Natural Language Process

KU-BIG

Index

1. What is NLP
2. Text Preprocessing & Vectorization
3. Model

1. What is NLP?

1. 자연어란?

: 우리가 일상 생활에서 사용하는 언어.

2. 자연어처리란?

: 자연어의 의미를 분석하여 컴퓨터가 처리 할 수 있도록 하는 것!

따라서, 기계에게 인간의 언어를 이해시킨다는 것!

1. What is NLP?

3. 활용

: 번역, 챗봇, 감정분석, 텍스트 분류 등.

2. Text Preprocessing

1. 텍스트 전처리란?

: 용도에 맞게 텍스트를 사전에 처리하는 작업

2. 필요성

- 기계가 텍스트를 이해할 수 있도록 텍스트를 정제
- 신호와 소음을 구분
- 아웃라이어 데이터로 인한 overfitting 방지

2. Text Preprocessing

Tokenization (토큰화)

- 의미있는 단위로 나누는 작업.

Ex) KU-BIG은 정말 좋은, 유익한, 사랑스러운 학회야.

```
from nltk.tokenize import WordPunctTokenizer
print(WordPunctTokenizer().tokenize("KU-BIG은 정말 유익한, 행복한, 사랑스러운 학회야!"))
```

```
['KU', '-', 'BIG은', '정말', '유익한', ',', '행복한', ',', '사랑스러운', '학회야', '!']
```

! 구두점이나 특수 문자를 단순 제외해서는 안된다.

2. Text Preprocessing

Cleaning(정제)

- 정제(cleaning) : 갖고 있는 데이터로부터 노이즈 데이터를 제거하는 과정.

(1) 등장 빈도가 적은 단어

: 너무 적게 등장하여 자연어 처리에 있어 도움이 되지 않는 단어들

(2) 길이가 짧은 단어

영어 : 길이 2~3인 단어를 제거함으로 의미 없는 단어들을 줄이는데 크게 효과적
Ex) it, at, to, on, in , by
한국어 : 한국어의 경우에 적용하기는 힘듦.

2. Text Preprocessing

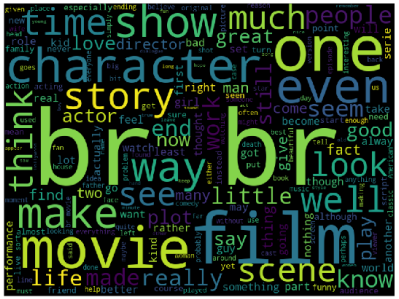
(3) 불용어(Stopword)

- 자주 등장하면서 분석에 있어 큰 도움이 되지 않는 것.

```
from nltk.corpus import stopwords
stopwords.words('english')[:10]
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

(4) 특수문자, 숫자, HTML 태그 등



"With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again.

→ HTML 태그

2. Text Preprocessing

Normalization(정규화)

- 정규화(normalization) : 표현 방법이 다른 단어들을 통합시켜 같은 단어로 만들어준다.

(1) 표기가 다른 단어의 통합
: 단어 인덱싱을 할 때 같은 의미임에도 다르게 인덱싱되는 것을 방지하기 위해
ex) KU-BIG, 빅데이터 학회, 쿠빅 …

(2) 대, 소문자 통합
: 효율적인 데이터의 개수를 줄이는 방법으로 주로 소문자로 변환을 해서 통일을 시켜줌.

(3) 어간추출(Stemming)
: 단어의 기본 형태를 추출하는 단계

새로운, 새로울 -> 새롭다 Ate, will eat, eating -> eat

2. Vectorization

원핫인코딩 (One-hot Encoding)

(1) 각 단어에 고유한 인덱스를 부여함. (정수 인코딩)

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical

text = "빅데이터 학회 정말 정말 최고야 너도 빅데이터 학회 돌어오지 않을래"

t = Tokenizer()
t.fit_on_texts([text])
print(t.word_index)
```

```
{'빅데이터': 1, '학회': 2, '정말': 3, '최고야': 4, '너도': 5, '돌어오지': 6,
'않을래': 7}
```

```
sub_text = "빅데이터 학회 최고야 너도 최고야"
encoded=t.texts_to_sequences([sub_text])[0]
print(encoded)
```

```
[1, 2, 4, 5, 4]
```

