청년 AI Bigdata 교육

Tokenizer, Embedding

포항공대 NLP연구실 박연수, 이지현

Tokenizer(토큰화)

- Tokenizer(토큰화)
  - 자연어 처리에서 크롤링 등으로 얻어낸 코퍼스가 필요에 맞게 전처리 되지 않은 상태라면, 사용 용도에 맞게 토큰화가 필요
- 단어 토큰화
  - 단어, 또는 단어구를 토큰화 하는 방법
  - 입력 : Time is an illusion. Lunchtime double so!
  - 출력 : "Time", "is", "an", "illustion", "Lunchtime", "double", "so"

Tokenizer(토큰화)

- 토큰화에서 생기는 선택의 순간
  - 구두점 문제, 아래 것들을 같은 토큰으로 봐야하나?
    - Don't
    - Don t
    - Dont
    - Do n’t
- 주의점
  - 특수문자를 단순 제외해도 될까?
    - 문맥에 따라 특수 문자도 의미를 가질 수 있다.
  - 줄임말은 어떻게 해야할까?

Tokenizer(토큰화)

- 품사 기준 토큰화
  - Part of Speech (POS)방법
  - 단어 표기는 같지만 품사에 따라 단어의 의미가 달라질 수도 있어서, 단어 토큰화 이외에도 품사를 함께 표기하는 것이중요.
    - Fly : 날다(동사), 파리(명사)
  - 단어 토큰화 : ['I', 'am', 'actively', 'looking', 'for', 'Ph.D.', 'students', '.', 'and', 'you', 'are', 'a', 'Ph.D.', 'student', '.']
  - 품사 태깅 : [('I', 'PRP'), ('am', 'VBP'), ('actively', 'RB'), ('looking', 'VBG'), ('for', 'IN'), ('Ph.D.', 'NNP'), ('students', 'NNS'), ('.', '.'), ('and', 'CC'), ('you', 'PRP'), ('are', 'VBP'), ('a', 'DT'), ('Ph.D.', 'NNP'), ('student', 'NN'), ('.', '.')]

Tokenizer(토큰화)

- 문장 토큰화
  - 문서에서 문장을 분리하는 방법
  - 가장 간단한 방법 -> 마침표, 느낌표, 물음표로 구분하기
  - 하지만 마침표를 적용해서는 안되는 경우..
    - IP 192.168.56.31 서버에 들어가서 로그 파일 저장해서 aaa@gmail.com로 결과 좀 보내줘. 그 후 점심 먹으러 가자.
  - NLTK 에서는 영어 문장의 토큰화를 수행하는 sent tokenizer를 별도로 지원

