10장 자연어 처리를 위한 임베딩

생성일	@2024년 12월 30일 오후 8:42
주차	

10.1 임베딩

- 임베딩은 자연어를 컴퓨터가 이해할 수 있는 숫자 형태인 vector로 변환한 결과 혹은 그 과정을 의미
- 임베딩의 역할은 1) 단어 및 문장 간 관련성 계산 2) 의미적/문법적 정보의 함축

10.1.1 희소 표현 기반 임베딩

- 대부분의 값이 0으로 채워져 있는게 희소 표현 ex) 원핫인코딩

그림 10-1 원-핫 인코딩

- 이때 단점은, one-hot vector는 sparse vector이기 때문에 두 단어의 벡터의 내적을 구해면 0을 가지므로 직교를 이룸. 따라서 단어끼리의 관계성 없이 서로 독립적인 관계가 됨
- 차원의 저주 - 하나의 단어를 표현하는 corpus 말뭉치에 있는 수 만큼 차원이 존재하기 때문에 복잡해짐

10.1.2 횟수 기반 임베딩

- counter vector는 문서 집합에서 단어를 토큰으로 생성하고 각 단어의 출현 빈도수를 이용하여 인코딩해서 벡터를 만드는 방법
- 횟수 기반은 따라서 단어가 출현한 빈도를 고려해서 임베딩하는 방법임
- TF-IDF는 정보 검색론에서 가중치를 구할 때 사용되는 알고리즘음
- TF(Term Frequency)는 문서 내 특정 단어가 출현한 빈도를 의미함

- ex) TF 에 딥러닝과 신문이라는 단어가 포함되어 있다면 → 신문에서 딥러닝이라는 단어가 몇번 등장했는지를 의미함

$$tf_{t,d} = \begin{cases} 1 + \log count(t,d) & count(t,d) > 0 \text{일 때} \\ 0 & \text{그 외} \end{cases}$$

이때 tf는 특정 문서 d에서 특정 단어 t의 등장 횟수를 의미함

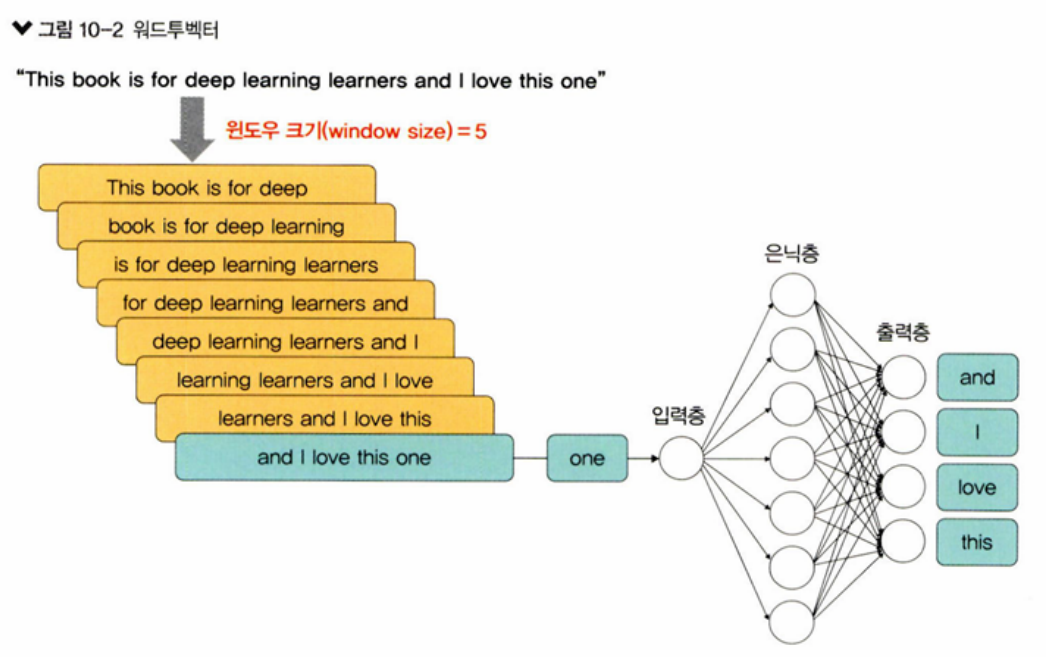
- DF(Documnet Frequency)란 한 단어가 전체 문서에서 얼마나 공통적으로 많이 등장하는지를 나타내는 값임 → df(t): 특정 단어 t가 포함된 문서 갯수
- 이때 특성 단어 t가 모든 문서에 등장하는 the와 같은 일반적인 단어라면, 즉 DF값이 크면 DF에 역수를 취해 가중치를 낮춰야 하는데 그 값이 바로 IDF(Inverse Document Frequency)

