08 텍스트 분석

NLP는 머신이 인간의 언어를 이해하고 해석하는 데 더 중점을 두고 기술이 발전해 왔다.

텍스트 분석은 비정형 텍스트에서 의미 있는 정보를 추출하는 것에 좀 더 중점을 두고 기술이 발전해 왔다.

NLP는 기계 번역, 질의응답 시스템의 영역에서 텍스트 분석과 차별점이 있다. **NLP는 텍스트 분석을 향상하게 하는 기반 기술**이라고 볼 수 있다.

텍스트 분석의 영역은 다음과 같다.

- **텍스트 분류** : Text Categorization이라고도 한다. 문서가 특정 분류 또는 카테고리에 속하는 것을 예측하는 기법을 통칭한다. 예를 들어 특정 신문 기사 내용이 연애/정치/사회/문화 중 어떤 카테고리에 속하는 자동으로 분류하거나 스팸 메일 검출 같은 프로그램이 이에 속한다. 지도학습을 적용한다.
- **감성 분석** : 텍스트에서 나타나는 감정/판단/믿음/의견/기분 등의 주관적인 요소를 분석하는 기법을 총칭한다. 소셜 미디어 감정 분석, 영화나 제품에 대한 긍정 또는 리뷰, 여론조사 의견 분석 등의 다양한 영역에서 활용된다. Text Analytics에서 가장 활발하게 사용되고 있는 분야다. 지도학습 방법 뿐만 아니라 비지도학습을 이용해 적용할 수 있다.
- **텍스트 요약** : 텍스트 내에서 중요한 주제나 중심 사상을 추출하는 기법을 말한다. 대표적으로 토픽 모델링이 있다.
- **텍스트 군집화와 유사도 측정** : 비슷한 유형의 문서에 대해 군집화를 수행하는 기법을 말한다. 텍스트 분류를 비지도학습으로 수행하는 방법의 일환으로 사용될 수 있다.

01 텍스트 분석 이해

텍스트 분석은 **비정형 데이터인 텍스트를 분석하는 것**이다.

비정형 텍스트 데이터를 어떻게 피처 형태로 추출하고 추출된 피처에 의미 있는 값을 부여하는가 하는 것이 **매우 중요한 요소**이다.

피처 벡터화 또는 피처 추출은 **텍스트를 word 기반의 다수의 피처로 추출하고 이 피처에 단어 빈도수와 같은 숫자 값을 부여하면 텍스트는 단어의 조합인 벡터값으로 표현하는 것을 말한다.**

텍스트 분석 수행 프로세스

1. 텍스트 사전 준비작업 : 텍스트를 피처로 만들기 전에 미리 클렌징, 대/소문자 변경, 특수문자 삭제 등의 클렌징 작업, 단어 등의 토큰화 작업, 의미 없는 단어 제거 작업, 어근 추

출 등의 텍스트 정규화 작업을 수행하는 것을 통칭한다.

2. 피처 벡터화/추출 : 사전 준비 작업으로 가공된 텍스트에서 피처를 추출하고 여기에 벡터 값을 할당한다. 대표적인 방법은 BOW와 Word2Vec이 있으며, BOW는 대표적으로 Count 기반과 TF-IDF 기반 벡터화가 있다.
3. ML 모델 수립 및 학습/예측/평가 : 피처 벡터화된 데이트 세트에 ML 모델을 적용해 학습/예측 및 평가를 수행한다.

02 텍스트 사전 준비 작업(텍스트 전처리) - 텍스트 정규화

텍스트 자체를 바로 피처로 만들 수는 없다. 사전에 텍스트를 가공하는 준비 작업이 필요하다.

텍스트 정규화 작업은 다음과 같이 분류할 수 있다.

- 클렌징
- 토큰화
- 필터링/스톱 워드 제거/ 철자 수정
- Stemming
- Lemmatization

클렌징

텍스트에서 분석에 오히려 방해가 되는 불필요한 문자, 기호 등을 사전에 제거하는 작업이다. 예를 들어 HTML, XML 태그나 특정 기호 등을 사전에 제거한다.

텍스트 토큰화

토큰화의 유형은 문서에서 문장을 분리하는 문장 토큰화와 문장에서 단어를 토큰으로 분리하는 단어 토큰화로 나눌 수 있다.

