Data Science

Text Data Analysis

노기섭 교수

(kafa46@hongik.ac.kr)

Lecture Goals

- **텍스트 마이닝 흐름과 비정형 데이터 특성 이해**
  - 수집 → 전처리 → 분석 → 시각화
- **영어·한국어 전처리기법**
  - 토큰화, 정규화, 불용어·어간/표제어 처리
- **단어 빈도 분석과 시각화**
- **감성 분석 수행과 결과 해석**
- **워드 클라우드 등 시각화 기법 적용**

텍스트 데이터란?

텍스트 데이터와 텍스트 마이닝

■ **현대 사회에서 우리가 남기는 데이터의 상당수는 텍스트 형태로 존재**

- 예: 뉴스, SNS, 리뷰, 이메일, 논문 등

■ **이러한 데이터는 비정형 데이터(Unstructured Data)**

- 숫자·표 형태의 구조적 데이터(Structured Data)와 달리 일정한 형식이 없음

■ **텍스트 마이닝(Text Mining)**

- 텍스트 데이터를 분석해 의미 있는 정보를 추출하는 기술
- 자연어처리(NLP) 기술과 밀접하게 연결되어 있음

텍스트 분석 단계

단계	설명
수집 (Collection)	뉴스, 블로그, SNS 등에서 텍스트 데이터를 수집
전처리 (Preprocessing)	불필요한 기호나 단어를 제거하고 정제
분석 (Analysis)	단어 빈도, 감성(긍정/부정), 주제 등을 분석
시각화 (Visualization)	워드 클라우드, 그래프 등을 통해 결과 표현

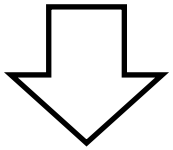
텍스트 전처리

토큰화 (Tokenization)

- **토큰화는 문장을 단어나 의미 있는 단위로 나누는 과정**

- **예:**

"데이터 사이언스는 흥미로운 학문이다"



[데이터, 사이언스는, 흥미로운, 학문이다]

NLTK 소개

NLTK (Natural Language Tool Kit)

- 파이썬에서 가장 널리 사용되는 자연어처리(NLP) 라이브러리 중 하나
- 영어 텍스트 분석에 강력한 도구로, 교육용 및 연구용으로 매우 자주 활용
- 한국어 텍스트의 경우에는 KoNLPy와 같은 별도의 형태소 분석기 사용이 필요

기능	설명
토큰화(Tokenization)	문장을 단어, 문장 등으로 분리
불용어 제거(Stopword Removal)	의미 없는 단어 제거
표제어 추출(Lemmatization)	단어의 기본형(표제어) 변환
감성 분석(Sentiment Analysis)	문장의 긍·부정 감정 분석
말뭉치(Corpus) 제공	영어 텍스트 샘플 및 사전 데이터 포함

NLTK 실습

```
# 의존성 패키지 설치
pip install nltk
```

```
import nltk

# 최초 한 번만 실행하면 되는 부분 입니다.
#   NLTK 데이터 다운로드
#   이후 실행 시 download()는 불필요합니다.

# nltk.download('punkt')        # 토큰화용 데이터
# nltk.download('punkt_tab')    # 탭 구분 토큰화용 데이터
# nltk.download('stopwords')    # 불용어 리스트
# nltk.download('wordnet')      # 표제어 추출용 데이터

text = "Data Science is an exciting field."
tokens = nltk.word_tokenize(text)

print(tokens)
```

```
# 실행 결과
['Data', 'Science', 'is', 'an', 'exciting', 'field', '.']
```

한국어 토큰화

한국어 특징

- 조사, 어미, 복합명사가 많기 때문에 단순한 공백 기준으로는 단어를 나누기 어려움
- 한국어 문장을 분석할 때는 전용 형태소 분석기를 사용
- 표적인 라이브러리는 KoNLPy 패키지이며, 내부에 여러 분석기(Okt, Komoran, Mecab 등)를 포함
- 다양한 한국어 분석기

  https://www.deepshark.org/courses/data_science/w/09_text_data_analysis#ko-tokenization

Konlpy 실습

```
# 의존성 패키지 설치
pip install konlpy
```

```
from konlpy.tag import Okt

okt = Okt()

text = "데이터 사이언스는 정말 흥미로운 학문이다."

# 단어 단위(토큰)로 분리
tokens = okt.morphs(text)
print("토큰:", tokens)

# 품사 태깅(Pos tagging)
pos_tags = okt.pos(text)
print("품사 태깅:", pos_tags)

nouns = okt.nouns(text) # 명사만 추출
print("명사:", nouns)
```

Okt(Open Korean Text) 소개

한국어 문장을 형태소 단위로 나누고,각 단어가 어떤 품사인지 알려주는 형태소 분석기

원래 이름은 트위터 형태소 분석기(Twitter) 나중에 Okt로 이름 변경

한국어를 다루는 자연어처리(NLP)에서 아주 사용

실행 결과
토큰: ['데이터', '사이언스', '는', '정말', '흥미로운', '학문', '이다', '.']

