임베딩

임베딩 : 자연어를 숫자 형태인 벡터로 변환한 결과 혹은 일련의 과정

- 단어 및 문장 간 관련성 계산
- 의미적 혹은 문법적 정보의 함축

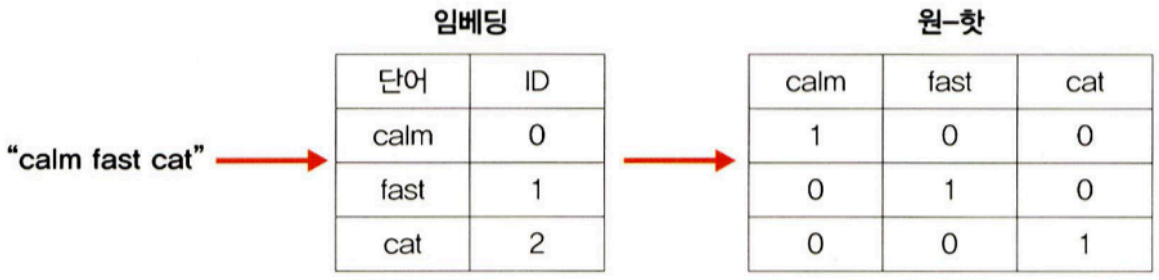
희소 표현 기반 임베딩

희소 표현 : 대부분의 값이 0으로 채워져 있는 경우

원-핫 인코딩

- 주어진 텍스트를 숫자로 변환해 주는 것
- 단어 N개를 N차원의 벡터로 표현하는 방식

❯ 그림 10-1 원-핫 인코딩



-
- 단점이 존재
  - 두 단어의 내적을 구하면 직교를 이룸 -> 단어끼리 관계성 없이 서로 독립적인 관계가 됨.
  - 차원의 저주가 발생 -> 하나의 단어를 표현하는 데 말뭉치에 있는 수만큼 차원이 존재하므로 복잡해짐

횟수 기반 임베딩

단어가 출현한 빈도를 고려하여 임베딩하는 방법

카운터 벡터

- 문서 집합에서 단어를 토큰으로 생성하고 각 단어의 출현 빈도수를 이용하여 인코딩
- 토크나이징과 벡터화가 동시에 가능

TF-IDF

- 정보 검색론에서 가중치를 구할 때 사용되는 알고리즘
- **TF** : 문서 내에서 특성 단어가 출현한 빈도

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

혹은

$$tf_{t,d} = \log(count(t,d) + 1)$$

(t(term): 단어, d(document): 문서 한 개)

- **IDF** : DF의 역수
  - **DF** : 한 단어가 전체 문서에서 얼마나 공통적으로 많이 등장하는지 나타내는 값

$$df_t = \text{특정 단어 } t\text{가 포함된 문서 개수}$$

- 스무딩 : 분모에 1을 더해줌으로써 특정 단어의 발생 빈도가 0일 때 분모가 0이 되는 상황을 방지
- 다음과 같은 상황에서 사용됨
  - 키워드 검색을 기반으로 하는 검색 엔진
  - 중요 키워드 분석
  - 검색 엔진에서 검색 결과의 순위를 결정
- 특정 문서 내에서 단어의 출현 빈도가 높거나 전체 문서에서 특정 단어가 포함된 문서가 적을수록 TF-IDF 값이 높음
  - -> 문서에 나타나는 흔한 단어들을 걸러 내거나 특정 단어에 대한 중요도를 찾을 수 있음.

