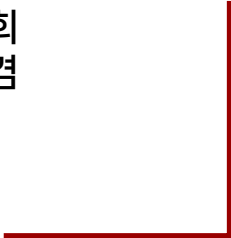




NLP

박진우 이영신 유건희
조상현 유재형 하은겸



원-핫 인코딩 (One-hot encoding)

- 분석하고자 하는 텍스트의 단어 사전(vocabulary) 생성
- 단어마다 단어사전 상의 인덱스 생성(정수 인코딩)
- 표현하고 싶은 단어의 인덱스의 위치에 1을 , 다른 단어의 위치에는 0 부여하는 벡터 생성

Vocabulary

{박진우: 0, 유재형: 1, 유건희: 2, 이영신: 3, 조상현 4, 하은겸: 5}

유재형

>

[0,1,0,0,0,0]

희소 표현 vs 밀집 표현

- 원-핫 인코딩은 벡터의 값이 대부분 0으로 표현되는 **희소 표현(sparse representation)**
- 단어 개수에 따라 차원이 무한정 증가, 단어의 의미를 담지 못함, 단어간 유사도 표현 불가

Harry Potter

“프리벳 가 4번지에 사는 더즐리 부부는 자신들이 정상적이라는 것을 아주 자랑스럽게..”

Vocabulary

{프리벳: 0, 가: 1, 4번지에: 2, 사는: 3, 더즐리: 4, 부부는: 5, ...}

해그리드

>

[0,0,0,0,...,1,0,0,...]

희소 표현 vs 밀집 표현

■ 반면에 사용자가 설정한 값으로 모든 단어의 벡터 표현의 차원을 맞추고, (0,1) 이 아닌 실수값으로 표현하는 것이 밀집 표현(Dense representation)

■ 분포 가설 하에서 단어간 유사성을 표현하기 위해서 단어의 의미를 벡터화

분포 가설: '**비슷한 위치에서 등장하는 단어들은 비슷한 의미를 가진다**' 라는 가정

■ 분산 표현을 이용하여 단어를 벡터화하는 것이 워드 임베딩(embedding),
이렇게 표현된 벡터가 임베딩 벡터(embedding vector)

해그리드

>

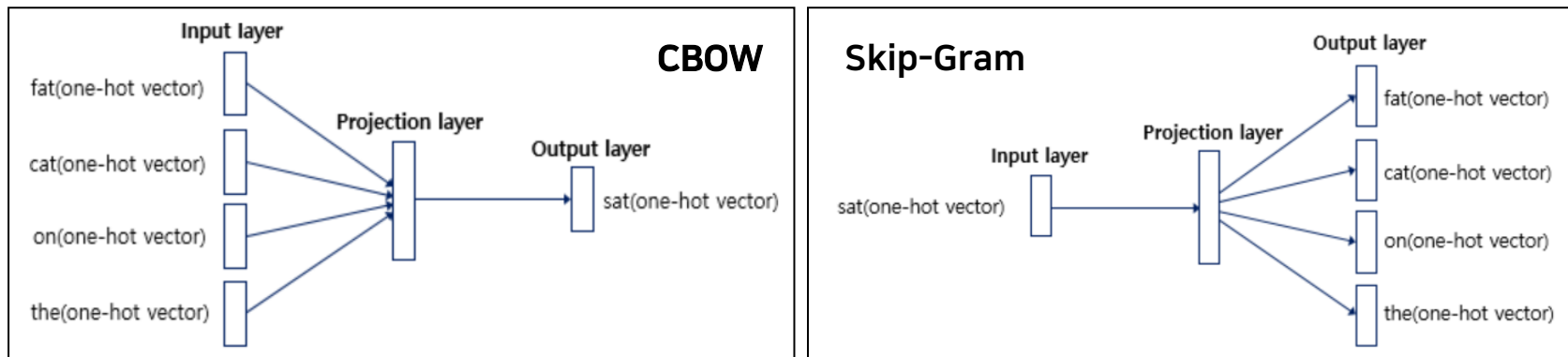
[0,0,0,0,...,1,0,0,...]

>

[0.05,0.2,...,0.7,0.2,...]

워드 투 벡터 (Word2Vec)

- 최근 들어 워드 임베딩 학습 방법으로 속도가 빠른 Word2Vec 이 가장 많이 사용됨
- 주변의 단어들로 중심 단어를 예측하는 CBOW 방식과,
중간의 단어로 주변 단어를 예측하는 Skip-Gram 방식 존재.