$$idf_t = \log(\frac{N}{1+df_t}) = \log(\frac{\text{전체 문서 개수}}{1+\text{특정 단어 } t\text{가 포함된 문서 개수}})$$

- TF-IDF가 사용되는 예시 → 키워드 검색을 기반으로 하는 검색 엔진, 중요 키워드 분석, 검색 엔진에서 검색 결과의 순위를 결정
- 따라서 특성 문서 내 출현 빈도가 높거나 전체 문서에서 특정 단어가 포함된 문서가 적을수록 TF-IDF 값이 높음

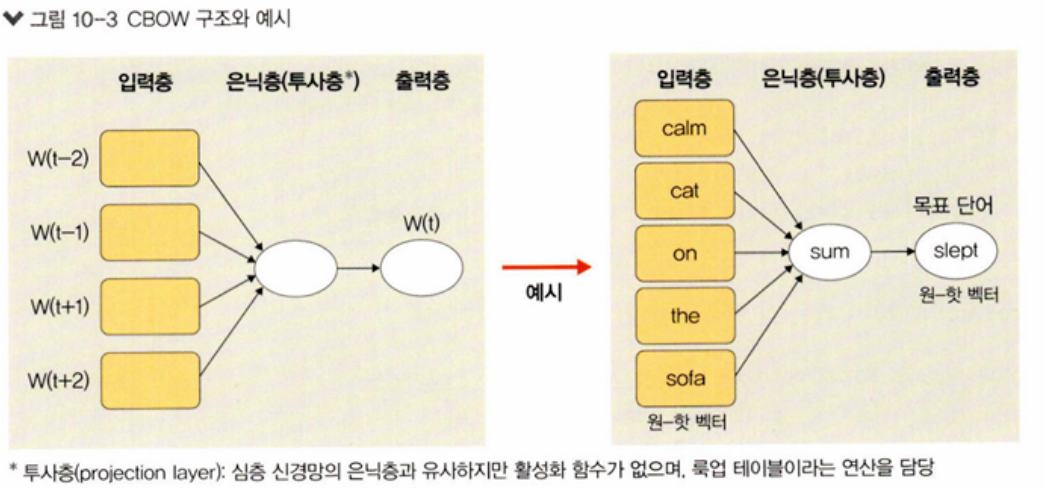
10.1.3 예측 기반 임베딩

- 신경망 구조 또는 모델링을 사용해서 특정 문맥에서 어떤 단어가 나올지 예측하면서 단어를 벡터로 만드는 방식
- Word2Vec은 신경망 알고리즘으로 주어진 텍스트에서 각 텍스트이 각 단어마다 하나식 일련의 벡터를 출력
- 이때 출력된 벡터가 2차원 그래프에 표시될 때 의미론적으로 유사한 단어의 벡터는 서로 가깝게 표현되고 코사인 유사도를 이용하여 각 단어간의 거리를 측정함 → 따라서 특정 단어의 동의어를 찾을 수 있음

