8장 텍스트 분석

텍스트 분석 개요

NLP: 머신이 인간의 언어를 이해하고 해석하는데 더 중점. ex)텍스트 마이닝

언어를 해석하기 위한 기계번역, 질의응답 시스템에서 차이가 존재함.

텍스트 분석: 머신러닝, 언어 이해, 통계 등을 활용해 모델을 수립하고 정보를 추출해 비즈니스 인텔리전스(Business Intelligence)예측 분석 등의 분석 작업을 주로 수행

- 텍스트 분류:문서가 특정 카테고리에 속하는 것을 판단.
- 감성 분석: 텍스트에서 나타나는 감정, 판단 등의 주관적 요소를 분석하는 기법
- 텍스트 요약: 중심 사상을 추출하는 기법-토픽 모델링
- 텍스트 군집화: 유사도를 측정해서 비슷한 문서끼리 모을 수 있음.

1. 텍스트 분석 이해

- 비정형 데이터인 텍스트를 분석하는 것 → 이를 피처 형태로 추출하고 피처에 의미있는 값을 부여하는가가 중요한 포인트임.
- 피처 벡터화(피처 추출): 텍스트는 단어의 조합인 벡터값으로 나타냄 방법으로는 BOW와 WORD2VEC 방법이 있음.

- 머신러닝 기반 텍스트 분석 프로세스

1. 텍스트 사전 준비작업:

텍스트를 피처로 만들기 전에 미리 클렌징, 대/소문자 변경, 특수문자 삭제
등의 클렌징 작업, 단어(Word) 등의 토큰화 작업, 의미 없는 단어(Stop word) 제거 작업, 어근 추출(Stemming/Lemmatization) 등의 텍스트 정규화 작업을 수행하는 것을 통칭

2. 피처 벡터화 추출: 사전 준비 작업으로 가공된 텍스트에서 피처를 추출하고 여기에 벡터 값을 할당
3. ML 모델 수립 및 학습,평가: 피처 벡터화된 데이터 세트에 ML 모델을 적용해 학습/예측 및 평가를 수행.

- 파이썬 기반의 NLP, 텍스트 분석 패키지

NLTK(Natural Language Toolkit for Python)： 파이썬의 가장 대표적인 NLP 패키지입니다. 방대한 데이터 세트와 서브 모듈을 가지고 있으며 NLP의 거의 모든 영역을 커버, 수행 속도가 아쉬움.

Gensim： 토픽 모델링 분야에서 가장 두각을 나타내는 패키지입니다. 오래전부터 토픽 모델링을 쉽게 구현할 수 있는 가능을 제공해 왔으며, Word2Vec 구현 등의 다양한 신기능도 제공합니다. SpaCy와 함께 가장 많이 사용되는 NLP 패키지

SpaCy： 뛰어난 수행 성능으로 최근 가장 주목을 받는 NLP 패키지입니다. 많은 NLP 애플리케이션에서 SpaCy를 사용하는 사례가 늘고 있습니다.

2. 텍스트 사전 준비 작업

- 클렌징: 불필요한 문자, 기호 등을 사전에 제거하는 작업
- 텍스트 토큰화:

-문장 토큰화: 문장의 마침표, 개행 문자 등 문장의 마지막을 뜻하는 기호에 따라 분리하는 것.

-단어 토큰화: 문장을 단어로 토큰화. 정규 표현식을 활용할 수도 있음.

문장을 단어별로 토큰화 할 경우, 문맥적 의미는 무시될 수밖에 없음. → n-gram을 활용.

-n-gram: 연속된 n개의 단어를 하나의 토큰화 단위로 분리하는 것을 말함. 연속적으로 2개의 단어들을 순차적으로 이동하면서 단어들을 토큰화

- 스톱 워드 제거:

분석에 큰 의미가 없는 단어를 지칭합니다. 가령 영어에서 is, the, a, will 등 문장을 구성하는 필수 문법 요소지만 문맥적으로 큰 의미가 없는 단어가 이에 해당

- stemming과 lemmatization:

문법적 또는 의미적으로 변화하는 단어의 원형을 찾는 것.

Lemmatization이 Stemming보다 정교하며 의미론적인 기반에서 단어의 원형을 찾습니다.

stem: 원래 단어에서 일부 철자가 훼손된 어근 단어를 추출하는 경향.

lem: 문법적인 요소와 의미적 부분을 더 감안. lemmatize()의 파라미터로 동사의 경우 V, 형용사의 경우 a를 입력

3. Bag of Words

:문서가 가지는 모든 단어(Words)를 문맥이나 순서를 무시하고 일괄적으로 단어에 대해 빈도 값을 부여해 피처 값을 추출하는 모델

- word counts 프로세스

문장 1:

'My wife likes to watch baseball games and my daughter likes to watch baseball games too'

문장 2:

'My wife likes to play baseball'

1. 문장 1과 문장 2에 있는 모든 단어에서 중복을 제거하고 각 단어(feature 또는 term)를 칼럼 형태로 나열합니다. 그러고 나서 각 단어에 고유의 인덱스를 다음과 같이 부여합니다.

