CH10

10.1 임베딩

임베딩 : 사람들이 사용하는 언어를 컴퓨터가 이해할 수 있는 벡터로 변환

- 단어 및 문장 관련성 계산
- 의미적/문법적 정보 함축

희소 표현 기반 임베딩

- 원핫 인코딩 → 텍스트를 숫자로 변환(포함되는 위치면 1, 아니면 0)

  단점

  - 단어에 대한 벡터 내적이 0으로, 직교하므로 단어의 관계성 고려 못함. 독립적 관계
  - 차원의 저주 → 말뭉치 개수만큼 차원이 늘어남

  ⇒ 대안 : 워드투벡터, 글로브, 패스트텍스트

횟수 기반 임베딩

: 단어 출현 빈도 고려

- 카운터 벡터
  - 문서 집합에서 단어 토큰으로 만들고, 출현 빈도수를 이용하여 인코딩 (토큰나이징 +벡터화)

    - sklearn CountVectorizer()
- TF-IDF
    - 정보 검색론에서 가중치 구할 때 사용
    - 키워드 검색 기반 검색 엔진/ 중요 키워드 분석/ 검색 결과 순위 결정 등
    - sklearn TfidfVectorizer()
    - TF → 단어 빈도. 문서 내 **특정 단어**가 출현한 빈도 (많이 등장하면 관련 높음)

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

    - IDF → 역문서 빈도
        - DF : 한 단어가 전체 문서에서 얼마나 공통적으로 많이 등장하는지. 특정 단어가 나타난 **문서 개수.**
        - 모든 문서에 나타나는 일반적 단어라면 가중치 낮춰야 함

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

      스무딩 버전

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

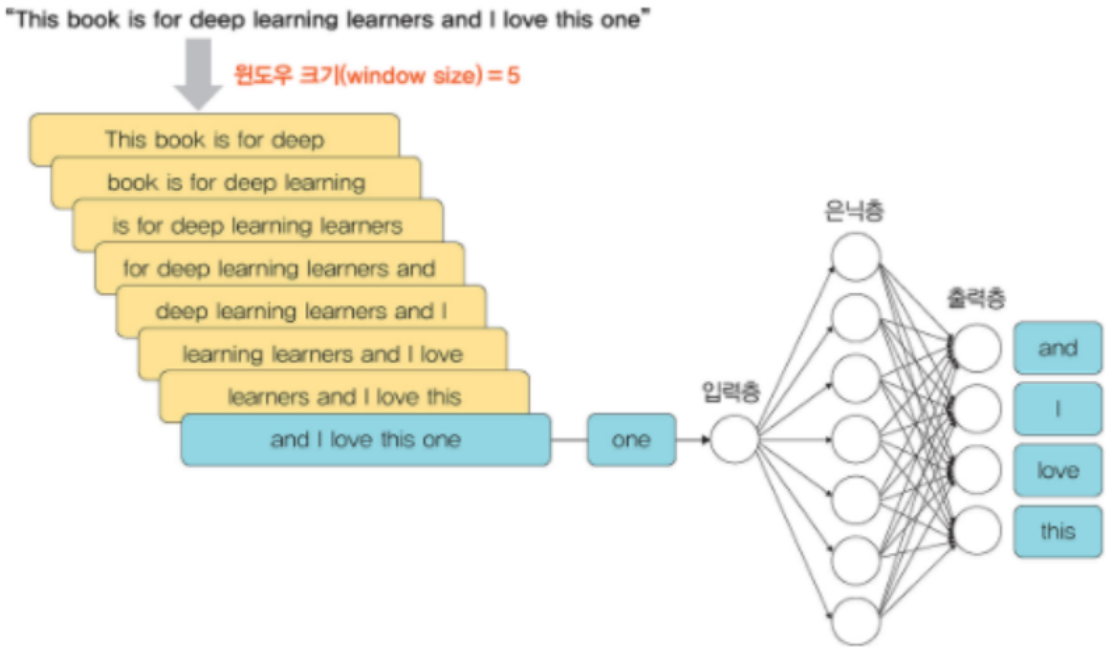
    ⇒ 특정 문서 내 단어 출현 빈도가 높거나 전체 문서에 특정 단어가 포함된 문서가 적을수록 TF-IDF 값 높음