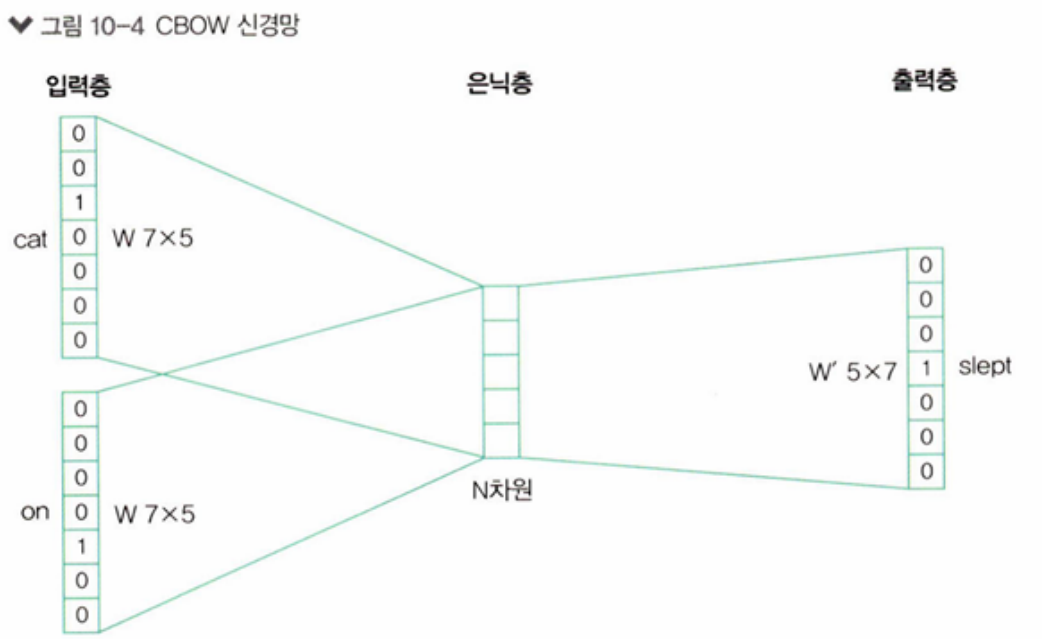


일정한 크기의 window로 분할된 텍스트를 신경망의 입력으로 사용

- 단어간 유사성을 확인하는 것으로 CBOW(Continuous Bag Of Words)는 단어를 여러개 나열한 후 이와 관련된 단어를 추정하는 방식으로 문장에서 n개의 단어 열에서 다음에 등장할 단어를 예측하는 방식임



- CBOW 신경망 크기가 N인 은닉층일 때 이때 N은 입력 텍스트를 임베딩한 벡터 크기임

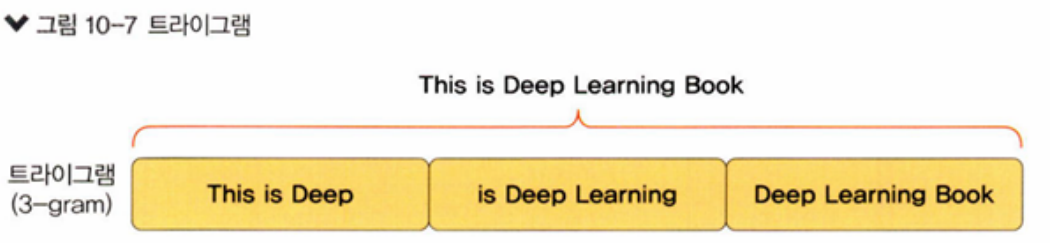


- skip-gram은 CBOW와 반대로 특정한 단어에서 문맥이 될 수 있는 단어를 예측함 → 중심 단어를 중심으로 주변 단어를 예측하는 방식임
- FastText는 Word2Vec의 단점을 보완하고자 페이스북에서 개발한 알고리즘으로 Word2Vec의 워드 임베딩은 분산 표현을 이용하여 단어의 분산 분포가 유사한 단어들에 비슷한 벡터 값을 할당하여 표현함 → 따라서 Word2Vec은 사전에 없는 단어들에 대해서는 벡터 값을 얻을 수 없음 + 자주 사용되지 않는 단어에 대해서는 학습이 불안정함
- FastText는 이러한 단점을 보완하고자 개발된 단어 표현 방식 → 노이즈에 강하고 새로운 단어에 대해서는 형태적 유사성을 고려한 벡터 값을 얻음

