DTM, TF-IDF

Bag of Words(BoW)

Bag of words는 단어의 등장 순서를 고려하지 않는 빈도수 기반의 단어 표현 방법

문서3: 정부가 발표하는 물가상승률과 소비자가 느끼는 물가상승률은 다르다. 소비자는 주로 소비하는 상품을 기준으로 물가상승률을 느낀다.

```
vocabulary : {'정부': 0, '가': 1, '발표': 2, '하는': 3, '물가상승률': 4, '과': 5, '소비자': 6, '느끼는': 7, '은': 8, '다르다': 9, '는': 10, '주로': 11, '소비': 12, '상품': 13, '을': 14, '기준': 15, '으로': 16, '느낀다': 17}
bag of words vector : [1, 2, 1, 2, 3, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1]
```

BoW는 각 단어가 등장한 횟수를 수치화하는 텍스트 표현 방법이므로 주로 어떤 단어가 얼마나 등장했는지를 기준으로 문서가 어떤 성격의 문서인지를 판단하는 작업에 쓰임

```python
from sklearn.feature_extraction.text import CountVectorizer

corpus = ['you know I want your love. because I love you.']
vector = CountVectorizer()

# 코퍼스로부터 각 단어의 빈도수를 기록
print('bag of words vector :', vector.fit_transform(corpus).toarray())

# 각 단어의 인덱스가 어떻게 부여되었는지를 출력
print('vocabulary :',vector.vocabulary_)
```

```
bag of words vector : [[1 1 2 1 2 1]]
vocabulary : {'you': 4, 'know': 1, 'want': 3, 'your': 5, 'love': 2, 'because': 0}
```

문서 단어 행렬(Document-Term Matrix, DTM)

문서 단어 행렬(Document-Term Matrix, DTM)이란 다수의 문서에서 등장하는 각 단어들의 빈도를 행렬로 표현한 것

문서1 : 먹고 싶은 사과
문서2 : 먹고 싶은 바나나
문서3 : 길고 노란 바나나 바나나
문서4 : 저는 과일이 좋아요

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

문서 단어 행렬(Document-Term Matrix, DTM)

DTM의 한계점

DTM에서의 각 행(각 문서 벡터)의 차원은 전체 단어 집합의 크기
→ 문서 벡터의 차원은 수만 이상의 차원을 가질 수도 있음
→ 대부분의 값이 0을 가질 수도 있음 : 희소 표현(sparse representation)
→ 많은 양의 저장 공간과 높은 계산 복잡도 요구

각 문서에는 중요한 단어와 불필요한 단어들이 혼재되어 있음

예: 불용어(stopwords)는 빈도수가 높더라도 자연어 처리에 있어 의미를 갖지 못하는 단어

TF-IDF(Term Frequency-Inverse Document Frequency)

TF-IDF(Term Frequency-Inverse Document Frequency)는 단어의 빈도와 문서 빈도를 사용하여 DTM 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법

TF-IDF는 주로 문서의 유사도를 구하는 작업, 검색 시스템에서 검색 결과의 중요도를 정하는 작업, 문서 내에서 특정 단어의 중요도를 구하는 작업 등에 쓰임

(1) tf(d,t): 특정 문서 d에서의 **특정 단어 t의 등장 횟수**
(2) df(t) : 특정 단어 t가 등장한 **문서의 수**
(3) idf(t) : df(t)에 반비례하는 수
    - 단, 분모에 1을 더하고 로그를 취함
    - 총 문서의 수 n이 커질 수록, IDF의 값이 기하급수적으로 커지는 것을 방지

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

- $TF-IDF(t,d) = tf(d,t) * idf(t)$

```
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [
    'you know I want your love',
    'I like you',
    'what should I do ',
]

tfidfv = TfidfVectorizer().fit(corpus)
print(tfidfv.transform(corpus).toarray())
print(tfidfv.vocabulary_)
```

참고자료

- 딥 러닝을 이용한 자연어 처리 입문(https://wikidocs.net/book/2155)