품사 태깅: [('데이터', 'Noun'), ('사이언스', 'Noun'), ('는', 'Josa'), ('정말', 'Adverb'), ('흥미로운', 'Adjective'), ('학문', 'Noun'), ('이다', 'Josa'), ('.', 'Punctuation')]

명사: ['데이터', '사이언스', '학문']

정규화 (Normalization)

정규화

- 텍스트 데이터를 일관된 형태로 바꾸는 과정
- 같은 의미 단어라도 표현 방식이 다르면

  컴퓨터는 서로 다른 단어로 인식

  ➔ 정규화를 통해 이런 차이를 최소화해야 함
- 정규화의 목표:
  - "**사람이 보기엔 같지만, 컴퓨터는 다르게 보는**" 단어들을 하나의 형태로 통일

    ➔ 단어 중복이 줄고, 분석 정확도를 크게 향상시킬 수 있음.

문자	유니코드	10진수
D	U+0044	68
a	U+0061	97
t	U+0074	116
a	U+0061	97
d	U+0064	100
a	U+0061	97
t	U+0074	116
a	U+0061	97

D a t a
68 97 116 97

d a t a
100 97 116 97

"Data", "data.", "DATA!", "data?"
이 네 단어는 사람이 보기엔 모두 'data'를 의미하지만,
컴퓨터는 각각 다른 단어로 인식/처리한다.

정규화 변환 방법

단계	설명	예시
소문자 변환 (Lowercasing)	대소문자 구분을 없앰 동일 단어로 인식하게 함	"Data", "DATA" → "data"
특수문자 제거	문장부호나 불필요한 기호 제거	"data-science!!" → "data science"
공백 정리	여러 개의 공백을 하나로 줄임	"data science" → "data science"
숫자/이모티콘 제거 (필요 시)	분석 목적에 따라 숫자·이모티콘을 제거하거나 유지	"2024년 👍 데이터" → "2024년 데이터"

re 모듈

re 모듈

- 텍스트 데이터를 다룰 때 매우 강력한 도구

- 단순히 "기호를 지운다" 수준을 넘어 데이터를 깨끗하고 분석하기 좋은 형태로 만드는 정제 도구

구분	설명	링크
공식 문서	Python 표준 라이브러리의 re 모듈 설명. 모든 함수(match, search, sub, findall 등)의 동작과 인자 설명이 포함됨.	Python 공식 문서: re — Regular expression operations
정규표현식 HOW-TO	초보자를 위한 Python 공식 "정규표현식 입문 가이드". 메타문자, 그룹, 수량자 등 정규식의 원리를 체계적으로 설명함.	Regular Expression HOWTO — Python Docs
Real Python 튜토리얼	실습 중심의 예제로 re 모듈 사용법을 설명. 정규표현식을 단계별로 익히기에 적합함.	Regular Expressions in Python: the re Module
블로그	re.match, re.search, re.findall 등의 차이점을 한눈에 비교. 한국어로 정리된 예제 중심 설명.	Python 정규표현식 정리 (note.nkmk.me)

re 간단 실습

```
import re  # re는 Regular Expression(정규표현식) 모듈이다.

text = "Data-Science!! is, Exciting???  "

# 모든 문자를 소문자로 변환
text = text.lower()

# 알파벳과 공백만 남기기
# [^a-z\\s] → 알파벳(a~z)과 공백(\\s)을 제외한 나머지를 모두 찾아 공백으로 치환
text = re.sub(r'[^a-z\s]', ' ', text)

# 여러 개의 공백을 하나로 줄이고, 양쪽 공백 제거
# \\s+ → 공백 문자(스페이스, 탭, 줄바꿈 등)가 하나 이상 연속될 때
text = re.sub(r'\s+', ' ', text).strip()

print(text)
```

```
# 실행 결과
datascience is exciting
```

불용어 제거

불용어(stopwords)

- 텍스트에 자주 등장하지만 분석 목적에 큰 기여를 하지 않는 단어
- "의미가 없다"기보다는, 구분력을 떨어뜨리거나 노이즈가 되는 경향이 있어서 제거함
  - 영어의 the, is, and, 한국어의 조사/어미 은, 는, 이, 가, 을, 를, 에, 와, 과 등이 해당
- **불용어는 절대적 목록이 아니라 '과제·도메인 의존적'**

제거하면 안되는 불용어

- 부정어/부사: not, no, never, 없다, 않다, 못하다 같은 단어는 감성 분석에서 결정적 의미를 가짐
- 지시어나 의문사: 검색, QA 시스템에서는 what, where, how / 이것, 저것, 여기 단어가 핵심 역할
- 짧은 문장: SNS나 댓글처럼 문장이 짧을 때는 불용어를 제거하면 전체 의미가 사라짐.