[FastText가 Word2Vec의 단점을 보완하는 방식]

1. 사전에 없는 단어에 대해서는 벡터 값을 부여함

: 주어딘 문서의 각 단어를 n-gram으로 표현하는데, n의 설정에 따라 단어의 분리 수준이 결정됨.



n=3일때의 예시, 3개의 단어씩 묶음

: FastText는 신경망을 통한 학습이 완료된 후 데이터셋의 모든 단어를 n-gram에 대해 임베딩함 → 만약 사전에 없는 단어가 있다면 n-gram으로 분리된 부분 단어와 유사도를 계산하여 의미를 유추

2. 자주 사용되지 않는 단어에 학습 안정성을 확보

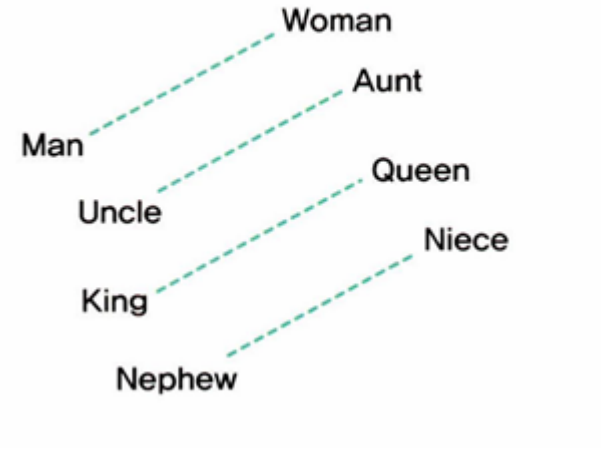
: Word2Vec의 단점은 단어의 출현 빈도가 적으면 임베딩의 정확도가 낮다는 것

: FastText는 등장 빈도수가 적더라고, n-gram으로 임베딩하기 때문에 참고할 수 있는 경우의 수가 많아짐

10.1.4 횟수/예측 기반 임베딩

- GloVe(Global Vectors for Word Representation)은 횟수 기반의 LSA(Latent Semantic Analysis, 잠재 의미 분석)과 예측 기반의 Word2Vec의 단점을 보완하기 위한 모델임
- 단어에 대한 global 동시 발생 확률 정보를 포함하는 단어 임베딩 방식으로 단어에 대한 통계정보와 skip-gram을 합친 방식임