예측 기반 임베딩

신경망 구조 혹은 모델을 이용하여 특정 문맥에서 어떤 단어가 나올지 예측하면서 단어를 벡터로 만드는 방식

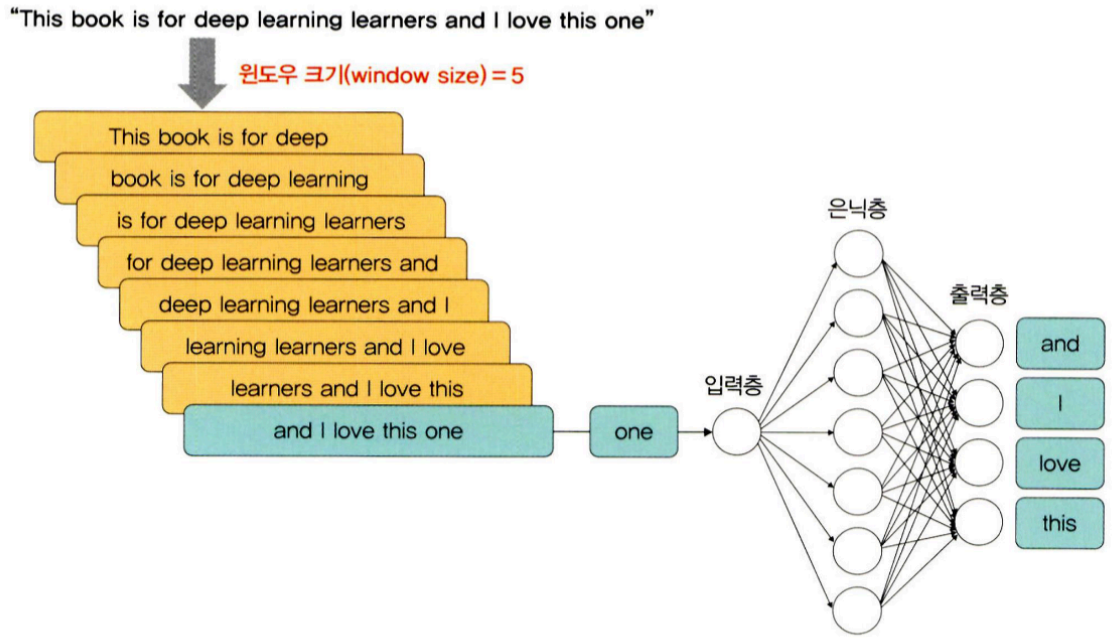
워드투벡터

- 신경망 알고리즘
- 주어진 텍스트에서 텍스트의 각 단어마다 하나씩 일련의 벡터를 출력
- 워드투벡터에서 의미론적으로 유사한 단어의 벡터는 서로 가깝게 (코사인 유사도로 측정) 표현됨

- -> 특정 단어의 동의어를 찾을 수 있음.

- 수행 과정
  1. 일정한 크기의 윈도우로 분할된 텍스트를 신경망 입력으로 사용
  2. 분할된 텍스트는 한 쌍의 대상 언어와 컨텍스트로 네트워크에 공급됨
  3. 네트워크의 은닉층에는 각 단어에 대한 가중치가 포함되어 있음.