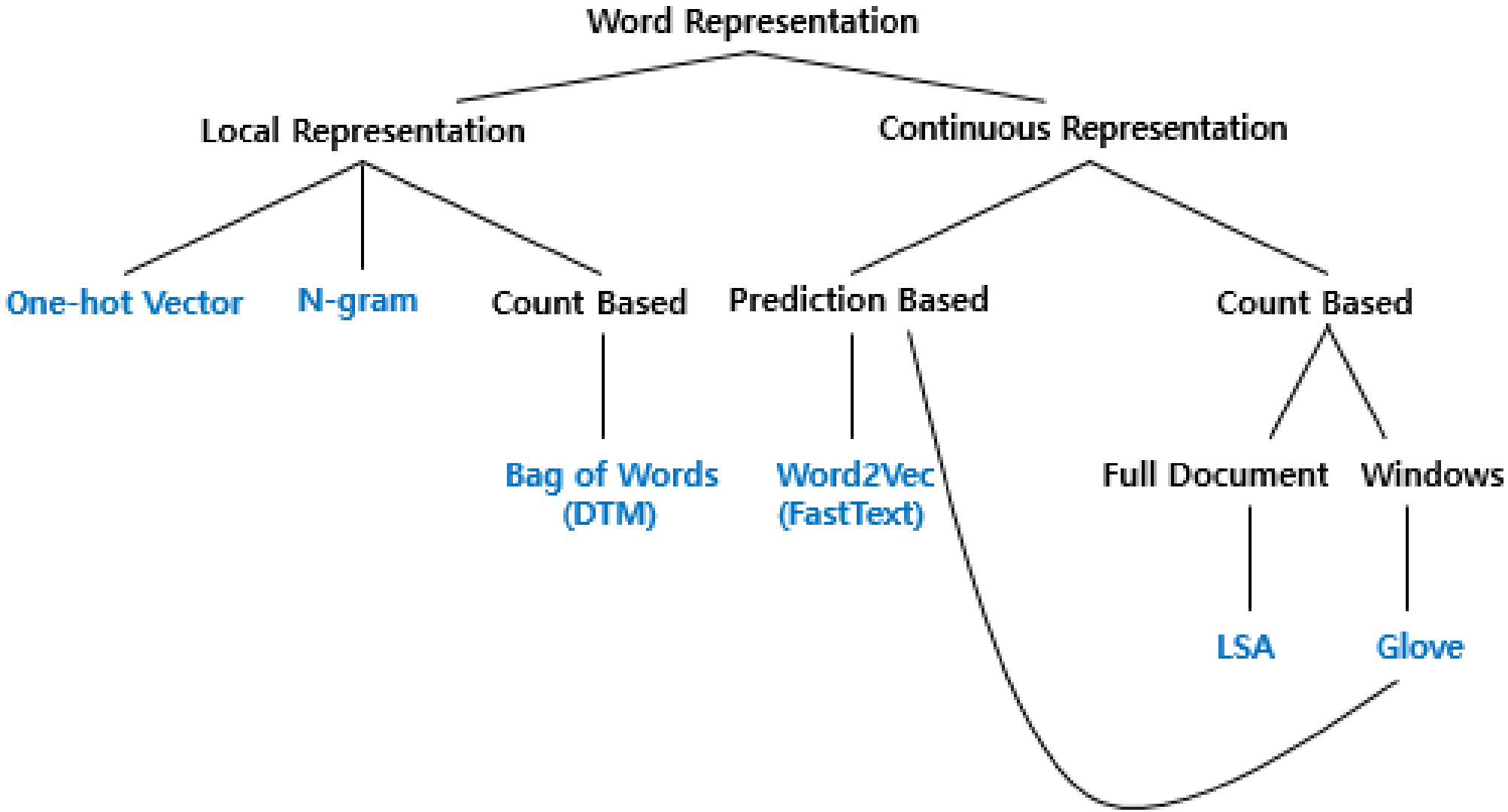
Tokenizer 실습

한국어 토크나이저 사용
영어 토크나이저 사용
영어 sentence tokenizer사용

단어의 표현 방법 (Embedding)

- 단어의 표현 방법 (Embedding)
  - Local Representation (Discrete Representation)
    - 강아지 : 1, 고양이 :2, 귀여운 :3 과 같이 단어 하나에 정해진 숫자를 매핑하는 방법
  - Distribution Representation (Continuous Representation)
    - 단어를 표현하기 위해 주변 단어를 참고하는 방법
    - 강아지와 귀여운 이라는 단어는 자주 같이 등장하므로, 강아지를 표현할 때 귀여운 이라는 단어를 이용하는 방법

단어의 표현 방법


Word Representation
Local Representation
Continuous Representation
One-hot Vector
N-gram
Count Based
Prediction Based
Count Based
Bag of Words
(DTM)
Word2Vec
(FastText)
Full Document
Windows
LSA
Glove

Local representation

- One-hot Encoding
  - “나는 자연어 처리를 배운다“
    - 나는 = [1,0,0,0]
    - 자연어 = [0,1,0,0]
    - 처리를 = [0,0,1,0]
    - 배운다 =[0,0,0,1]
  - 한계점:
    - 단어의 개수가 늘어날 수록 벡터를 저장하기 위한 공간이 계속 늘어난다
      - 단어의 개수가 1000개 -> 벡터 길이도 1000
    - 단어 사이의 유사도를 판단하지 못한다
      - 고양이 = [1,0,0,0] 강아지 [0,1,0,0] 호랑이 [0,0,1,0]