불용어 설계 가이드

단계	설명
1. 기본 리스트 사용	영어: nltk.corpus.stopwords.words('english') 한국어: KoNLPy 형태소 분석 후 조사(Josa), 어미(Eomi) 제거
2. 화이트리스트 정의 (보존할 텍스트)	부정어·강조어(예: not, 없다, 않다)는 제거하지 않는다
3. 빈도 기반 제거	문서의 일정 비율(%) 이상 등장하는 단어(max_df), 너무 희귀한 단어(min_df)는 제거
4. 지속적 업데이트	분석 결과를 검토해 커스텀 불용어 목록을 보완

불용어 제거 실습

영어 불용어 제거 실습

- https://www.deepshark.org/courses/data_science/w/09_text_data_analysis#en_stopword

한국어 불용어 제거 실습

- https://www.deepshark.org/courses/data_science/w/09_text_data_analysis#ko_stopword

어간 (Stemming) 추출

어간 추출(Stemming)과 표제어 추출(Lemmatization)

- 단어의 다양한 변형 형태를 하나의 공통된 형태로 통일하는 과정

어간 (Stem)

- 단어의 기본 형태 중에서 변하지 않는 핵심 부분을 의미
- 동사의 시제, 명사의 복수형 등은 변할 수 있지만, 어간은 단어의 뿌리 역할
- 단어 변화의 공통된 부분으로 의미를 유지하는 최소 단위

원형 단어	변형된 형태	공통 어간
play	playing, played, plays	**play**
study	studying, studied, studies	**studi**
move	moving, moved, moves	**mov**

표제어 (Lemmatization) 추출

표제어 (Lemma)

- 표제어는 사전(dictionary)에 등록된 단어의 기본형이다.
- 단어의 품사(POS)와 문법 규칙을 고려하여 의미적으로 올바른 기본 형태로 변환
- 예를 들어, "better"는 어간 추출로는 그대로 남지만, 표제어 추출에서는 "good"으로 변경

단어	표제어(Lemma)	품사
running	run	동사(verb)
studies	study	동사(verb)
better	good	형용사(adj)
was	be	동사(verb)

어간 vs. 표제어 추출

구분	어간 추출 (Stemming)	표제어 추출 (Lemmatization)
정의	단순히 단어의 어미를 자르고 통일	사전과 품사 정보를 이용해 의미적으로 정확한 기본형 변환
예시	studying → studi	studying → study
결과 형태	실제 단어가 아닐 수 있다	실제 존재하는 단어
처리 속도	빠르다	느리지만 정확하다
사용 목적	단순 검색, 키워드 중심 분석	문법적·의미적 분석이 필요한 작업

어간 및 표제어 추출 실습

```python
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import wordnet
import nltk

nltk.download('wordnet')

# 어간 추출 객체 생성 (Stemming)
stemmer = PorterStemmer()

# 표제어 추출 객체 생성 (Lemmatization)
lemmatizer = WordNetLemmatizer()

words = ["studies", "studying", "studied", "better", "running"]

print("단어\t어간추출\t표제어추출")
for w in words:
    stem = stemmer.stem(w)
    # 동사 기준으로 변환
    lemma = lemmatizer.lemmatize(w, pos=wordnet.VERB)
    print(f"{w}\t{stem}\t{lemma}")
```