예측 기반 임베딩

: 신경망 모델로 특정 문맥에서 어떤 단어가 나올지 예측하며 벡터화

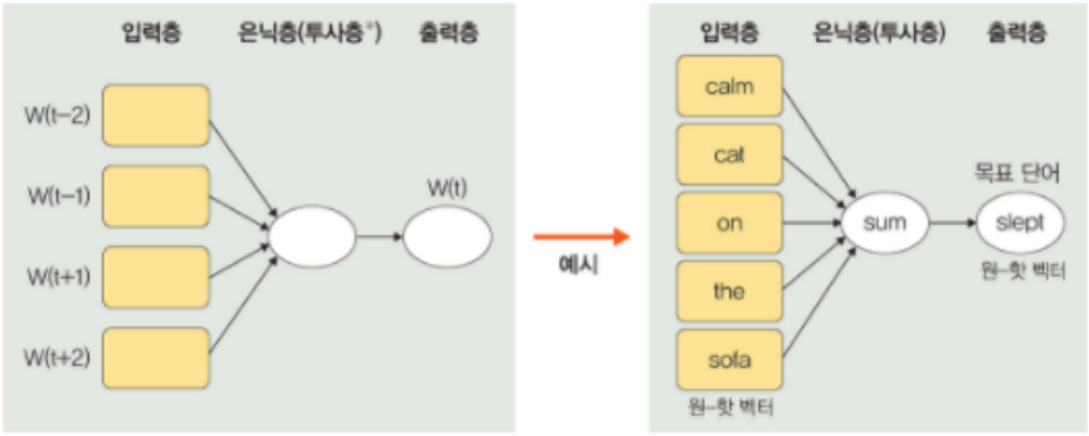
- 워드투벡터
    - 의미론적으로 유사한 단어 벡터끼리 가까움(코사인 유사도 등)
    - 동의어 찾기 가능

