텍스트 빈도수 분석 모듈 제작

TECHNOLOGY & PROGREMMER

목차

Part1. 자연어 처리 (NLP: Natural Language Processing)

1. 정의

- 언어 모델(Language Model, LM)은 언어라는 현상을 모델링하고자 단어 시퀀스(문장)에 확률을 할당(assign)하는 모델로서 은 특정 문장(=단어의 나열)이 등장할 확률을 계산해줌
- 언어 모델을 만드는 방법은 크게는 통계를 이용한 방법과 인공 신경망을 이용한 방법으로 구분할 수 있으며 최근에는 통계를 이용한 방법보다는 인공 신경망을 이용한 방법이 더 좋은 성능을 보여주고 있음.
- 문장에 대해 확률을 계산할 수 있다는 건, 단어(혹은 문장)를 적절하게 선택하거나 생성해야하는 경우 여러 후보 중에서 더 적절한(확률이 높은) 후보를 선택하는 데에 사용될 수 있음.

언어모델 종류	• Unigram 언어 모델 • n-gram 언어 모델

Part1. 자연어 처리 (NLP: Natural Language Processing)

2. 언어모델 종류

Unigram 언어 모델 (bag of words)

$$P_{uni}(w_1, w_2, w_3, \cdots) = P(w_1)P(w_2)P(w_3)\cdots$$

제일 단순한 언어모델로서 각각의 단어가 출현할 확률이 독립이라고 가정하고 모두 따로따로 계산.

- 각각의 단어를 독립적으로 보기 때문에 단어의 순서를 전혀 고려하지 않게 됨.
- 이 모델에서는 전체 문장에서 각 어휘가 몇 번씩 등장했는지만 고려함.
- 학습용 말뭉치에서 각 어휘의 빈도를 조사해서 등장 확률을 계산

확률을 추정하는 것과 계산하는 것이 매우 빠르기 때문에 노이즈 데이터를 1차로 필터링하는데 사용되며 SentencePiece라는 서브워드 토크나이저에서 토큰의 유용성을 판단하는 데에도 사용됨

```
txt=['파이썬 차트 파이썬 머신러닝',
     '차트 파이썬 R 차트',
      'R 분석 시각화'
     ]
```

	R	머신러닝	분석	시각화	차트	파이썬
0	0	1	0	0	1	2
1	1	0	0	0	2	1
2	1	0	1	1	0	0

Part1. 자연어 처리 (NLP: Natural Language Processing)

2. 언어모델 종류

n-gram 언어 모델

$$P_2(w_1, w_2, w_3, \cdots) = P(w_1)P(w_2|w_1)P(w_3|w_2)\cdots$$

Unigram과 달리 이전 단어를 고려하여 다음 단어의 확률을 계산
n-gram 모델에서는 다음 단어를 예측하기 위해 이전의 n−1개의 단어를 활용

- n은 보통 2~6개 선택함
- 2-gram 모델예: a 다음에 b가 나올 확률 계산 → a? 의 빈도수 / ab의 빈도수
- 3-gram 모델예: aba 다음에 c가 나올 확률 계산 → abc의빈도수 / ab?의 빈도수
- 출현빈도수가 0인 자료에 대한 문제해결: 학습 말뭉치에 등장하지 않는 패턴에 대해 적당히 추정하는 Smoothing 기법

Kneser-ney 언어 모델

Kneser-ney smoothing은 n-gram 모델의 여러 smoothing 기법 중에서 가장 높은 성능을 내는 것으로 알려져 있음

- ab 다음에 c가 등장할 확률을 계산할 때, b 다음에 c가 등장할 확률도 함께 고려함
- b 다음 c가 등장할 확률을 계산할 때도 c가 등장할 확률도 함께 고려함.
- 따라서 만약 abc가 등장한 빈도가 0일지라도, bc가 등장한 빈도와 그냥 c가 등장한 빈도가 모두 0일리는 없으므로, 최종적으로 항상 0보다 큰 확률값을 계산해낼 수 있게 됨