(2) 표현할 단어의 인덱스의 위치에 1을 부여, 다른 단어의 인덱스에는 0을 부여함.

```
one_hot = to_categorical(encoded)
print(one_hot)
```

```
[[0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0.]]
```

2. Vectorization

카운트기반의 벡터화 - TF/IDF

TF(Term Frequency) – IDF(Inverse Document Frequency)

d : 특정 문서, t : 단어, n : 문서의 총 개수

tf(d,t) : 특정 문서 d에서의 특정 단어 t의 등장 횟수.

df(t) : 특정 단어 t가 등장한 문서의 수.

idf(d,t) : df(t)에 반비례하는 수.

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

2. Vectorization

```
'먹고 싶은 사과',
'먹고 싶은 바나나',
'길고 노란 바나나 바나나',
'저는 과일이 좋아요'
```

-	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

2. Vectorization

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

-	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

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

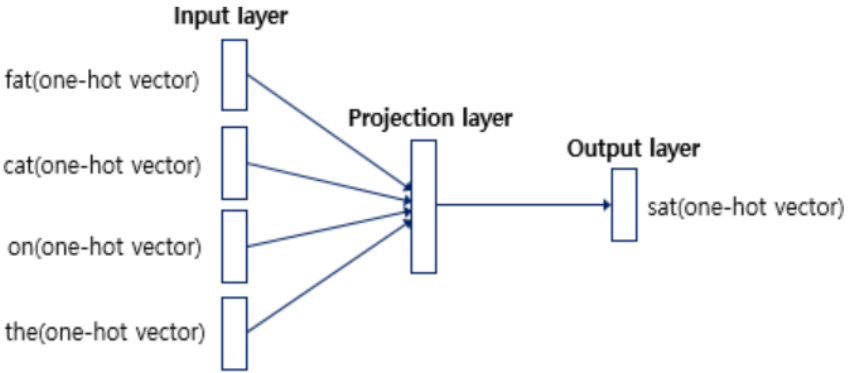
2. Vectorization

예측기반 벡터화 - Word2Vec

1)CBOW : 친구를 보면 그 사람을 안다.

중심 단어 / 주변 단어

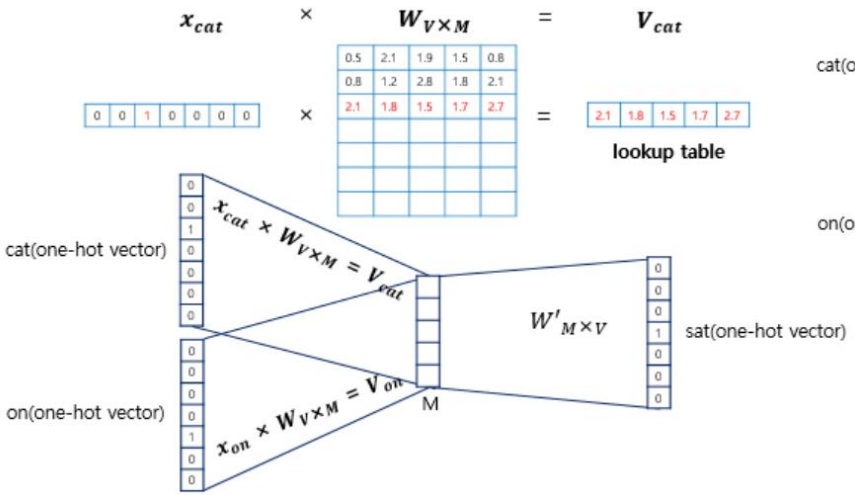
	중심 단어	주변 단어
The fat cat sat on the mat	[1, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0]
The fat cat sat on the mat	[0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 1, 0, 0]
The fat cat sat on the mat	[0, 0, 1, 0, 0, 0]	[1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 0, 1,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