문장 토큰화

문장 토큰화는 문장의 마침표, 개행문자 등 문장의 마지막을 뜻하는 기호에 따라 분리하는 것이 일반적이다.

```
from nltk import sent_tokenize
import nltk
nltk.download('punkt')

text_sample = 'The Matrix is everywhere its all around us, here even in this room. ₩
               You can see it out your window or on your television. ₩
               You feel it when you go to work, or go to church or pay your taxes.'
sentences = sent_tokenize(text=text_sample)
print(type(sentences),len(sentences))
print(sentences)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:₩Users₩leech₩AppData₩Roaming₩nltk_data...
```

```
<class 'list'> 3
['The Matrix is everywhere its all around us, here even in this room.', 'You can see it out your window or on your television.', 'You feel
it when you go to work, or go to church or pay your taxes.']
```

```
[nltk_data]   Unzipping tokenizers₩punkt.zip.
```

`sent_tokenize()` 가 반환하는 것은 각각의 문장으로 구성된 `list` 객체이다. 반환된 `list` 객체가 3개의 문장으로 된 문자열을 가지고 있는 것을 알 수 있다.

단어 토큰화

단어 토큰화는 문장을 단어로 토큰화하는 것이다. 기본적으로 공백, 콤마, 마침표, 개행문자 등으로 단어를 분리하지만, 정규 표현식을 이용해 다양한 유형으로 토큰화를 수행할 수 있다.

```
from nltk import word_tokenize

sentence = "The Matrix is everywhere its all around us, here even in this room."
words = word_tokenize(sentence)
print(type(words), len(words))
print(words)
```

```
<class 'list'> 15
['The', 'Matrix', 'is', 'everywhere', 'its', 'all', 'around', 'us', ',', 'here', 'even', 'in', 'this', 'room', '.']
```

NLTK에서 기본으로 제공하는 `word_tokenize()` 를 이용해 단어로 토큰화할 수 있다.

`sent_tokenize` 와 `word_tokenize` 를 조합해 문서에 대해서 모든 단어를 토큰화해보자.

```
from nltk import word_tokenize, sent_tokenize

#여러개의 문장으로 된 입력 데이터를 문장별로 단어 토큰화 만드는 함수 생성
def tokenize_text(text):

    # 문장별로 분리 토큰
    sentences = sent_tokenize(text)
    # 분리된 문장별 단어 토큰화
    word_tokens = [word_tokenize(sentence) for sentence in sentences]
    return word_tokens

#여러 문장들에 대해 문장별 단어 토큰화 수행.
word_tokens = tokenize_text(text_sample)
print(type(word_tokens),len(word_tokens))
print(word_tokens)
```

```
<class 'list'> 3
[['The', 'Matrix', 'is', 'everywhere', 'its', 'all', 'around', 'us', ',', 'here', 'even', 'in', 'this', 'room', '.'], ['You', 'can', 'se
e', 'it', 'out', 'your', 'window', 'or', 'on', 'your', 'television', '.'], ['You', 'feel', 'it', 'when', 'you', 'go', 'to', 'work', ',',
'or', 'go', 'to', 'church', 'or', 'pay', 'your', 'taxes', '.']]
```

문장을 단어별로 하나씩 토큰화 할 경우 문맥적인 의미는 무시될 수 밖에 없다.

이를 위해 n-gram을 도입해보자. n-gram은 연속된 n개의 단어를 하나의 토큰화 단위로 분리해 내는 것이다.

스톱 워드 제거

스톱 워드는 분석에 큰 의미가 없는 단어를 지칭한다.

```
import nltk

stopwords = nltk.corpus.stopwords.words('english')
all_tokens = []
for sentence in word_tokens:
    filtered_words = []
    for word in sentence:
        word = word.lower()
        if word not in stopwords:
            filtered_words.append(word)
    all_tokens.append(filtered_words)

print(all_tokens)
```

```
[['matrix', 'everywhere', 'around', 'us', ',', 'even', 'room', '.'], ['see', 'window', 'television', '.'], ['feel', 'go', 'work', ',', 'g
o', 'church', 'pay', 'taxes', '.']]
```