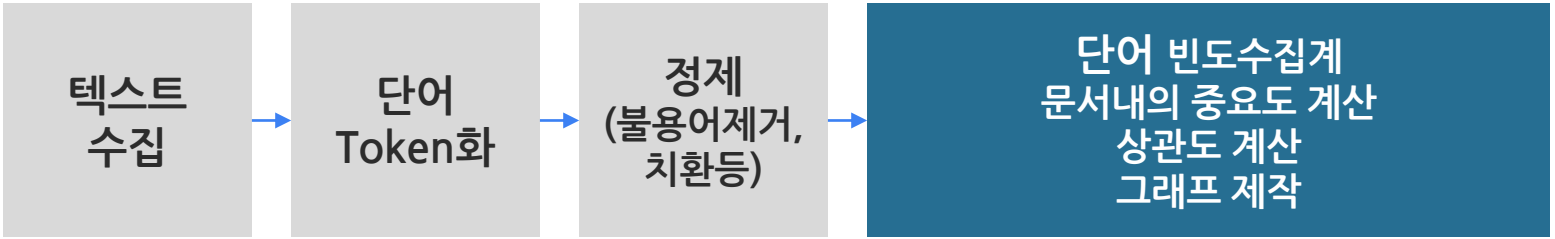
Part2. Count기반 단어 표현 모델

1. TF-IDF((Term Frequency - Inverse Document Frequency) 개념

- 정보 검색과 텍스트 마이닝에서 이용하는 가중치로, 여러 문서로 이루어진 문서군이 있을 때 어떤 단어가 특정 문서 내에서 얼마나 중요한 것인지를 나타내는 통계적 수치임.
  (기본적으로 단어의 빈도 수를 이용한 수치화 방법이기 때문에 단어의 의미를 고려하지 못한다는 단점이 있음)

- TF-IDF(Term Frequency-Inverse Document Frequency)는 단어의 빈도와 역 문서 빈도(문서의 빈도에 특정 식을 취함)를 사용하여 DTM 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법임.

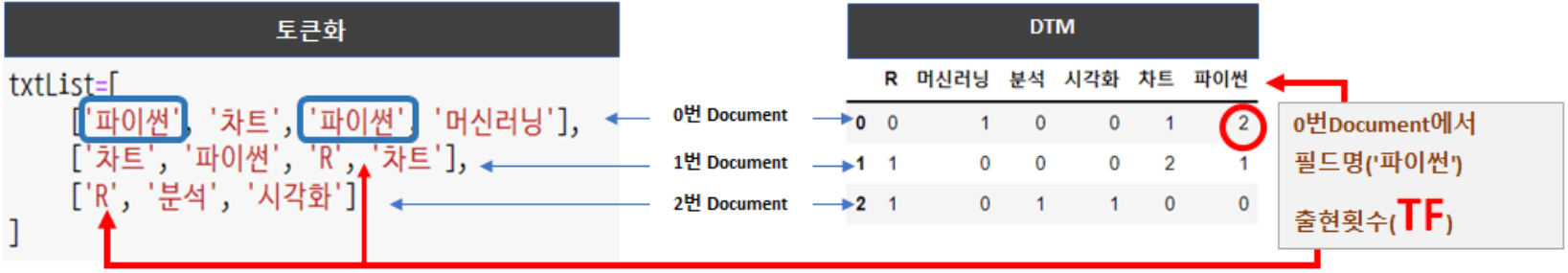
- 우선 DTM을 만든 후, TF-IDF 가중치를 부여하며 TF-IDF는 주로 문서의 유사도를 구하는 작업, 검색 시스템에서 검색 결과의 중요도를 정하는 작업, 문서 내에서 특정 단어의 중요도를 구하는 작업 등에 쓰일 수 있음.
- 사이킷런에서 TfidfVectorizer 패키지로 사용해도 되나 한글과 영문숫자가 혼용된경우 한글만 처리하는 경우가 있어 상황에 따라 직접 모듈을 제작해야함.