CBOW

■ 중심단어를 기준으로 앞 뒤 몇 단어를 참고? > 윈도우 설정. (2*윈도우 크기 의 단어를 참고)



The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

중심 단어	주변 단어
[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
[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]
[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]
[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]
[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, 0], [0, 0, 0, 0, 0, 0, 1]
[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]
[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]

문장 "The fat cat sat on the mat" 를 학습할 때 사용되는 전체 데이터셋

Window= 2

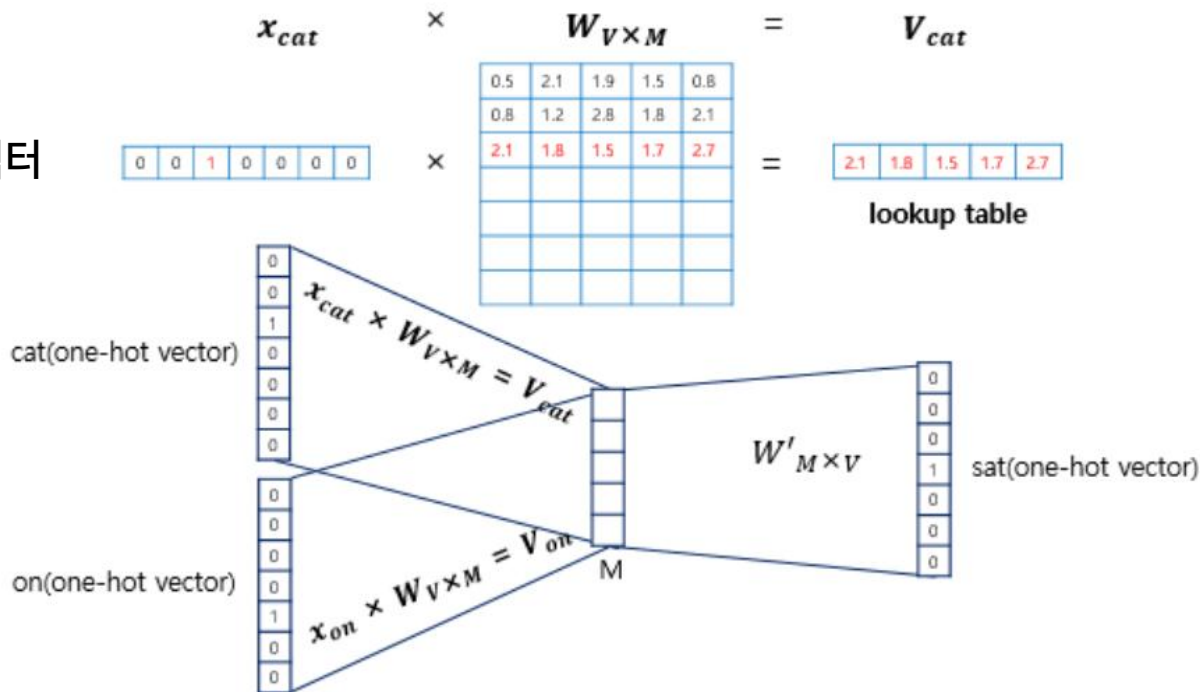
CBOW

■ 가중치 행렬 W, W'

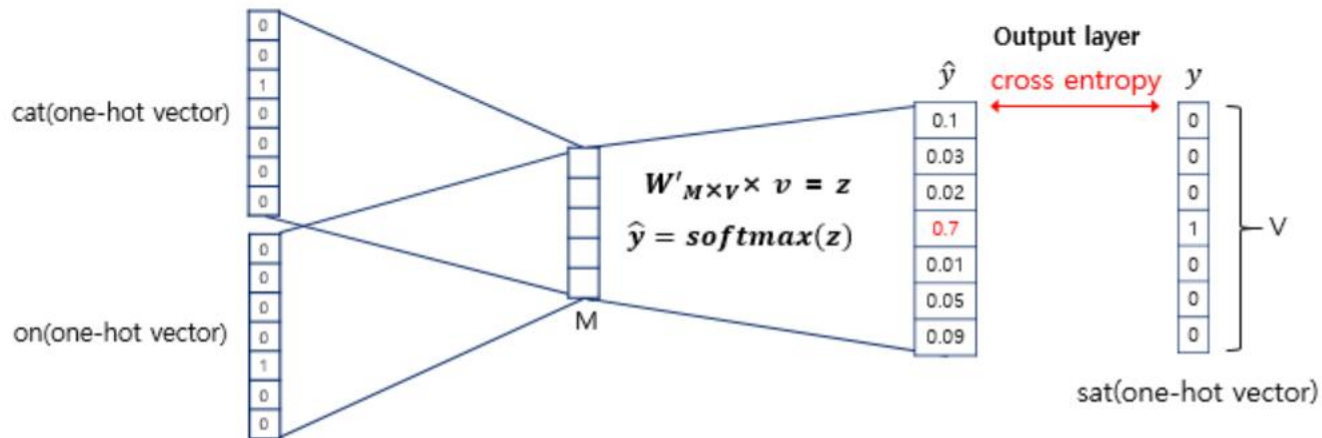
■ $W = M$ 차원 크기의 임베딩 벡터

■ CBOW의 목적은 W 와 W' 를

잘 훈련시키는 것.