Local representation

- N-gram 언어 모델
  - “An adorable little boy is spreading smiles“
    - unigrams : an, adorable, little, boy, is, spreading, smiles
      bigrams : an adorable, adorable little, little boy, boy is, is spreading, spreading smiles
      trigrams : an adorable little, adorable little boy, little boy is, boy is spreading, is spreading smiles
      4-grams : an adorable little boy, adorable little boy is, little boy is spreading, boy is spreading smiles
  - 코퍼스에서 n개의 단어 뭉치 단위로 끊어서 이를 하나의 토큰으로 간주하고, 각 토큰에 번호 부여
  - 한계점
    - 희소문제 : n gram에서 나오지 않은 단어들은 표현 할 수 없다 (bigram의 경우 adorable boy표현 불가)
    - N의 적절한 크기를 선택하는 방법 :
      - N을 작게하면 주위 문맥을 반영을 적게하게 됨.
      - N을 크게하면 n-gram에 반영되지 않는 토큰들이 많아짐

Local representation

- Bag of Words (BoW)
  - 단어의 등장 순서를 고려하지 않는 빈도수 기반의 표현 방법 (문장, 문서단위 임베딩)
  - 만드는 방법
    - 각 단어에 고유한 정수 인덱스를 부여
    - 각 인덱스의 위치에 단어 토큰의 등장 횟수를 기록한 벡터 만드는 방법
  - 예시:
    - Doc1 : “정부가 발표하는 물가상승률과 소비자가 느끼는 물가상승률은 다르다.”
    - vocabulary : {'정부': 0, '가': 1, '발표': 2, '하는': 3, '물가상승률': 4, '과': 5, '소비자': 6, '느끼는': 7, '은': 8, '다르다': 9}
    - bag of words vector : [1, 2, 1, 1, 2, 1, 1, 1, 1, 1]
    - ‘정부’ 등장횟수 : 1 ‘가 ' 등장횟수 : 2

Local representation

- 문서 단어 행렬 (Document Term Matrix, DTM)
  - 서로 다른 문서들의 BoW를 결합한 방법
  - 문서 간 비교가 가능해짐
    - 문서1 : 먹고 싶은 사과
    - 문서2 : 먹고 싶은 바나나
    - 문서3 : 길고 노란 바나나 바나나
    - 문서4 : 저는 과일이 좋아요
  - 한계점
    - 희소 표현 (Sparcity)
      - 단어가 다양해질수록 벡터 공간의 크기가 커짐 (0으로 비워진 벡터수가 많아짐)
    - 단순 빈도 수 기반 접근
      - 문서1, 문서2의 유사도 비교할 때 두 벡터의 유사성을 판단 (뒷장에 자세히) 할 수도 있지만,,, The라는 단어가 동시에 많이 나왔다고 둘을 비슷한 문서로 볼 수 있을까?

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

https://wikidocs.net/24559

+ 벡터의 유사도 구하기

- Cosine similarity (코사인 유사도)
  - “방향”이 얼마나 유사한지 구하는 방법
  - 방향이 유사하면 유사도 1, 반대면 -1
    - 문서1 : 저는 사과 좋아요 ->[0,1,1,1]
      문서2 : 저는 바나나 좋아요 -> [1,0,1,1]
      문서3 : 저는 바나나 좋아요 저는 바나나 좋아요-> [2,0,2,2]
  - 문서 1,2의 유사도
    - 위 식에 A = [0,1,1,1], B = [1,0,1,1] 넣고 계산
      = 0.67

$$similarity = cos(\Theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

	바나나	사과	저는	좋아요
문서1	0	1	1	1
문서2	1	0	1	1
문서3	2	0	2	2

Local representation

- TF-IDF (Term Frequency – Inverse Document Frequency)
  - 기존 방법 : 문장에서의 단어의 빈도수를 그대로 사용
  - TF-IDF : 문서 집합전체의 단어 빈도수로 나눠줌
  - TF : term frequency : DTM과 동일. 한 문서에 단어가 몇번 나왔는지
  - Df : 특정 단어가 등장한 문서의 수:
    - '싶은' 단어의 등장횟수 =2 번
  - IDF:
    - DF의 반비례. 즉 IDF(싶은_) = ½

      실제로 사용할 때는 log와 분모에 1을 더해줌 (이유 : 문서수는 매우 많고, 분모에 0이 가는 것을 막기 위해)

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

$$idf(d,t) = log(\frac{n}{1+df(t)})$$

https://wikidocs.net/24

Local representation

- TF-IDF (Term Frequency – Inverse Document Frequency)
  - 기존 방법 : 문장에서의 단어의 빈도수를 그대로 사용
  - TF-IDF : 문서 집합전체의 단어 빈도수로 나눠줌
  - TF : term frequency : DTM과 동일. 한 문서에 단어가 몇번 나왔는지
  - Df : 특정 단어가 등장한 문서의 수:
    - '싶은' 단어의 등장횟수 =2 번
  - IDF:
    - DF의 반비례. 즉 IDF(싶은_) = ½