   'and': 0, 'baseball': 1, 'daughter': 2, 'games': 3, 'likes': 4, 'my': 5, 'play': 6, 'to': 7, 'too': 8, 'watch': 9, 'wife': 10

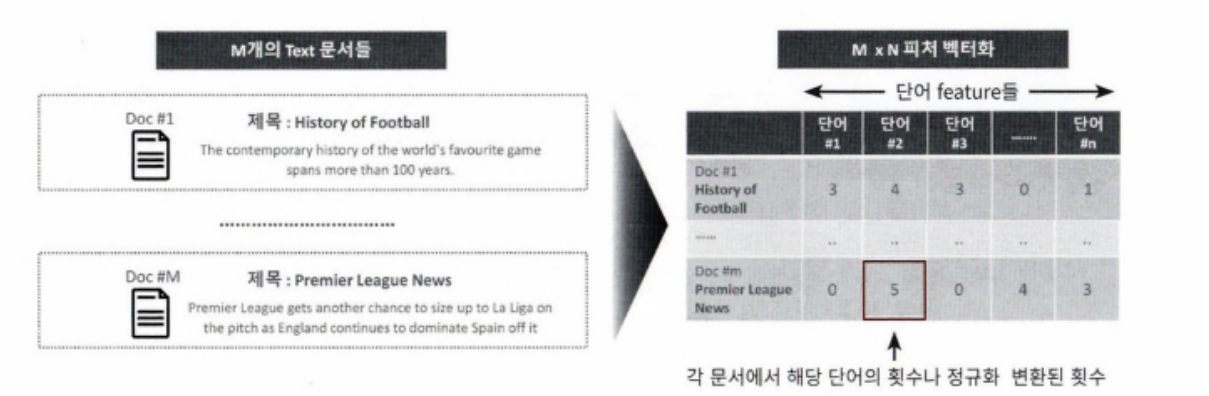
2. 개별 문장에서 해당 단어가 나타나는 횟수(Occurrence)를 각 단어(단어 인덱스)에 기재합니다. 예를 들어 baseball은 문장 1, 2에서 총 2번 나타나며, daughter는 문장 1에서만 1번 나타납니다.

	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7	Index 8	Index 9	Index 10
	and	baseball	daughter	games	likes	my	play	to	too	watch	wife
문장 1	1	2	1	2	2	2		2	1	2	1
문장 2		1			1	1	1	1			1

→ 문장 1에서 baseball은 2회 나타남

- 장점: 쉽고 빠른 구축. 예상보다 문서의 특징을 잘 나타낼 수 있는 모델.
- 단점: 문맥의 의미 반영 부족. 희소 행렬(구성된 행렬의 대부분의 값이 0으로 채워지는 행렬) 문제. 문서가 너무 많을 경우, 문서의 단어가 다시 나타나지 않을 수 있기 때문에 희소행렬이 나타날 가능성이 높음.
- BOW의 피처 벡터화:

모든 문서의 모든 단어를 칼럼 형태로 나타내고 각 문서에서 해당 단어의 횟수나 정규화된 빈도를 값으로 부여하는 데이터 셋 모델로 변경하는 것을 말함.



〈 BOW 피처 벡터화 〉

-카운트 기반의 벡터화: 나타내는 횟수를 Count를 부여하는 경우. 카운트 벡터화에서는 카운트 값이 높을수록 중요한 단어로 인식.