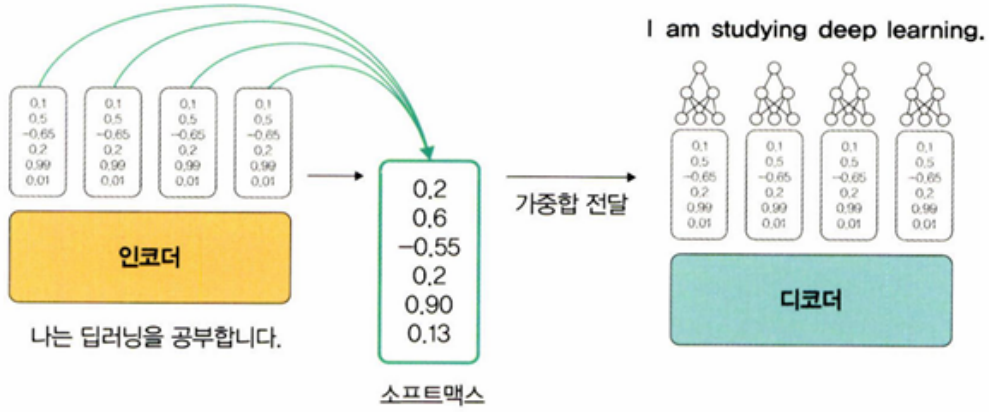
그림 10-8 글로브를 이용한 단어 간 관련성 예시



10.2 Transformer Attention

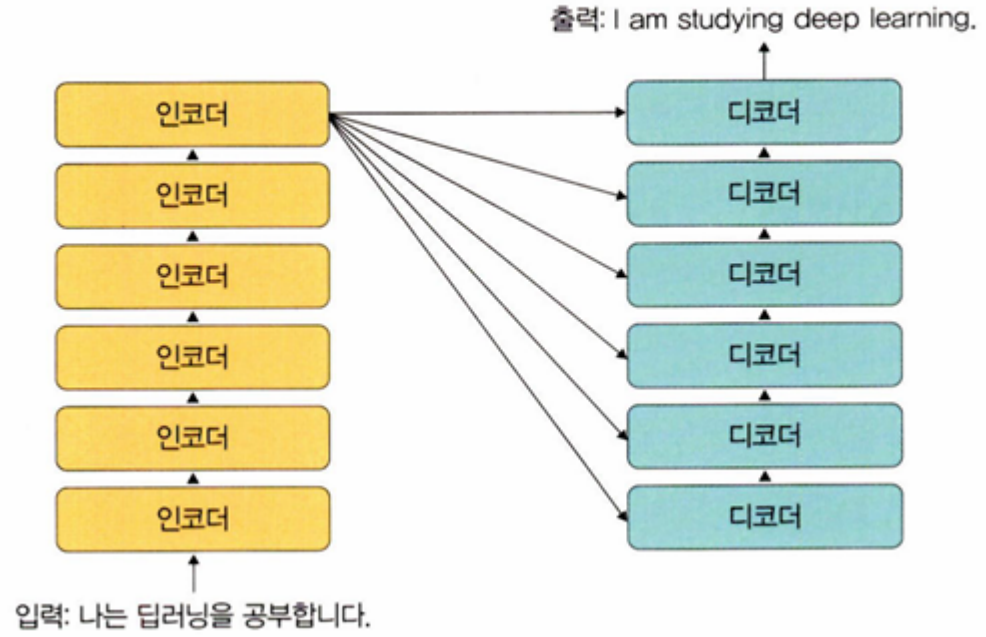
- attention은 주로 언어 번역에서 사용되는데 인코더와 디코더 네트워크를 사용함: 입력에 대한 벡터 변환을 encoder에서 처리하고 모든 벡터를 decoder로 내보냄 → 이때 모든 벡터가 전달 되기 때문에 행렬이 커지는 단점을 보완하기 위해 softmax함수를 사용하여 가중합을 구하고 그 값을 decoder에 전달 → decoder는 hidden state 일때 attention해서 보아야할 벡터를 softmax 함수로 점수를 매긴 후 각각 hidden state의 vector와 곱합 → 모두 더해서 하나의 값을 만듦

❮ 그림 10-9 어텐션



- Transformer은 encoder과 decoder를 여러 개 중첩시킨 구조로 각각의 encoder과 decoder를 묶어 block이라고 함

❮ 그림 10-10 어텐션에서 인코더와 디코더



- 이때 하나의 encoder는 self-attention과 전방향 신경망으로 구성되어있는데, encoder는 단어를 벡터로 임베딩하여 이를 self-attention과 전방향 신경망으로 전달함 → 이때 self-attention은 문장에서 각 단어끼리 얼마나 관계가 있는지를 계산해서 반영함 → 이 단어 간 관계가 전방향 신경망으로 전달됨
- decoder는 3개의 층으로 구성되어 있음 1) self-attention 층은 encoder로 넘어온 벡터가 처음 만나는 것으로 encoder과 동일 2) encoder-decoder attention 층은

encoder가 처리한 정보를 받아 attention 매커니즘으로 수행하고 마지막으로 전방향 신경망으로 데이터가 전달됨

attention 매커니즘이란?