      실제로 사용할 때는 log와 분모에 1을 더해줌 (이유 : 문서수는 매우 많고, 분모에 0이 가는 것을 막기 위해)

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

$$idf(d,t) = log(\frac{n}{1+df(t)})$$

https://wikidocs.net/24

Local representation

- TF

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

- IDF

단어	IDF(역 문서 빈도)
과일이	ln(4/(1+1)) = 0.693147
길고	ln(4/(1+1)) = 0.693147
노란	ln(4/(1+1)) = 0.693147
먹고	ln(4/(2+1)) = 0.287682
바나나	ln(4/(2+1)) = 0.287682
사과	ln(4/(1+1)) = 0.693147
싶은	ln(4/(2+1)) = 0.287682
저는	ln(4/(1+1)) = 0.693147
좋아요	ln(4/(1+1)) = 0.693147

Local representation

- TF -IDF

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서 1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서 2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서 3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서 4	0.693147	0	0	0	0	0	0	0.693147	0.693147

Continuous Representation

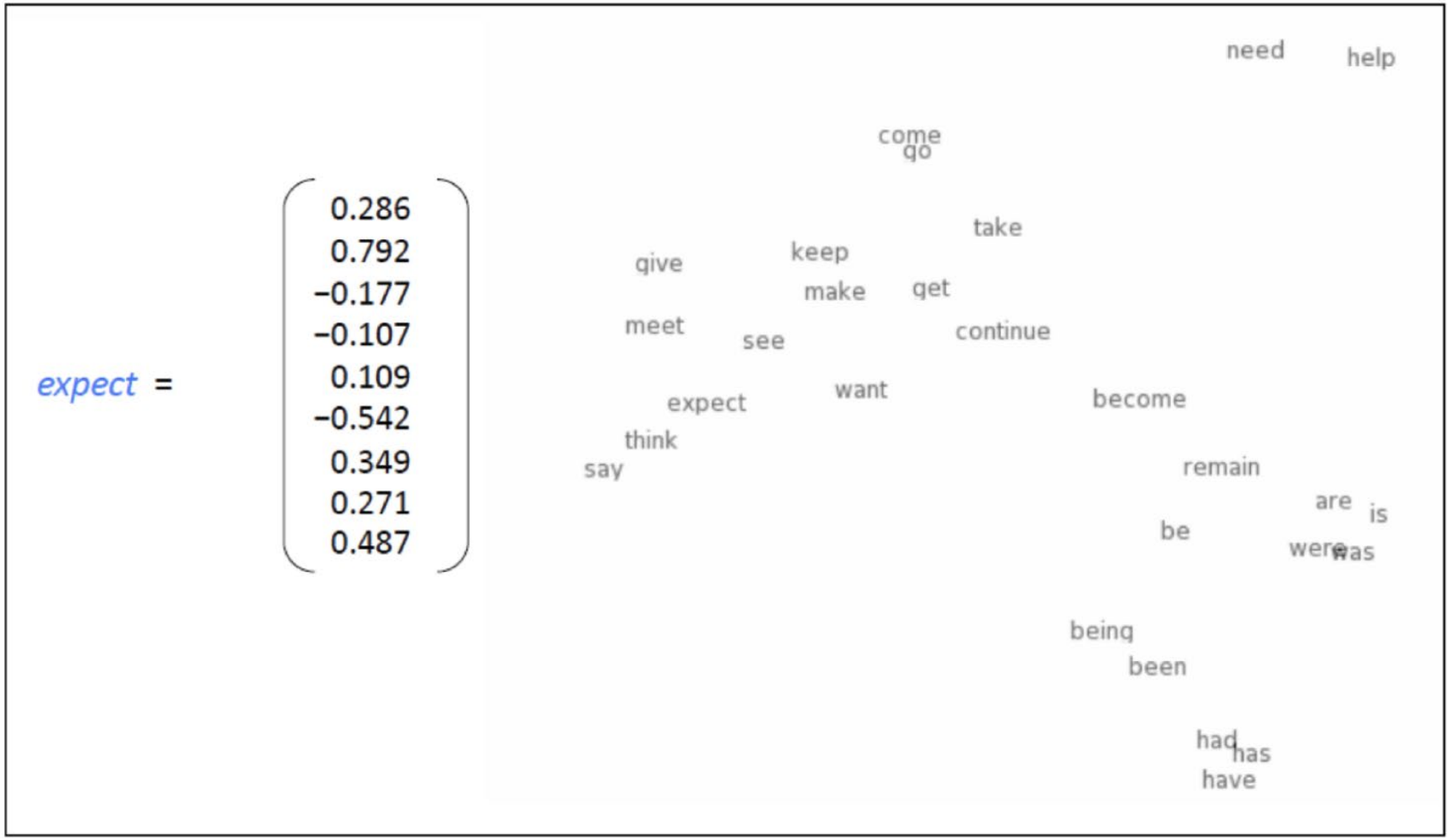
- 단어를 벡터로 표현하는 방법
  - 밀집 표현 : Dense representation
  - 기존 원-핫 벡터는 강아지 = [0,1,0,0,0,.] 으로 0 이 많은 구조
  - Dense representation 은 강아지 = [0.1, 0.3, 0.004 , …] : 차원이 밀집하게 표현
  - 원-핫벡터와는 다르게 학습 데이터를 통한 학습이 필요함