사이킷런의 CountVectorizer

파라미터 명	파라미터 설명
max_df	전체 문서에 걸쳐서 너무 높은 빈도수를 가지는 단어 피처를 제외하기 위한 파라미터입니다. 너무 높은 빈도수를 가지는 단어는 스톱 워드와 비슷한 문법적인 특성으로 반복적인 단어일 가능성이 높기에 이를 제거하기 위해 사용됩니다. max_df = 100과 같이 정수 값을 가지면 전체 문서에 걸쳐 100개 이하로 나타나는 단어만 피처로 추출합니다. Max_df = 0.95와 같이 부동소수점 값(0.0 ~ 1.0)을 가지면 전체 문서에 걸쳐 빈도수 0~95%까지의 단어만 피처로 추출하고 나머지 상위 5%는 피처로 추출하지 않습니다.
min_df	전체 문서에 걸쳐서 너무 낮은 빈도수를 가지는 단어 피처를 제외하기 위한 파라미터입니다. 수백~수천 개의 전체 문서에서 특정 단어가 min_df에 설정된 값보다 적은 빈도수를 가진다면 이 단어는 크게 중요하지 않거나 가비지(garbage)성 단어일 확률이 높습니다. min_df = 2와 같이 정수 값을 가지면 전체 문서에 걸쳐서 2번 이하로 나타나는 단어는 피처로 추출하지 않습니다. min_df = 0.02와 같이 부동소수점 값(0.0 ~ 1.0)을 가지면 전체 문서에 걸쳐서 하위 2% 이하의 빈도수를 가지는 단어는 피처로 추출하지 않습니다.
max_features	추출하는 피처의 개수를 제한하며 정수로 값을 지정합니다. 가령 max_features = 2000으로 지정할 경우 가장 높은 빈도를 가지는 단어 순으로 정렬해 2000개까지만 피처로 추출합니다.
stop_words	'english'로 지정하면 영어의 스톱 워드로 지정된 단어는 추출에서 제외합니다.
n_gram_range	Bag of Words 모델의 단어 순서를 어느 정도 보강하기 위한 n_gram 범위를 설정합니다. 튜플 형태로 (범위 최솟값, 범위 최댓값)을 지정합니다. 예를 들어 (1, 1)로 지정하면 토큰화된 단어를 1개씩 피처로 추출합니다. (1, 2)로 지정하면 토큰화된 단어를 1개씩(minimum 1), 그리고 순서대로 2개씩(maximum 2) 묶어서 피처로 추출합니다.
analyzer	피처 추출을 수행한 단위를 지정합니다. 당연히 디폴트는 'word'입니다. Word가 아니라 character의 특정 범위를 피처로 만드는 특정한 경우 등을 적용할 때 사용됩니다.
token_pattern	토큰화를 수행하는 정규 표현식 패턴을 지정합니다. 디폴트 값은 '\b\w\w+\b로, 공백 또는 개행 문자 등으로 구분된 단어 분리자(\b) 사이의 2문자(문자 또는 숫자, 즉 영숫자) 이상의 단어(word)를 토큰으로 분리합니다. analyzer= 'word'로 설정했을 때만 변경 가능하나 디폴트 값을 변경할 경우는 거의 발생하지 않습니다.
tokenizer	토큰화를 별도의 커스텀 함수로 이용시 적용합니다. 일반적으로 CountTokenizer 클래스에서 어근 변환 시 이를 수행하는 별도의 함수를 tokenizer 파라미터에 적용하면 됩니다.

-TF-IDF 기반의 벡터화: 자주 나타나는 단어에 높은 가중치를 주되, 모든 문서에서 전반적으로 자주 나타나는 단어에 대해서는 페널티를 주는 방식으로 값을 부여. 문서마다 텍스트가 길고, 개수가 많은 경우 이 방식이 더 좋은 예측 성능을 보장할 수 있음.

사이킷런의 TfidfVectorizer 은 counts와 같음.

- 벡터화를 위한 희소 행렬

희소 행렬은 너무 많은 불필요한 0값이 메모리를 차지하고 연산이 오래 걸린다는 단점이 있음 → COO, CSR 활용.

COO: 0이 아닌 데이터만 별도의 데이터 어레이에 저장하고, 그 데이터가 가리키는 행과 열의 위치를 또 별도로 저장하는 방식.

CSR: Compressed Sparse Row의 약자이며, 이처럼 행 위치 배열 내에 있는 고유한 값의 시작 위치만 다시 별도의 위치 배열로 가지는 변환 방식을 의미

5. 감성 분석

- 지도 학습의 감성 분석: 학습 데이터와 타깃 레이블 값을 기반으로 감성 분석 학습을 수행한 뒤 이를 기반으로 다른 데이터의 감성 분석을 예측하는 방법으로 일반적인 텍스트 기반의 분류와 거의 동일
- 비지도 학습의 감성 분석: ‘Lexicon’이라는 일종의 감성 어휘 사전을 이용합니다. Lexicon은 감성 분석을 위한 용어와 문맥에 대한 다양한 정보를 가지고 있으며. 이를 이용해 문서의 긍정적, 부정적 감성 여부를 판단할 수 있음.

Lexicon: 긍정과 부정을 나타내는 감성 지수를 가지고 있음. → NLTK의 lexicon은 WorldNet을 활용하여서 감성 지수를 부여한 것임. 실습에서는 SentiWordNet,Vador 활용.