- 일정 크기 윈도우로 분할된 텍스트를 입력으로



단어 유사성 확인하기

1) CBOW : 문장에서 등장하는 n개의 단어 열에서 다음 단어 예측

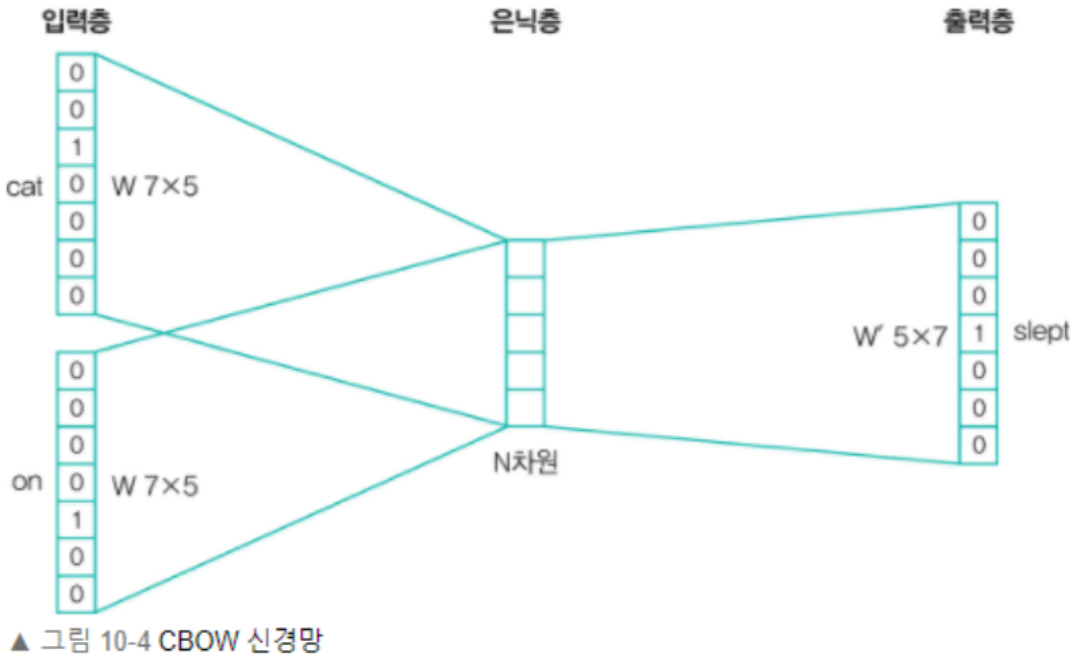


* 투사층(projection layer): 심층 신경망의 은닉층과 유사하지만 활성화 함수가 없으며, 룩업 테이블이라는 연산을 담당

▲ 그림 10-3 CBOW 구조와 예시

- 은닉층 크기 N = 입력 텍스트 임베딩 벡터 크기
- V = 단어 집합 크기
- 입력층~은닉층 가중치 차원 : V*N

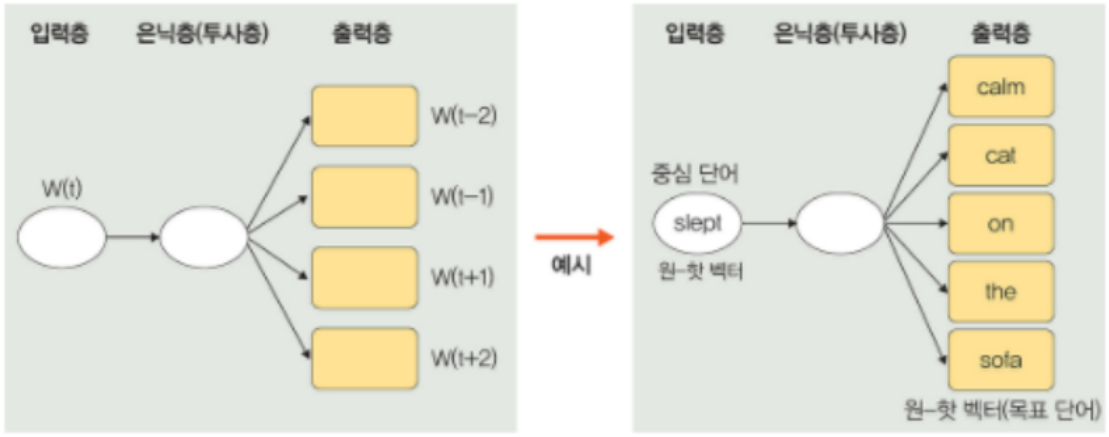
- 은닉층~출력층 가중치 차원 : N*V



▲ 그림 10-4 CBOW 신경망

2) skip-gram : 특정 단어에서 문맥이 될 수 있는 단어 예측

- 중심 단어에서 주변 단어 예측



▲ 그림 10-5 skip-gram

3) FastText : 워드투벡터 단점 보완하고자 페이스북에서 개발

- 워드투벡터(분산 표현을 이용하여 분산 분포가 유사한 단어에 비슷한 벡터값 할당하는 방식, 사전에 있어야만 하고 자주 사용하지 않는 단어에 불안정함)
- **단어 표현 방법 사용**

- 노이즈 강함. 새로운 단어에 대해 형태적 유사성을 고려해 사용

▼ 표 10-1 n 값에 따른 단어의 분리

문장	n 값	단어의 분리
This is Deep Learning Boo k	1	<This, is, Deep, Learning, Book>
	2	<This is, is Deep, Deep Learning, Learning Book>
	3	<This is Deep, is Deep Learning, Deep Learning Book>