$$e_{ij} = a(s_{i-1}, h_j)$$

attention score를 구하는 식

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

attention score에 softmax를 취함

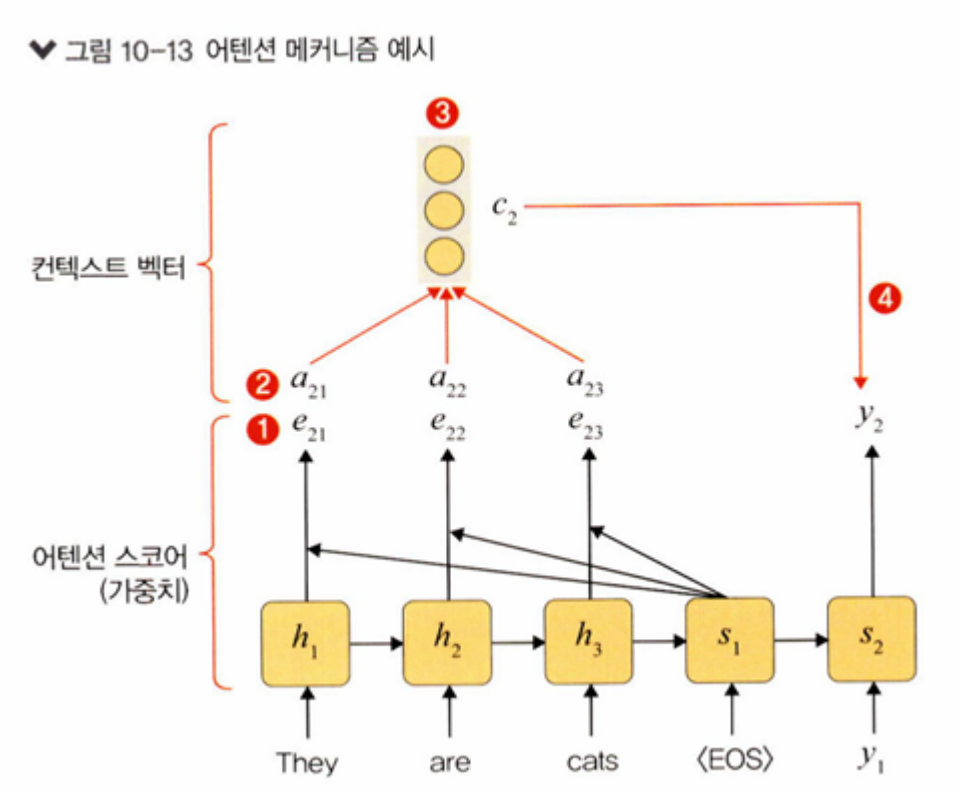
$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$

가중합을 계산하여 하나의 벡터 완성 → context vector

: attention score란 현재 decoder의 시점 i에서 단어를 예측하기 위해 encoder의 모든 은닉 상태의 값 hj이 현재 decoder의 은닉상태 sij와 얼마나 관련이 있는지를 판단하는 값

: 이를 softmax를 통해 확률로 변환하면 특정 시점에 대한 가중치가 계산됨

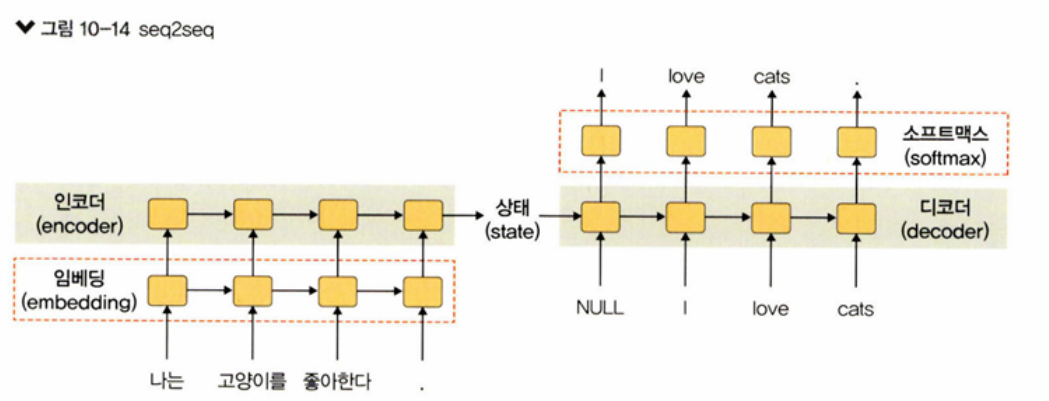
: context vector와 이전 시점의 hidden sate과 출력 → decoder의 hidden state 계산