Part2. Count기반 단어 표현 모델

2. DTM과 TF

- DTM: 문서(Document)별의 단어(Term) 출현 빈도수(TF)를 가로 세로의 행열(matrix) 단위로 표시한 자료
- TF(Term Frequency): 어떤 단어가 특정 문서에 얼마나 많이 쓰였는지의 횟수



'파이썬' TF

- 0번 Documents에서 '파이썬' 단어 TF → 2
- 1번 Documents에서 '파이썬' 단어 TF → 1
- 2번 Documents에서 '파이썬' 단어 TF → 0

•단어 Martrix DTM 제작시 비어있는 자료(0값)가 훨씬더 많은 희소(Sparse) 행렬이 됨.

Part2. Count기반 단어 표현

3. 파이썬 DTM 패키지 (사이킷런 CountVectorizer 모듈 사용시)

★★★★★★ 주의: 사이킷런 패키지에서 작성한 DTM의 문제점 ★★★★★★★★★★★

직접작성한 DTM

	R	머신러닝	분석	시각화	차트	파이썬
0	0	1	0	0	1	2
1	1	0	0	0	2	1
2	1	0	1	1	0	0

CountVectorizer 모듈로 작성한 TDM
[한글, 영문 혼합시 영문 단어 계산못함]

	머신러닝	분석	시각화	차트	파이썬
0	1	0	0	1	2
1	0	0	0	2	1
2	0	1	1	0	0

Part2. Count기반 단어 표현 모델

4. TF-IDF

- **TF(Term Frequency)** : 특정 단어가 하나의 데이터 안에서 등장하는 횟수
- **DF(Document Frequency)** : 특정 단어가 여러 데이터에 자주 등장하는지를 알려주는 지표.
- **IDF(Inverse Document Frequency)** : DF에 역수를 취해(inverse) 구함
- **TF-IDF** : TF와 IDF를 곱한 값. 즉 TF가 높고, DF가 낮을수록 값이 커지는 것을 이용하는 것
  여러문서에 출현빈도가 높으면 TF-IDF값은 낮아짐

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

값2 (DF)

값1
Document(문서)
갯수 4개

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
0	0	0	0	1	0	1	1	0	0
1	0	0	0	1	1	0	1	0	0
2	0	1	1	0	2	0	0	0	0
3	1	0	0	0	0	0	0	1	1

예) 단어 '먹고'의 **IDF**는 **ln(값1/(값2+1))**

참고) ln: 자연로그(밑값이 자연상수 e인 로그)

값1: Document 갯수 (행의갯수)

값2: DF(Document Frequency)

Document 전체에서의 단어출현횟수 (행에 값이 False가 아닌경우)

===> ln(4/(2+1)) == 0.287682

DF는 자기의 문서에 몇번을 나오던 1번으로 카운팅하여서 총 문서(여기서는 4개행)에서 몇번 false가 아닌지를 계산하는 자료임.

예) 바나나 TF

0번문서 TF: 0 / 1번문서 TF: 1
2번문서 TF: 2 / 3번문서 TF: 0

예) 바나나 DF 는 2

0번 문서 tf가 0 임으로 Fasle
1번문서 tf가 1이상임으로 True
2번문서 tf가 1이상임으로 True
3번 문서 tf가 0 임으로 False
True의 횟수 2임

Part2. Count기반 단어 표현 모델

4. TF-IDF

•출처: https://wikidocs.net/31698

문서의 총 수는 4이기 때문에 ln 안에서 분자는 늘 4으로 동일합니다. 분모의 경우에는 각 단어가 등장한 문서의 수(DF)를 의미하는데, 예를 들어서 '먹고'의 경우에는 총 2개의 문서(문서1, 문서2)에 등장했기 때문에 2라는 값을 가집니다. 각 단어에 대해서 IDF의 값을 비교해보면 문서 1개에만 등장한 단어와 문서2개에만 등장한 단어는 값의 차이를 보입니다. IDF는 여러 문서에서 등장한 단어의 가중치를 낮추는 역할을 하기 때문입니다.