Continuous Representation

- Represent words as vectors

expect =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$



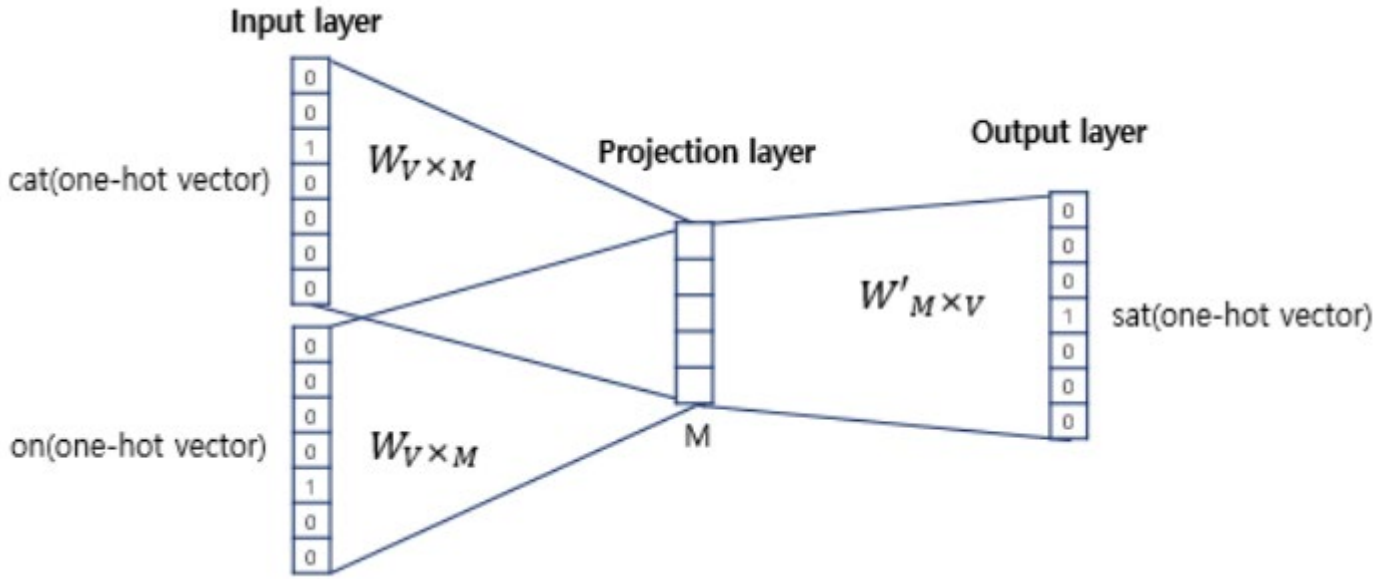
Continuous Representation

- CBOW방식
  - 중심단어와 주변단어 데이터셋 생성 (현재 sliding window = 2)