10.2.1 seq2seq

- input sequence와 output sequence를 만들기 위한 모델로 품사 판별보다 번역에 초점을 둔 모델

- 따라서 입력 시퀀스와 의미가 동일한 출력 시퀀스를 만드는 것이 목표



- 이때 seq2seq는 입력 문장이 긴 시퀀스일 경우 정확한 처리가 어려움 (RNN의 마지막 hidden layer만 decoder로 전달되기 때문)

→ attention 매커니즘으로 입력 시퀀스의 모든 숨겨진 상태를 유지+활용 → 정보의 손실과 기울기 소멸 문제가 발생하지 않음

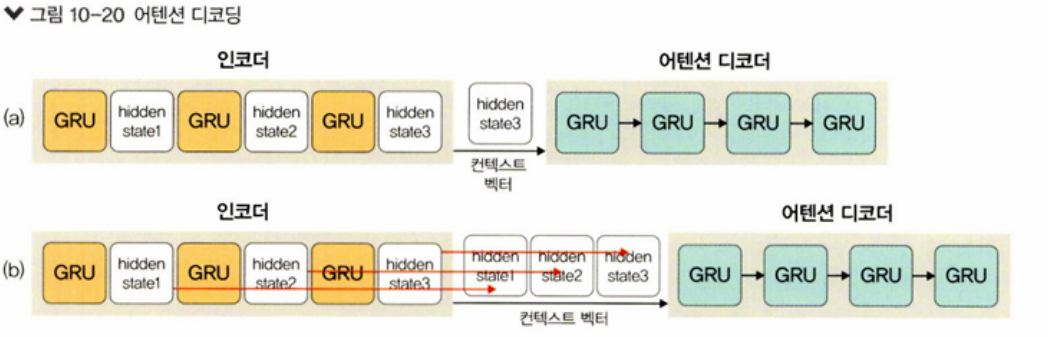
