텍스트 데이터 다루기

텍스트 데이터

- 데이터 종류
  - 정량적인 연속형 특성 데이터
  - 고정된 목록에서 값이 정해지는 범주형 특성
  - 텍스트 데이터

- 텍스트
  - 주로 글자가 연결된 문자열로 표현
  - 일반적으로 전처리 필요

BOW (Bag of Words)

- 광범위하게 사용되는 간결하고 효과적인 머신러닝 텍스트 표현 기법

- 구조와 상관없이 단어의 출현 횟수만 계산
  - 장, 문장, 서식 등 입력 텍스트의 구조 대부분은 유실
  - 텍스트에 각 단어가 얼마나 많이 나타나는지 계산

- 단계
  - 토큰화 → 각 문서를 공백, 구두점 등을 기준으로 단어(토큰)로 분리
  - 어휘 사전 구축 → 모든 문서에 나타난 모든 단어를 모으고 번호 부여(알파벳 순서)
  - 인코딩 → 어휘 사전의 단어가 각 문서마다 몇 번 나타나는지 계산

- 출력은 각 문서에서 나타난 단어의 빈도수가 저장된 벡터
  - SciPy 희소 행렬로 저장 → toarray() 사용하면 밀집 배열 반환

BOW 구현

- CounterVectorizer 사용
  - 정규 표현식으로 토큰 추출 (기본 정규 표현식 → \b\w\w+\b)

- 의미 없는 단어 제거
  - min_df 매개변수로 최소 몇 개의 문서에서 나타나야 하는지 지정
  - 너무 빈번해서 유용하지 않은 단어 제거
    - » 지나치게 많이 나타나는 단어 제거 (max_df 매개변수)
    - » 언어별 불용어 목록 사용

- tf-idf (term frequency – inverse document frequency)
  - 얼마나 의미 있는 특성인지를 계산해서 스케일 조정
  - 특정 문서에서 자주 나타나는 단어에 높은 가중치 부여
  - TfidfTransformer, TfidfVectorizer 사용

n-gram (여러 단어로 만든 BOW)

- BOW는 단어의 순서는 고려하지 않기 때문에 문맥에 따른 의미 차이를 반영하지 못함
  - (사례) it’s bad, not good at all == it’s good, not bad at all

- 연속된 2개 이상의 토큰을 함께 고려해서 문맥 반영
  - unigram(1개), bigram(2개), trigram(3개), n-gram
  - CouterVectorizer와 TfidfVectorizer는 ngram_range 매개변수로 지정

어간 추출, 표제어 추출

- Replace, replaced, replacement, replaces, replacing 같은 단어는 하나의 토큰으로 추출하는 것이 일반화에 유리

- 각 단어를 그 단어의 어간으로 표현해서 같은 어간을 가진 모든 단어를 구분 또는 병합

- 방법
  - 어간 추출 → 일일이 어미를 찾아 제외하는 규칙 기반 방식
  - 표제어 추출 → 알려진 단어의 형태 사전을 사용해서 처리하는 방식
  - 일반적으로 표제어 추출이 복잡하지만 좋은 성능 발휘

- 마지막 일부 성능까지 높여야 하는 상황에서 시도할만한 기법