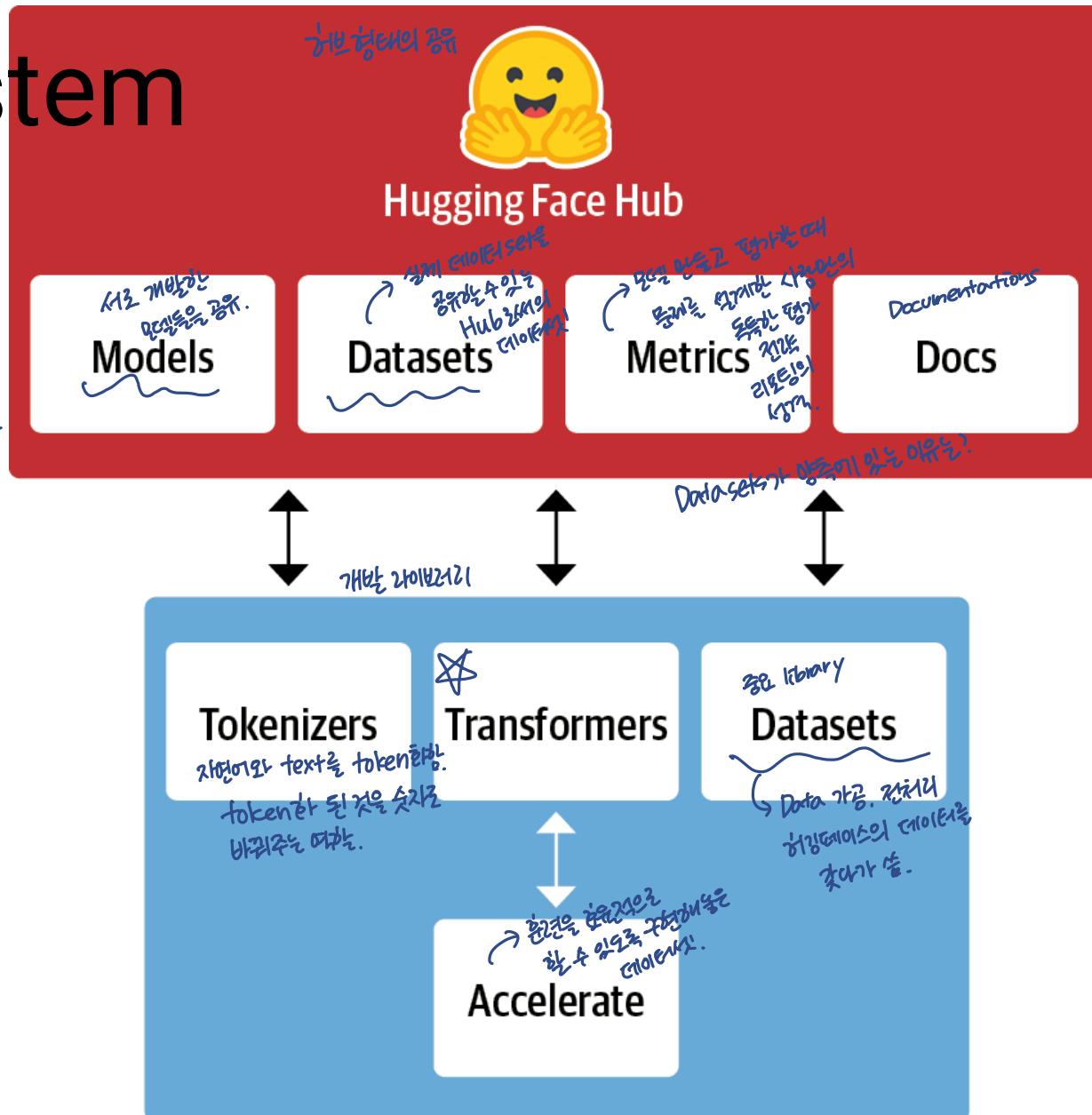
→ AI 개발자라면 보통 허깅페이스로 개발.

- Library:

- Provides code:
- Tokenizer: Sequence preprocessing.
- Transformers: Transformer implementation.
 - 허깅페이스에서 가장 학기적이고 중요한 라이브러리
- Datasets: Standardized data loading/transformation.
- Accelerate: High-performance training (multi-GPU, multi-node).

- Hub:

- Pre-trained model weights.
- Datasets.
- Scripts for evaluation metrics.