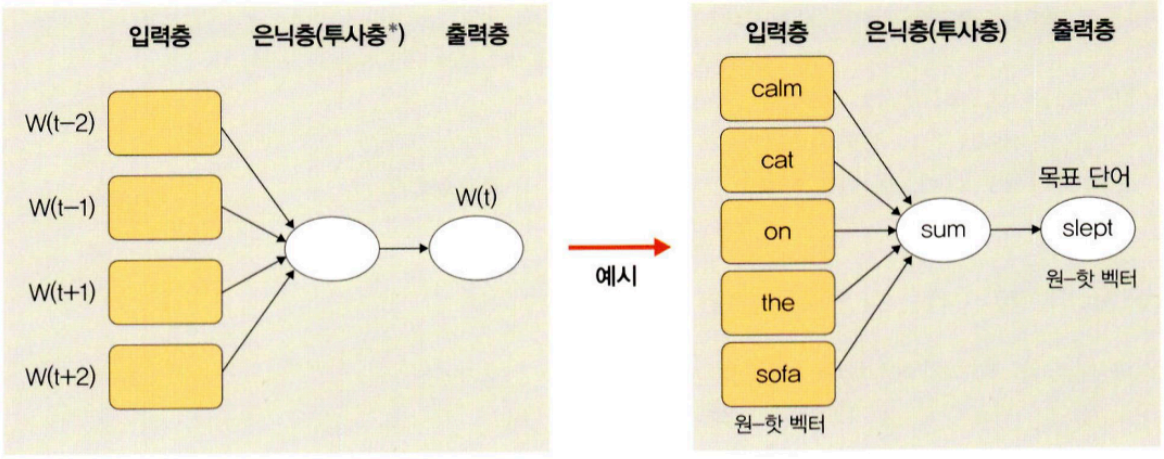
그림 10-2 워드투벡터



CBOW

- 단어를 여러 개 나열한 후 이와 관련된 단어를 추정하는 방식

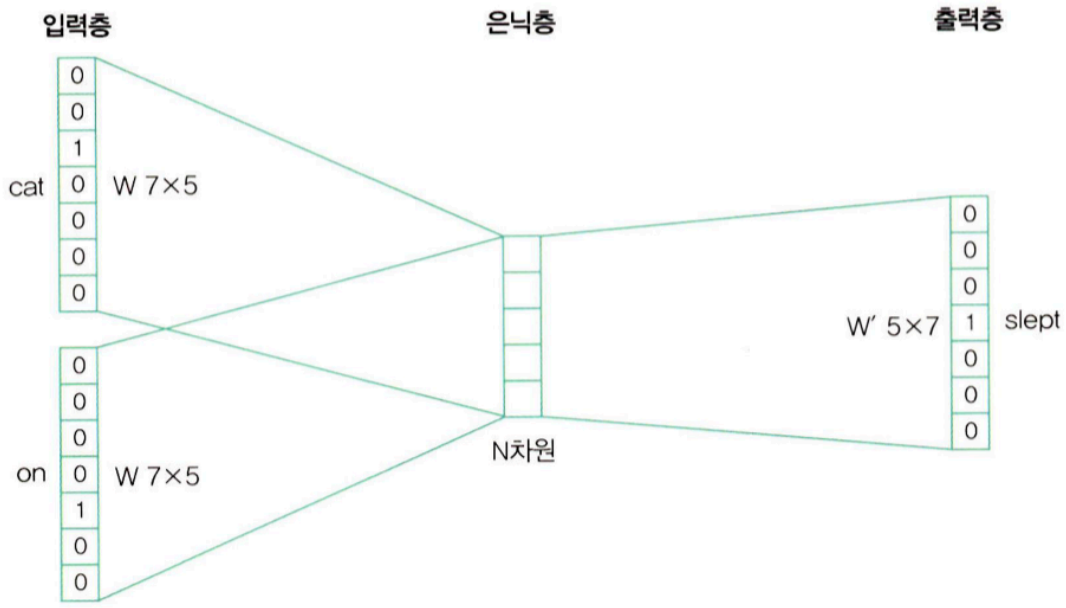
그림 10-3 CBOW 구조와 예시



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

- 다음과 같은 신경망 구조를 통해 구현됨.