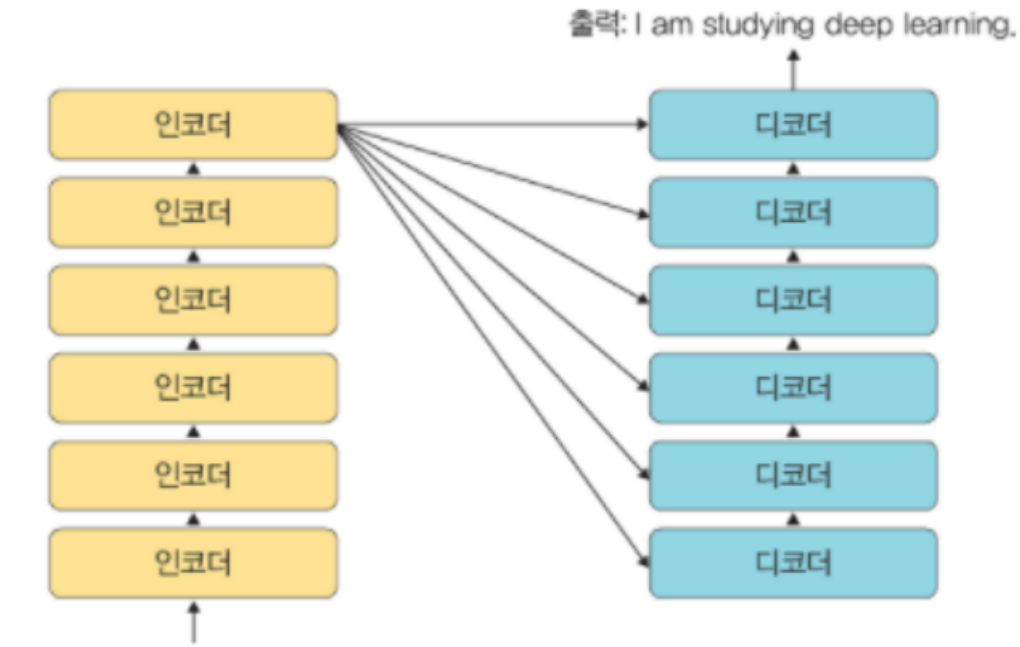
- 사전에 없는 단어는? 모든 단어를 각 n-그램에 대해 임베딩하고, 부분 단어와의 유사도를 구해 의미 유추
- 자주 사용되지 않는 단어에 학습 안정성 확보? n-그램으로 임베딩하기에 경우의 수가 많아 정확도가 높음

횟수/예측 기반 임베딩

- 글로브 : 횟수 기반 LSA 와 예측 기반 워드투벡터 단점 보완
  - 글로브 동시 발생 확률 정보 포함
  - skip-gram 방법 사용하되 통계적 기법 추가