CBOW



- 각 주변단어들에 W 행렬을 곱해 만든 임베딩 벡터의 평균 인 벡터 생성
- 평균 벡터가 두번째 가중치 행렬 W'와 곱해짐 (활성함수: softmax / 손실함수: cross-entropy)

시 분석

■ 윤동주, 김수영, 정지용 등 시인 별로 작품들을 분석

■ 예상: 시인별로 Word2Vec를 실행하고 유사도를 살펴본다면 각 시인의 특징, 성격을

파악 할 수 있지 않을까?



```
%time tokens = sentences.apply(tokenizer.tokenize)
tokens[:3]
```



```
CPU times: user 10.6 ms, sys: 1.03 ms, total: 11.6 ms
```

```
Wall time: 11.5 ms
```

```
article_id
```

```
1    [죽는, 날까지, 하늘을, 우러러, 한점, 부끄럼이, 없기를, 앞새에, 이는, 바람...
2    [산모퉁이를, 돌아, 논가, 외딴, 우물을, 홀로, 찾아가선, 가만히, 들여다봅니다...
3    [여기저기서, 단풍잎, 같은, 슬픈, 가을이, 똑똑, 떨어진다, 단풍잎, 떨어져, ...
```

```
Name: content, dtype: object
```

전처리 후
토큰화 하는 과정

시 분석 (윤동주, 김수영)



```
model.wv.most_similar('별')
```

윤동주



```
/usr/local/lib/python3.6/dist-packages/  
if np.issubdtype(vec.dtype, np.int):  
[('그', 0.305417001247406),  
 ('하나에', 0.2465084046125412),  
 ('붉은', 0.2306281477212906),  
 ('별을', 0.21637648344039917),  
 ('않은', 0.1899402141571045),  
 ('그러나', 0.17066815495491028),  
 ('남은', 0.16877669095993042),  
 ('까닭입니다', 0.1676110029220581),  
 ('게외다', 0.16152667999267578),  
 ('없어', 0.15842044353485107)]
```



```
[ ] model.wv.most_similar("사랑")
```

김수영



```
/usr/local/lib/python3.6/dist-packages/  
if np.issubdtype(vec.dtype, np.int):  
[('나', 0.6953631639480591),  
 ('것', 0.6799353361129761),  
 ('너', 0.6703925132751465),  
 ('우리', 0.6572244167327881),  
 ('의', 0.6554654240608215),  
 ('그', 0.6509417295455933),  
 ('수', 0.6410539150238037),  
 ('이', 0.6406413912773132),  
 ('더', 0.64027339220047),  
 ('때', 0.6390254497528076)]
```

■ 의미 해석이 어려운, 예상과 다른 결과

소설 분석 (해리 포터)

■ 동일한 방식으로 소설 “해리 포터 4권: 해리포터와 불의 잔” 단어간 유사도 계산

```
[ ] model.wv.most_similar("헤르미온느")
```

```
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737:  
    if np.issubdtype(vec.dtype, np.int):  
    [('론', 0.9745050668716431),  
     ('지니', 0.9260266423225403),  
     ('깜짝', 0.9248721599578857),  
     ('해리', 0.9195702075958252),  
     ('몹시', 0.9184021949768066),  
     ('듯이', 0.9162657260894775),  
     ('네빌', 0.91167151927948),  
     ('겁', 0.9111292362213135),  
     ('놀란', 0.9105744361877441),  
     ('귀', 0.9067764282226562)]
```

■ 시에 비해 의미 해석이 쉬움

문제점 & 개선방안

■ 동일한 방법으로 분석을 수행하였으나 시의 단어간 유사도 분석에서 한계를 보임

■ 예상되는 문제점:

-시의 매체적 특성 (짧은 문장 길이, 함축적 의미 전달, 문법 파괴 등)

-전처리 과정에서의 문제 (불용어 처리, 대명사, 동일 의미의 여러 형태 혼재)

-소설 등에 비해 충분하지 않은 텍스트 양

■ 차후 계획: 위의 문제점 보완을 통해 시의 자연어 처리 가능 여부를 검토하고,

seq2seq, Attention mechanism, Transformer 등 활용한 다양한 모델 구현 시도



감사합니다