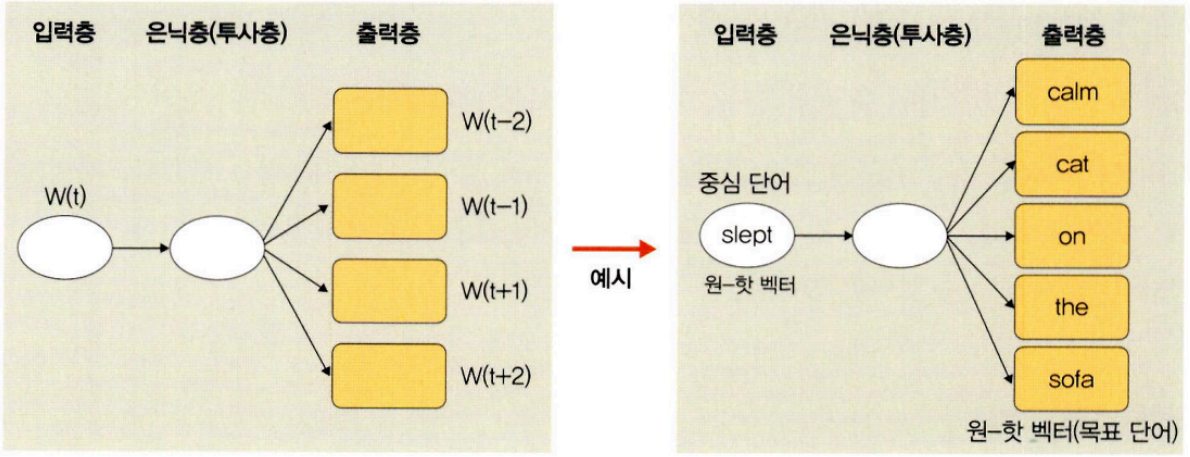
❯ 그림 10-4 CBOW 신경망



skip-gram

- 특정한 단어에서 문맥이 될 수 있는 단어를 예측
- 중심 단어에서 주변 단어를 예측하는 방식을 사용

❯ 그림 10-5 skip-gram



패스트텍스트

- 워드투벡터의 문제점
  - 사전에 없는 단어에 대해서는 벡터 값을 얻을 수 없음.
  - 자주 사용되지 않는 단어에 대해서는 학습이 불안정
- 이러한 단점들을 보완하려고 개발된 단어 표현 방법을 사용
- 사전에 없는 단어에 벡터 값을 부여
  - 데이터셋의 모든 단어로 n-그램에 대해 임베딩 -> 사전에 없는 단어가 등장한다면 부분 단어와 유사도를 계산해 의미를 유추

- 자주 사용되지 않는 단어에 학습 안정성을 확보
    - 등장 빈도수가 적더라도 n-그램으로 임베딩하기 때문에 참고할 수 있는 경우의 수가 많음

횟수/예측 기반 임베딩

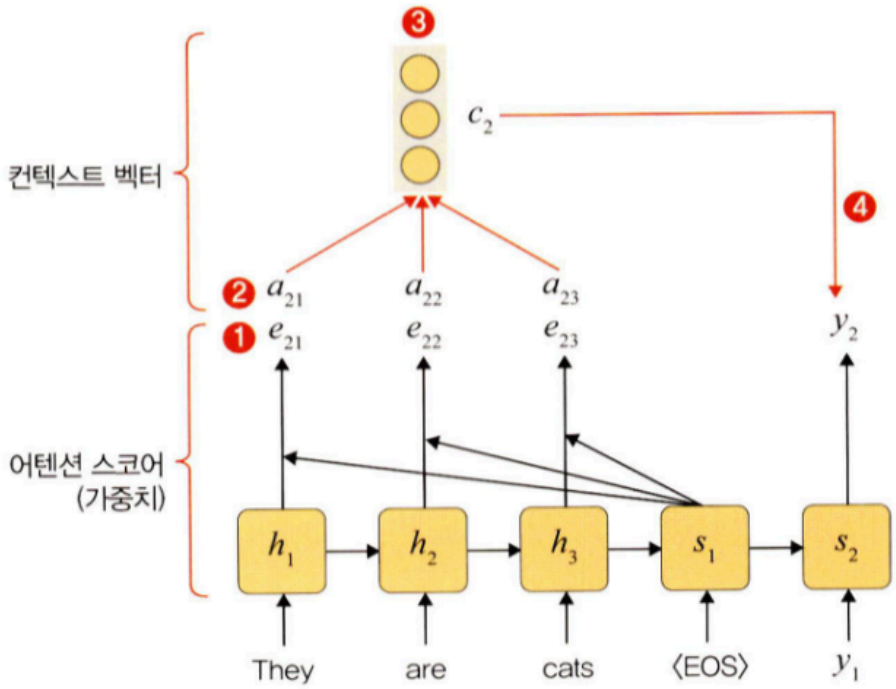
글로브

- 횟수 기반(LSA)과 예측 기반(워드투벡터)의 단점을 보완하기 위한 모델
- 단어에 대한 글로벌 동시 발생 확률 정보를 포함하는 단어 임베딩 방법

트랜스포머 어텐션

- 어텐션 : 인코더와 디코더 네트워크를 사용 -> 입력에 대한 벡터 변환을 인코더에서 처리하고 모든 벡터를 디코더로 보냄
    - 디코더의 부담을 줄이기 위해 디코더 : 은닉 상태에 대해 중점적으로 집중해서 보아야 할 벡터를 소프트맥스 함수로 점수를 매긴 후 각각을 은닉 상태의 벡터들과 곱함. -> 꼭 살펴보아야 할 벡터들에 집중
- 트랜스포머 : 어텐션을 극대화하는 방법
    - 인코더와 디코더를 여러 개 중첩시킨 구조
    - 인코더 : 단어를 벡터로 임베딩 후 이를 셀프 어텐션과 전방향 신경망으로 전달
    - 디코더
        - 셀프 어텐션 층 : 인코더에서 넘어온 벡터가 처음으로 만나는 층
        - 인코더-디코더 어텐션 층 : 인코더가 처리한 정보를 받아 어텐션 메커니즘을 수행
        - 전방향 신경망으로 데이터가 전달됨.
    - 수행 과정
        1. s1과 모든 인코더 은닉 상태에 대한 어텐션 스코어를 계산
        2. 소프트맥스 함수를 적용하여 시간의 가중치를 구함
        3. 시간의 가중치와 인코더의 은닉 상태 값들을 이용하여 가중합을 계산함으로써 컨텍스트 벡터를 구함
        4. 컨텍스트 벡터와 디코더 이전 시점의 은닉 상태와 출력을 이용하여 최종적으로 다음 디코더의 은닉 상태를 출력.