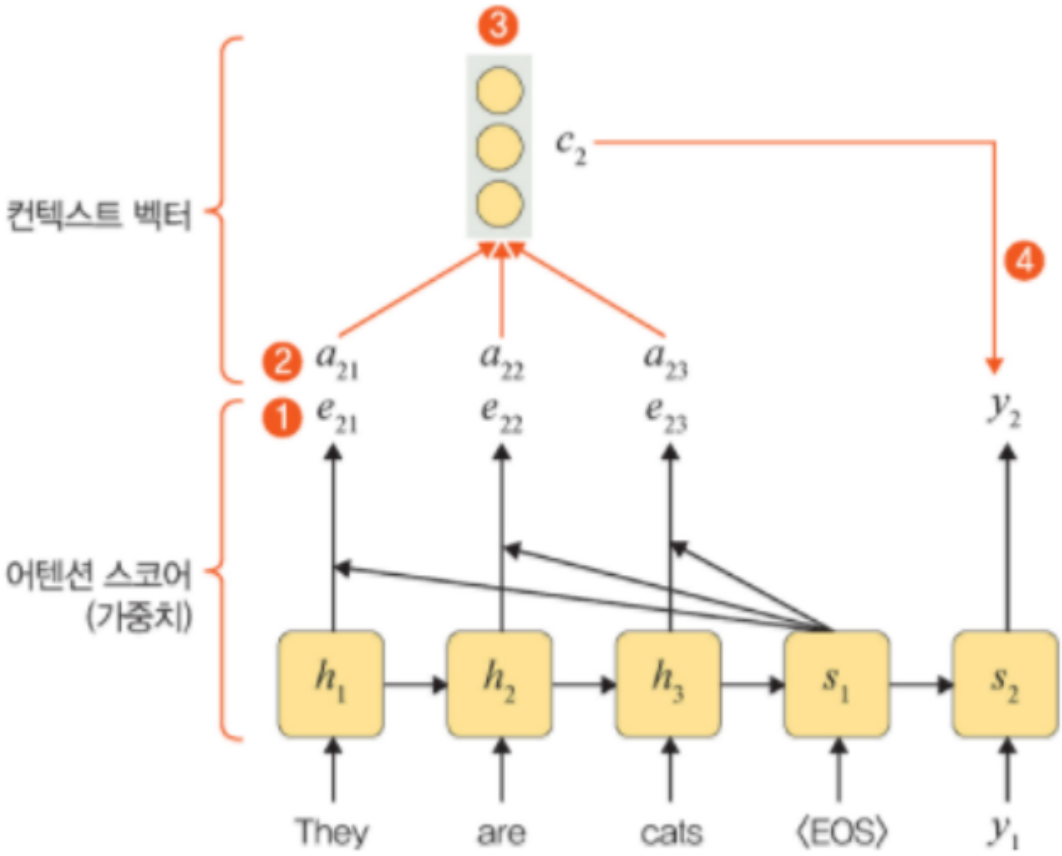
10.2 트랜스포머 어텐션

- **Attention** : 입력에 대한 벡터 변환을 인코더에서 처리하고 모든 벡터를 디코더로 보냄
  - 시간이 흐를수록 초기 정보를 잃어버리는 기울기 소멸 문제를 해결하기 위해 모두 보냄
  - 행렬 크기 커지는 단점 해결을 위해 소프트맥스로 가중합 구해 디코더로 전달
- Transformer : 인코더 디코더를 여러 개 중첩시킴
  - from Attention is All You Need



▲ 그림 10-10 어텐션에서 인코더와 디코더

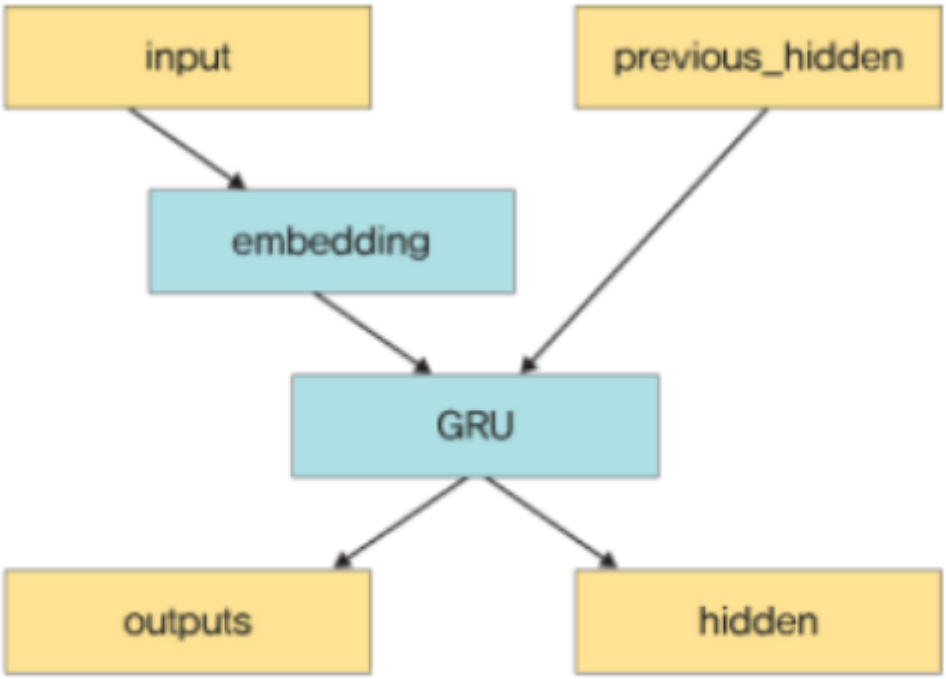
- 인코더 블록 : 셀프 어텐션(문장에서 각 단어끼리 얼마나 관계가 있는지 계산) + 전방향 신경망으로 구성
- 디코더 블록 : 셀프 어텐션 층 + 인코더-디코더 어텐션층 + 전방향 신경망
- **어텐션 스코어** : 현재 디코더의 시점 i에서 단어를 예측하기 위해, 인코더의 모든 은닉 상태 값(hj)이 디코더의 현 시점의 은닉 상태(si)와 얼마나 관련이 있는지(유사한지)를 판단하는 값 → 소프트맥스에 적용하여 확률 변환 = 시간의 가중치
- **컨텍스트 벡터** : 시간의 가중치와 은닉 상태의 가중합