Hugging Face Hub

The screenshot shows the Hugging Face Hub interface. At the top, there is a navigation bar with a search bar, icons for Models, Datasets, Spaces, Posts, Docs, Pricing, and a user profile. Below the navigation bar, there are tabs for Tasks, Libraries, Datasets, Languages (which is selected), and Licenses. There is also an "Other" link. A sidebar on the left lists various languages with their respective icons. The main content area displays a list of trending models:

- black-forest-labs/FLUX.1-dev**
Text-to-Image • Updated 18 days ago • ↓ 688k • ⚡ • ❤ 3.66k
- THUDM/CogVideoX-5b**
Text-to-Video • Updated 5 days ago • ↓ 14.3k • ❤ 301
- Qwen/Qwen2-VL-7B-Instruct**
Image-Text-to-Text • Updated about 7 hours ago • ↓ 20.6k • ❤ 231
- black-forest-labs/FLUX.1-schnell**
Text-to-Image • Updated 18 days ago • ↓ 1.88M • ⚡ • ❤ 2.06k
- meta-llama/Meta-Llama-3.1-8B-Instruct**
Text Generation • Updated 14 days ago • ↓ 2.88M • ⚡ • ❤ 2.2k
- Shakker-Labs/AWPortrait-FL**
Text-to-Image • Updated about 6 hours ago • ↓ 1.57k • ❤ 124

Hugging Face Hub

- Hosts over 900K models
 - Quick access to candidate models
 - Simple model loading with pipelines
 - Dataset hosting
 - Hosting scripts for evaluation metrics
 - Model cards
 - Dataset cards

loss와 metric의 차이
↓
loss는 학습률을
metric은 update
loss는 미분 가능.

epoch 중간
업데이트 된 모델 평가
metric이 학습된지 유무
metric은 미분 불가능.
epoch로 metric 평가를 이용.

Hugging Face Tokenizer

- Covered in detail in Chapter 2.
 - Sentence Tokenization:
 - By word
 - By character
 - Intermediate forms between words and characters
- Sentence를 최소 단위로 잘라냄.
예전엔 word 단위(공백 기준)으로 잘랐는데, 품격이 좋았음.
오타나, 여러 번 중복되는 단어가 있을 때에는 단점이 있었음.(토큰의 개수수가 너무 많아짐).
- 개별 글자로 표현하면 너무 세분화되어서 품격이 좋지 않음.
이제는 word와 character 중간의 성격으로 토큰화이 가능함.

Hugging Face Datasets

- Significant time-consuming area in deep learning applications.
- Data loading, processing, and storage.
- Various evaluation scripts.
- Handles data processing exceeding memory limits, optimized I/O, and other engineering challenges.

→ 내가 대용량 데이터셋을
직접 다루면, 디플린스가
종종 인증을 더 어렵다.
제공됨.

Hugging Face Accelerate

- Speeds up training/inference workflows
- Supports:
 - Multi-GPU
 - Multi-node
 - TPU → 구글 클라우드에서 자체적으로 운영하는 가속장치
 - Simplifies utilization of these resources

Key Challenges in Transformers

- Language:
• Limited data availability.
• Multilingual transformers, zero-shot cross-lingual transfer.

어떤 언어에 대해서는
트랜스포머가 잘 학습이 안되어 있음.
 - Data Availability:
• Scarcity of labeled data.
 - Handling Long Documents:
• Constraints due to sequence length limitations.
 - Opacity:
• Lack of explainability.
 - Bias:
• Issues related to racial and gender bias.
- AI는 세계의 언어들은
트랜스포머가 학습하는데
다른 언어의 경우
가장 먼저 사용 가능 .
- sequence 길이의 제한?
최근의 GPT는
128000단위의 토큰을 다룰 수 있음
- 구글 트랜스포머는
100,000단위의
토큰 .

Practical Resources

- Online GitHub Repository : <https://github.com/nlp-with-transformers/notebooks>
- Google Colab Instructions : File -> Open Notebook -> GitHub

노트 열기

예 >	GitHub URL을 입력하거나 조직 또는 사용자로 검색하세요.		
	<input type="text" value="nlp-with-transformers/notebooks/blob/main/01_introduction.ipynb"/> <input type="button" value=""/>		
최근 사용 >	저장소: <input type="button" value=""/>	브랜치: <input type="button" value=""/>	<input checked="" type="checkbox"/> 비공개 저장소 포함
Google Drive >	nlp-with-transformers/notebooks main		
GitHub >	경로		
업로드 >	 01_introduction.ipynb <input data-bbox="1710 923 1761 951" type="button"/> <input data-bbox="1786 923 1838 951" type="button"/>		
	 02_classification.ipynb <input data-bbox="1710 1031 1761 1059" type="button"/> <input data-bbox="1786 1031 1838 1059" type="button"/>		
	 03_transformer-anatomy.ipynb <input data-bbox="1710 1139 1761 1167" type="button"/> <input data-bbox="1786 1139 1838 1167" type="button"/>		
	 04_multilingual-ner.ipynb <input data-bbox="1710 1247 1761 1275" type="button"/> <input data-bbox="1786 1247 1838 1275" type="button"/>		
	 05_text-generation.ipynb <input data-bbox="1710 1340 1761 1369" type="button"/> <input data-bbox="1786 1340 1838 1369" type="button"/>		

Next Week

- Text Classification (Chapter 2 in Textbook)
- Using pre-trained model BODY
- Attaching classification HEAD
- Learning how to fine-tune to minimize classification loss