그러면 이제 TF-IDF를 계산해보도록 하겠습니다. TF는 DTM을 그대로 가져오면 각 문서에서의 각 단어의 TF를 가져오게 되기 때문에, 앞서 사용한 DTM에서 단어 별로 위의 IDF값을 그대로 곱해주면 TF-IDF가 나오게 됩니다.

-	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

사실 예제 문서가 굉장히 간단하기 때문에 계산은 매우 쉽습니다. 문서3에서의 바나나만 TF 값이 2이므로 IDF에 2를 곱해주고, 나머진 TF 값이 1이므로 그대로 IDF 값을 가져오면 됩니다. 문서2에서의 바나나의 TF-IDF 가중치와 문서3에서의 바나나의 TF-IDF 가중치가 다른 것을 볼 수 있습니다. 수식적으로 말하면, TF가 각각 1과 2로 달랐기 때문인데 TF-IDF에서의 관점에서 보자면 TF-IDF는 특정 문서에서 자주 등장하는 단어는 그 문서 내에서 중요한 단어로 판단하기 때문입니다. 문서2에서는 바나나를 한 번 언급했지만, 문서3에서는 바나나를 두 번 언급했기 때문에 문서3에서의 바나나를 더욱 중요한 단어라고 판단하는 것입니다.

Part3. 파이썬 프로그램 제작

1. 사용자정의함수를 이용한 TF-IDF 모듈 제작

- 모듈이란? 함수나 변수 또는 클래스를 모아 놓은 파일 로서 다른 파이썬 프로그램에서 불러와 사용할 수 있음
- 모듈은 직접 함수나 클래스로 구성하여서 작성하여 사용할 수 있음.

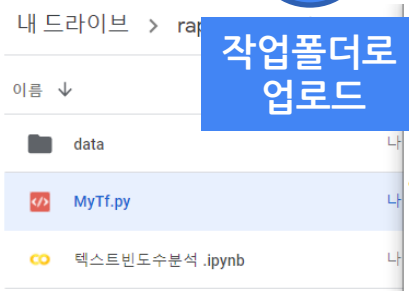
3

모듈 사용

```
import MyTf as tf
vocab = tf.split(docs)
result[-1].append(tf.tfidf(t,d,docs)
```

2

작업폴더로 업로드



1

모듈제작후 MyTf.py로 저장

MyTf.py - Windows 메모장

```
def tf(t, d):
    return d.count(t)

def idf(t,docs):
    N = len(docs)
    df = 0
    for doc in docs:
        df += t in doc
    return log(N/(df + 1))

def tfidf(t, d,docs):
  return tf(t,d)* idf(t,docs)


def split(docs):
  return list(set(w for doc in docs for w in doc.split()))
```

Part3. 파이썬 프로그램 제작

2. 사이킷런 모듈 사용 & 확률적 경사하강법을 이용한 문장 유사도 구하기

```
###################################
# 확률적 경사하강법을 이용한 문장 유사도 구하기
###################################

from sklearn.linear_model import SGDClassifier
from sklearn.feature_extraction.text import
TfidfVectorizer

corpus = [

  '전화 문의 불편 빠른 응답 전화 친절',
  '안내 불편 전화 무응답 전화 안내 불편',
  '배달비 비쌈 배달 느림 배달 불친절 안내 불편',
  '배달 용기 변경 요망 배달 속도 개선'

]
```

```
tfidfv = TfidfVectorizer().fit(corpus)

X=tfidfv.fit_transform(corpus)
y=[1,2,3,4]

model=SGDClassifier(loss='perceptron')
model.fit(X,y)

newX=tfidfv.transform(['전화 연결 안됨'])
pred=model.predict(newX)
pred
```

출력물:
array([1])