▲ 그림 10-13 어텐션 메커니즘 예시

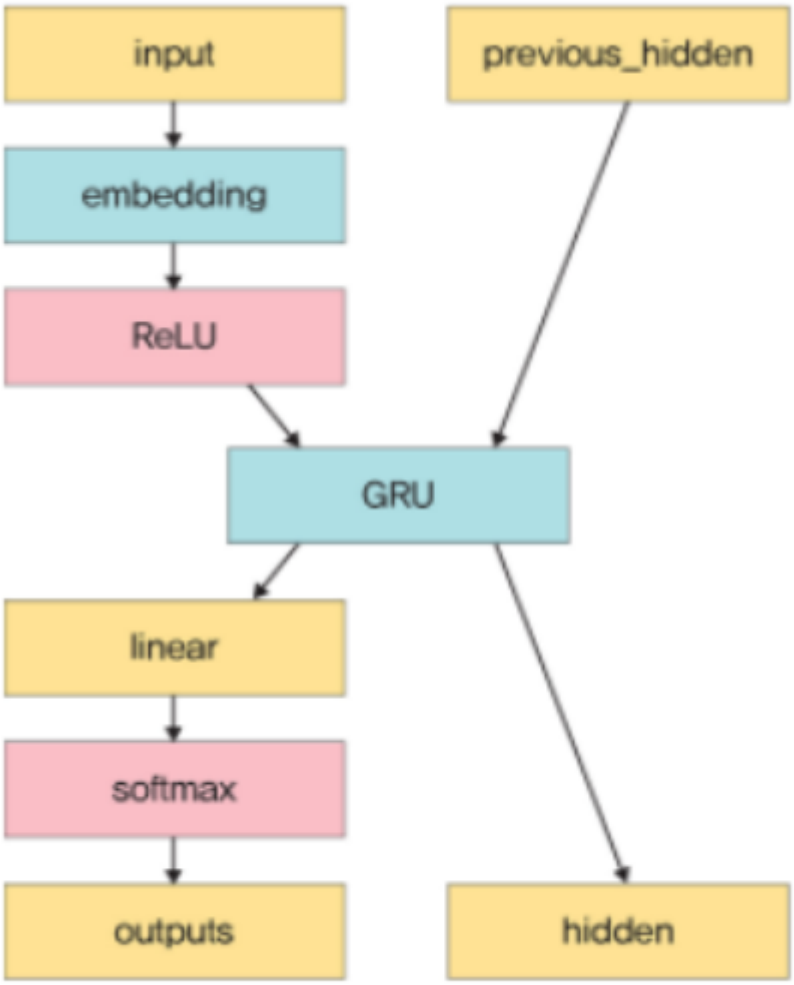
seq2seq : 입력 시퀀스에 대한 출력 시퀀스 만들기 위한 모델

- 번역에 초점. 입출력 간 관계는 안 중요. 길이 다를 수도.
    - 파이썬 기준



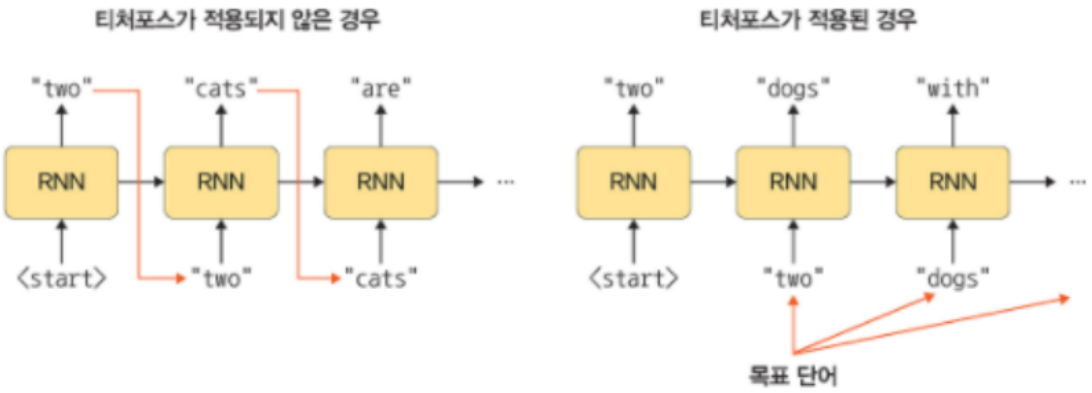
▲ 그림 10-16 임베딩 네트워크

- GRU 계층은 연속적인 입력 계산. 이전 계측 은닉 상태 계산 후 망각 게이트와 업데이트 게이트 갱신



▲ 그림 10-17 디코더 네트워크

- 티처포스 : 목표 단어를 디코더 다음 입력으로 넣어주는 기법
  - 초기 안정적 훈련/ 빠른 수렴 BUT 불안정 가능



▲ 그림 10-19 티처포스

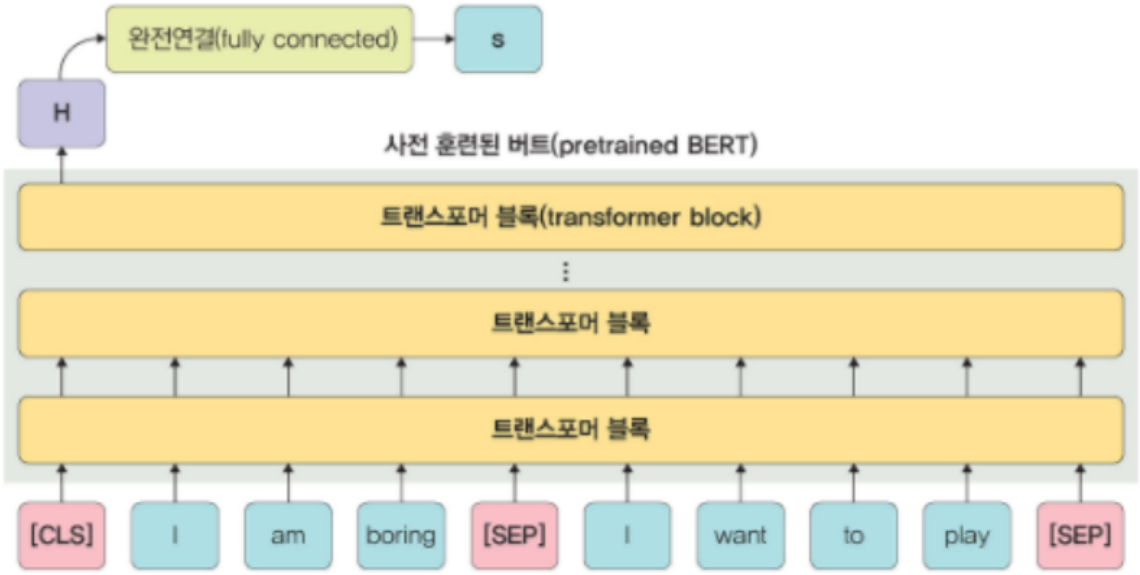
seq2seq with attention

- 기존 seq2seq 의 정보 손실/rnn 기울기 소멸 문제 해결을 위해
- 입력 시퀀스의 모든 숨겨진 상태를 활용함
- 디코더에서 특정 시점마다 다른 컨텍스트 벡터 사용

▲ 그림 10-21 어텐션이 적용된 디코더

버트 BERT

- 구글에서 공개. 양방향 자연어 처리 모델
- 트랜스포머로 구현. 방대한 텍스트 데이터로 사전 훈련된 모델
- 문장 예측에 사용



▲ 그림 10-23 버트 모델

- CNN/RNN X. 어텐션 개념 도입. 전이 학습에서 인코더만 사용
- BERT-base/BERT-large 두 버전 (클수록 블록, 은닉층, 어텐션 개수 큼)

10.3 한국어 임베딩

bert-base-multilingual-cased

KoBert