중심 단어	주변 단어
The fat cat sat on the mat	[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
The fat cat sat on the mat	[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
The fat cat sat on the mat	[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
The fat cat sat on the mat	[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
The fat cat sat on the mat	[0, 0, 0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
The fat cat sat on the mat	[0, 0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
The fat cat sat on the mat	[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]

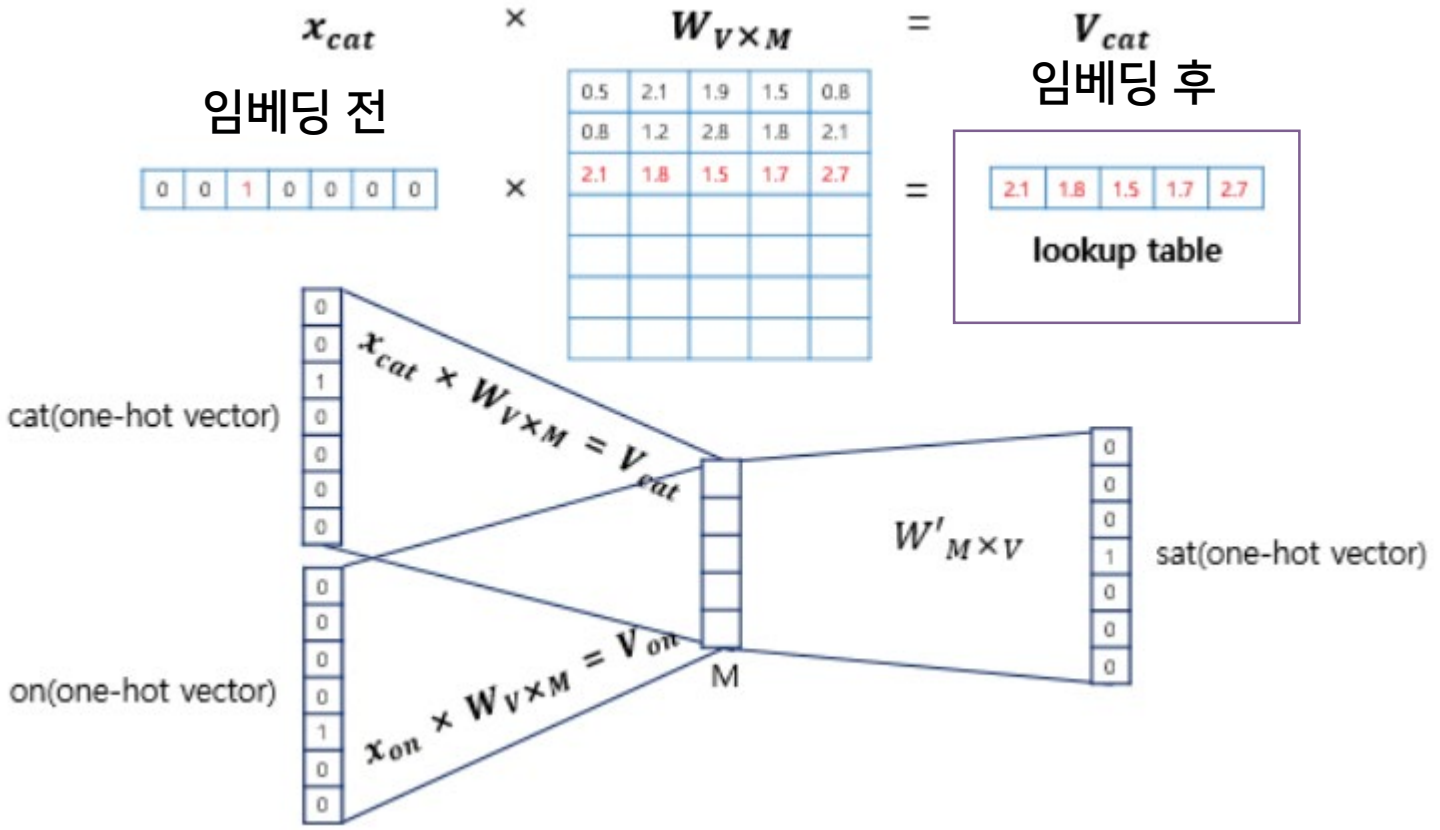
Continuous Representation

- CBOW방식
  - V : 전체 단어 수
  - M : hidden space
  - 가중치 벡터 W의 크기 (VxM)