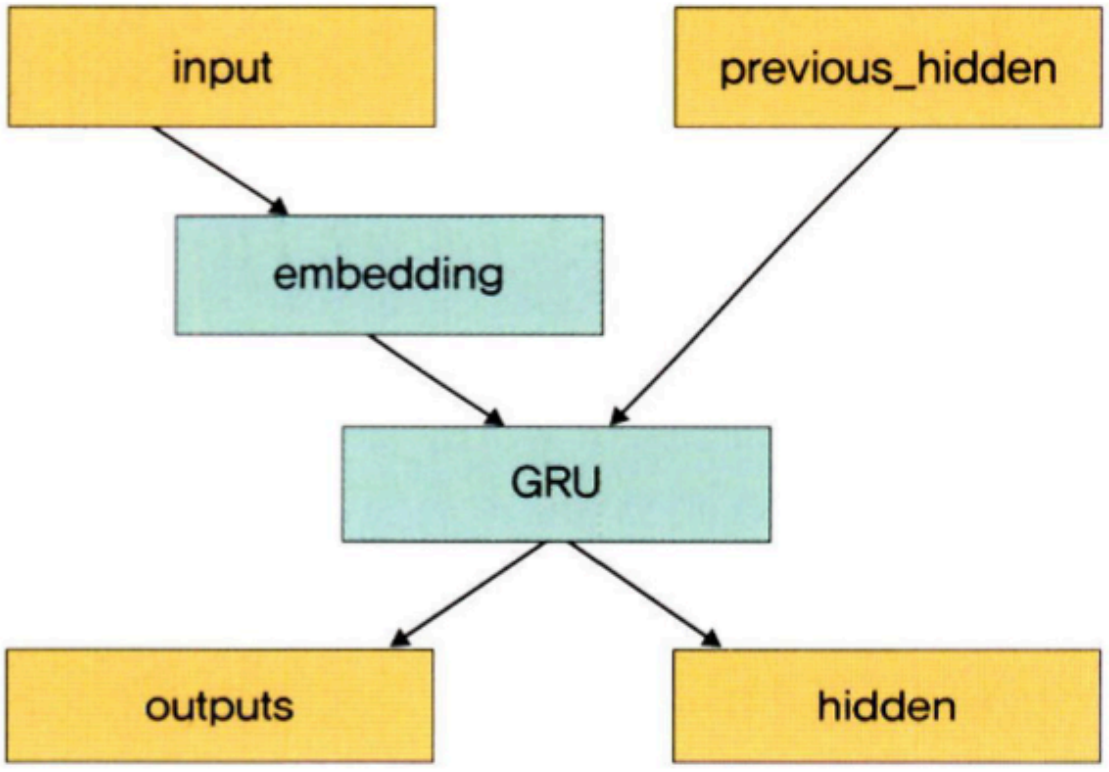
그림 10-13 어텐션 메커니즘 예시



seq2seq

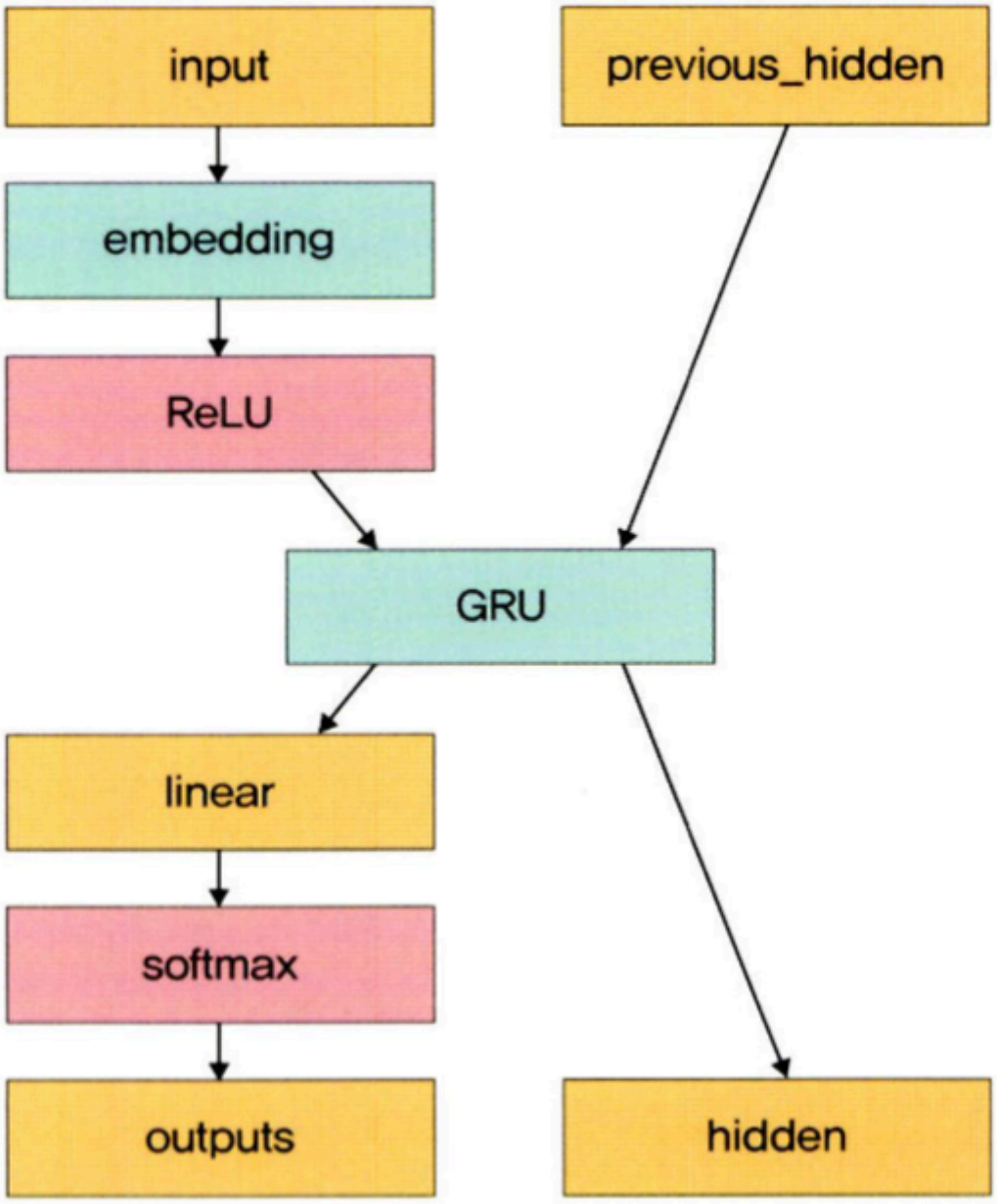
- 입력 시퀀스에 대한 출력 시퀀스를 만들기 위한 모델
- 인코더 : 임베딩 계층과 GRU 계층으로 구성됨.

그림 10-16 임베딩 네트워크



- 디코더 : 임베딩 계층, GRU 계층, 선형 계층으로 구성됨.

❮ 그림 10-17 디코더 네트워크



-
- 일반적인 seq2seq 모델 : 마지막의 은닉 상태만 디코더로 전달되므로 입력 문장이 긴 시퀀스일 경우 정확한 처리가 어려움. -> 어텐션 메커니즘 등장

버트

- 트랜스포머라는 인코더를 쌓아 올린 구조
- 다음과 같은 학습 절차를 거침
  1. 문장을 버트의 입력 형식에 맞게 변환
  2. 한 문장의 단어들에 대해 토큰화를 진행
  3. 각 토큰들에 대해 고유의 아이디를 부여