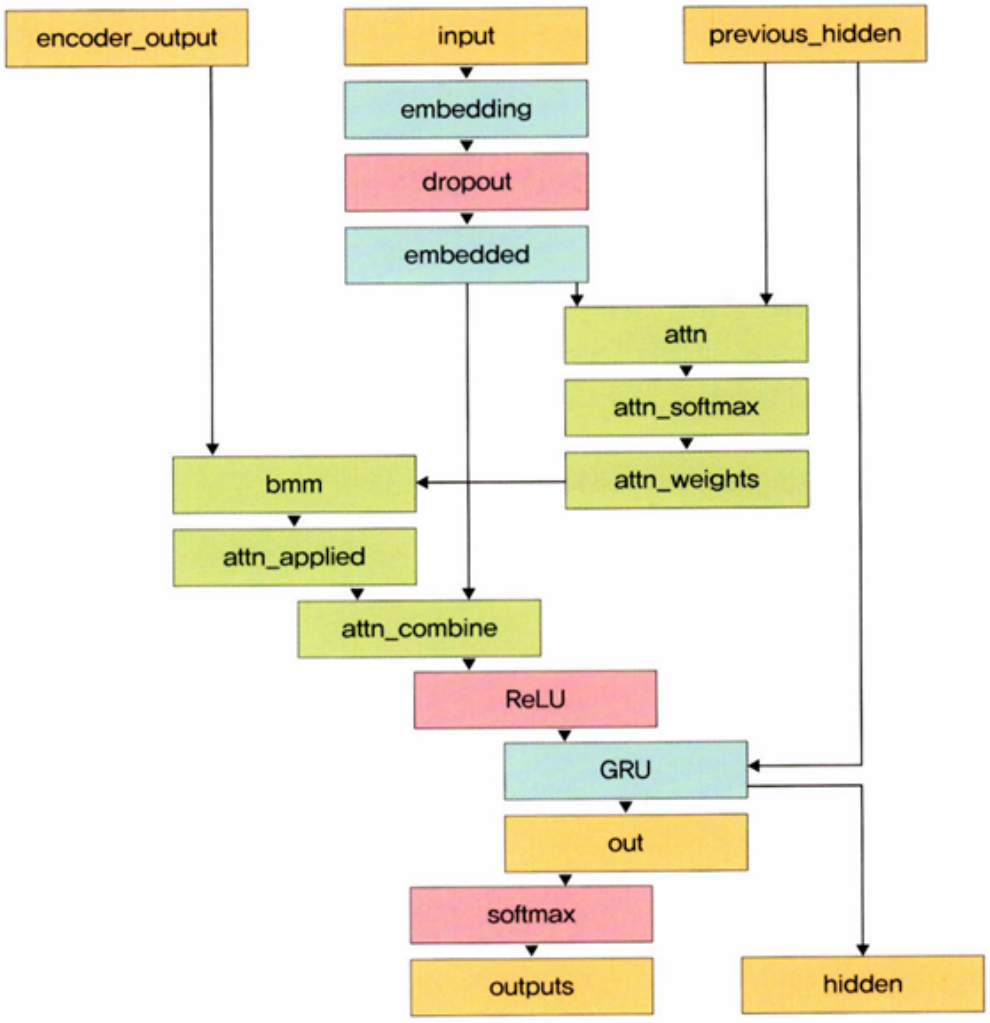
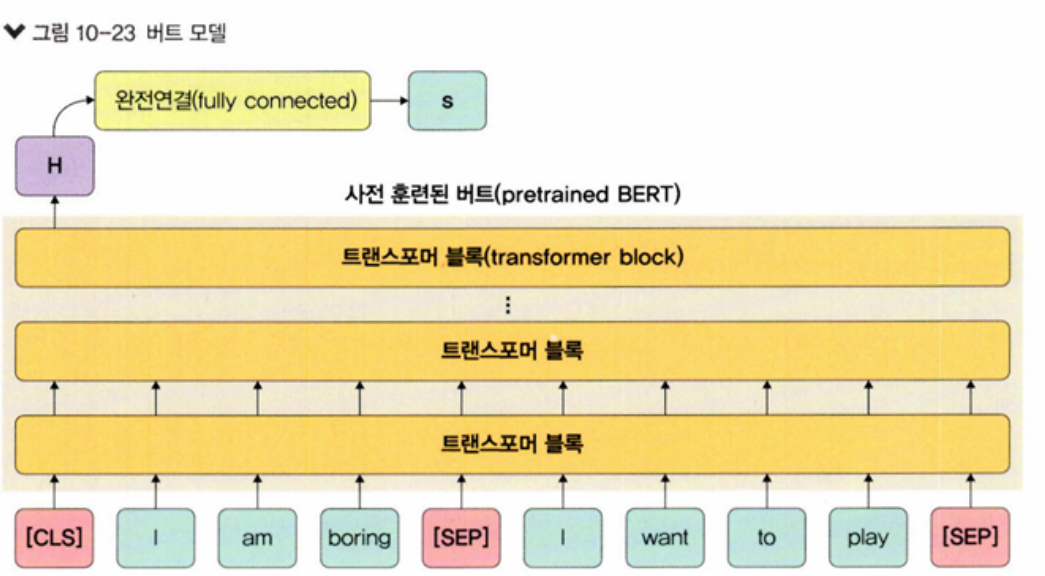


그림 10-21 어텐션이 적용된 디코더



10.2.2 BERT

- 기존 단방향 자연어 처리 모델들의 단점을 보완한 양방향 자연어 처리 모델로 검색 문장이 입력된 순서대로 하나씩 처리하는 것이 아닌 Transformer encoder를 쌓아올린 구조 → 문장 예측을 할 때 주로 사용됨



문장의 시작이 CLS, 끝이 SEP, 이때 한 문장의 각각 단어 ex) 고,양,이 에 대해 토큰화 진행