Continuous Representation

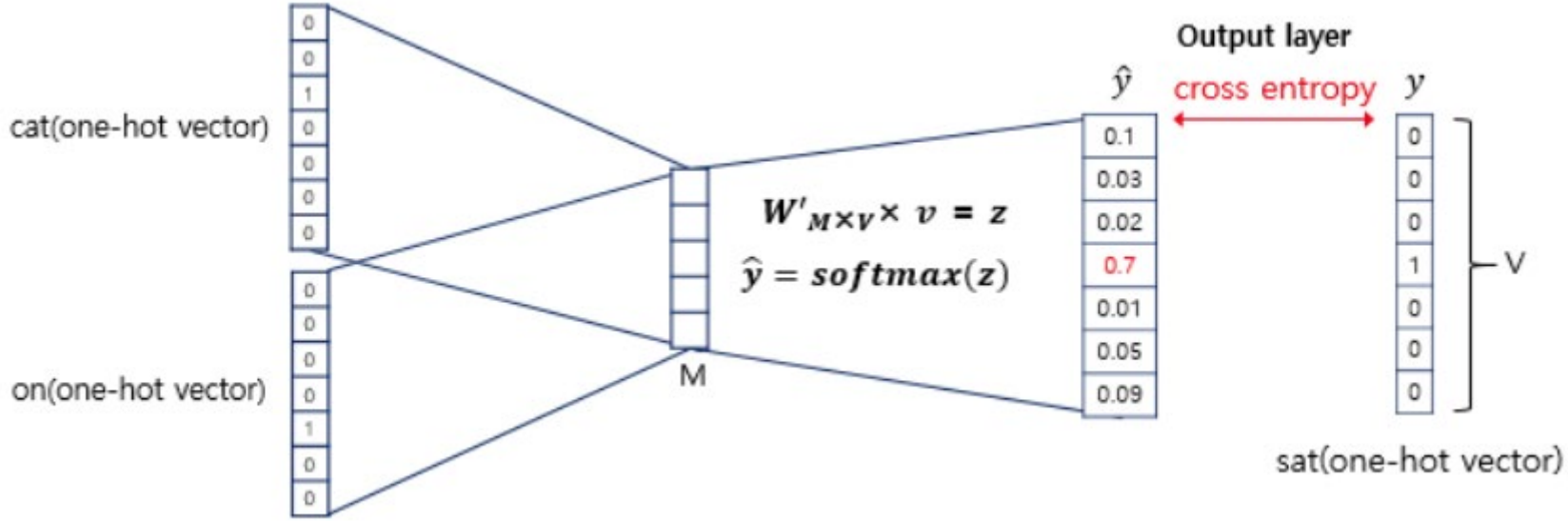
- CBOW방식
  - Cat x 가중치벡터
    - > Cat 에 대한 임베딩



Continuous Representation

$$H_p(q) = -\sum_{i=1}^{n} q(x_i) log p(x_i)$$

- CBOW방식
  - 가중치 벡터(W)는 어떻게 학습될까??
  - 학습과정 (MxV)크기의 가중치 벡터를 다시 곱해준 값과, one-hot vector 사이의 cross entropy로 가중치 벡터 학습



Embedding 실습

네이버 블로그 글 -> 명사만 추출 -> 글 주제별 cluster -> 시각화

Reference

- https://wikidocs.net/24557

코드를 작성해서 보여주세요

1 한국어 코퍼스 이용 (https://github.com/e9t/nsmc/blob/master/ratings_test.txt), 양이 많으므로, 500개만 사용

2 형태소를 분석하여 가장 많이 나온 명사, 동사의 막대그래프 그리기 (tokenizer.ipynb 참고)

```
from konlpy.tag import Okt # Okt tokenizer사용
nouns = [word for word, pos in morphs if pos.startswith('N') and word not in stopwords]
verbs = [word for word, pos in morphs if pos.startswith('V') and word not in stopwords]
```

3 '포스코' 를 제외한 명사만을 이용해서 counter vectorize만들기 (embedding.ipynb)

4 클러스터링 진행 n=4 로 수행 (embedding.ipynb)

5 Counter vectorizer 이용해서 tf-idf 행렬 만들기 (embedding.ipynb)

6 워드 클라우드 4개 결과 보여주기