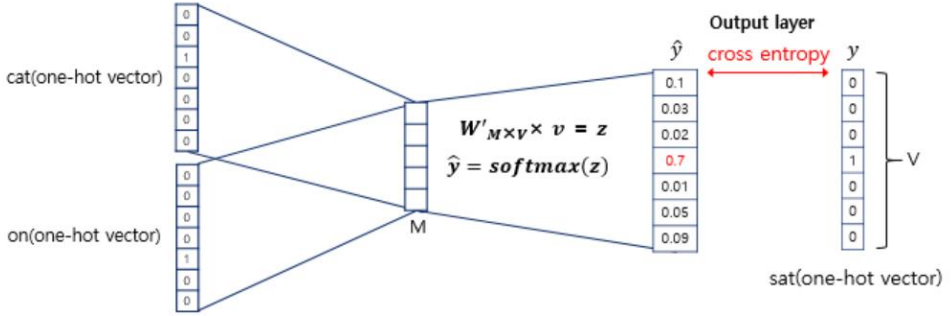


KU-BIG

2. Vectorization

1)CBOW

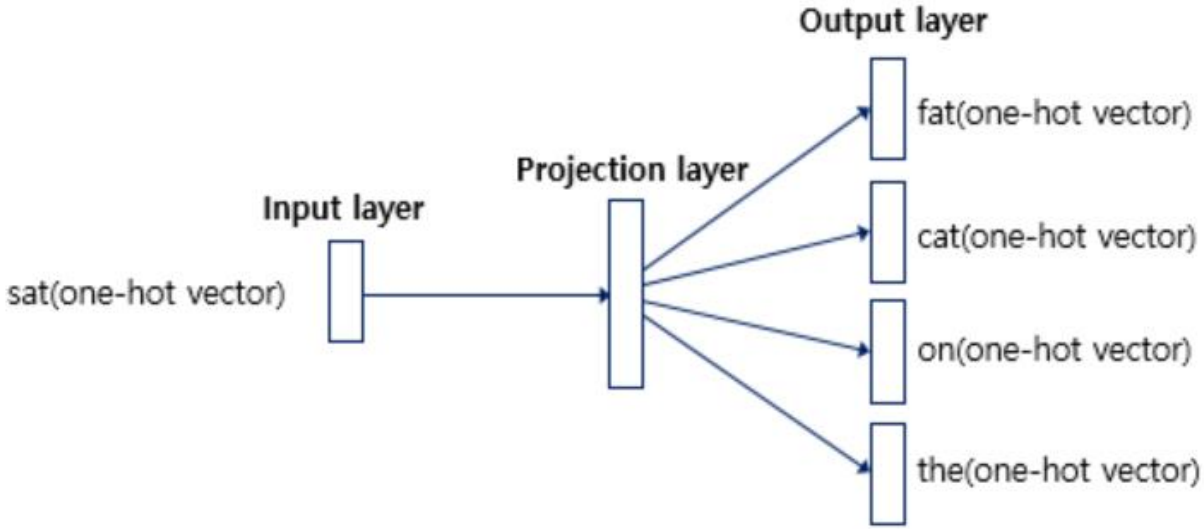




$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

2. Vectorization

2)Skip-gram



단어간 유사도 측정에 강점

복잡한 특징까지도 잘 잡아냄.

Skip-gram이 더 효과가 좋다.

2. Vectorization

3) GloVe(Global Vectors for Word Representation)

카운트 기반의 단점 => 단어 의미의 유추 작업에서의 성능이 떨어짐.

예측 기반의 단점 => 윈도우 크기 내에서만 주변 단어를 고려, 전체적인 통계 정보 반영에 떨어짐.