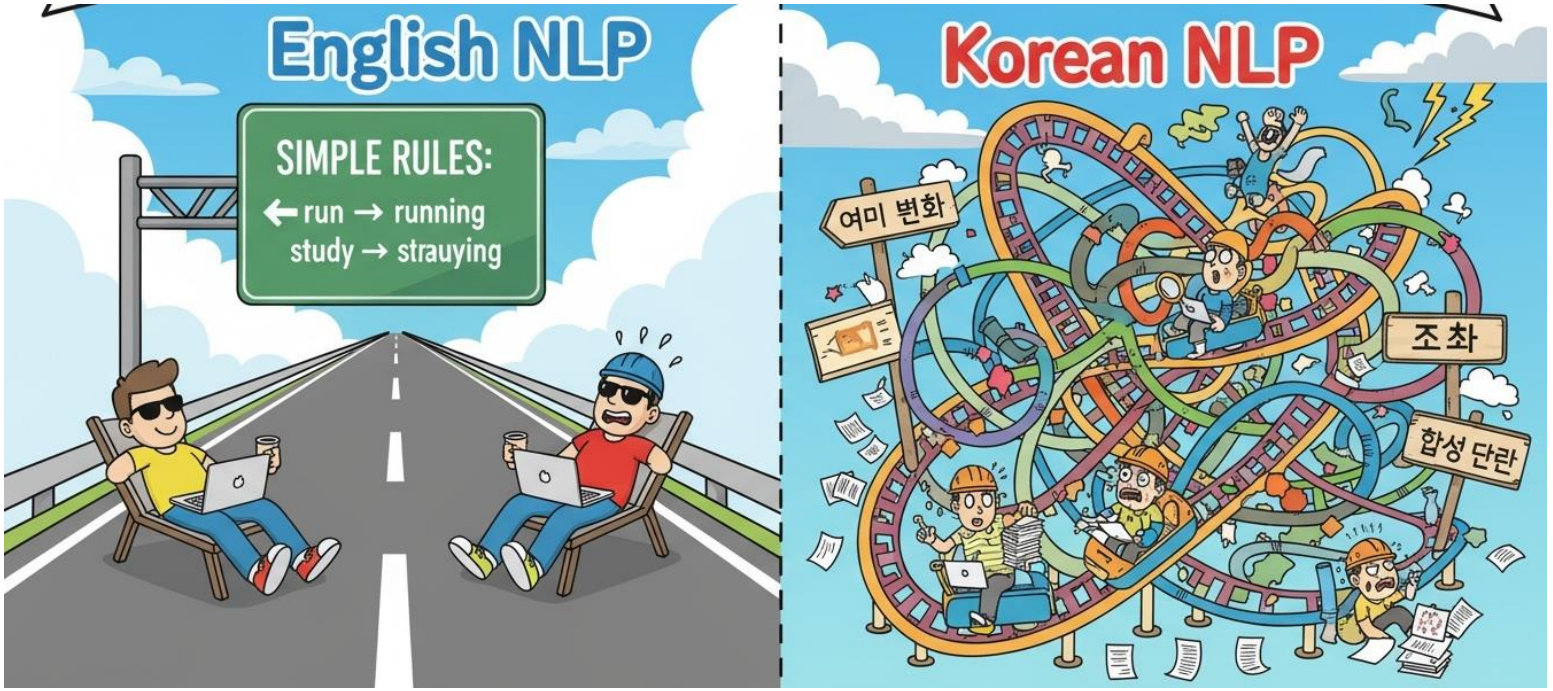
```
# 실행 결과
단어        어간추출    표제어추출
studies     studi       study
studying    studi       study
studied     studi       study
better      better      good
running     runn        run
```

한국어에서 어간 및 표제어 추출을 하지 않는 이유 (1/3)

■ 형태가 복잡하고, 규칙이 단순하지 않다!

- 영어는 run → running, study → studying처럼 접사 규칙이 단순하지만,
- 한국어는 어미 변화와 조사 결합이 매우 다양

원형	변형된 형태
가다	갑니다, 가요, 가고, 갔지만, 가니까, 가려면
하다	합니다, 해요, 했지만, 하는, 하려고, 하겠다고



한국어에서 어간 및 표제어 추출을 하지 않는 이유 (2/3)

■ **대신 "형태소 분석(Morphological Analysis)"을 사용한다!**

- 한국어에서는 단순히 어미를 자르는 것보다, 문장을 형태소 단위로 나누고
- 각 형태소의 품사(POS)를 분석하는 접근이 훨씬 정확함

```
# Target Sentence

   "가고 싶지만 가지 못했다"

# 품사 태깅 (POS) 결과

   [('가', 'Verb'), ('고', 'Eomi'), ('싶', 'Verb'),  ('지만', 'Eomi'),

    ('가', 'Verb'), ('지', 'Eomi'),  ('못하', 'Verb'), ('었', 'Eomi'), ('다', 'Eomi')]
```

한국어에서 어간 및 표제어 추출을 하지 않는 이유 (3/3)

KoNLPy 분석기들이 이미 “어간화” 옵션 제공

- Okt.pos(text, stem=True) 옵션을 쓰면 자동으로
  - `필요하다 → 필요`,
  - `먹었다 → 먹다`,
  - `한다 → 하다`로 변환
- 영어의 Lemmatizer처럼 별도의 단계가 아니라,

  형태소 분석 단계에서 이미 표제어(lemma) 처리가 함께 수행

실제 연구·실무에서는 형태소 기반 처리로 충분

- 한국어 NLP에서는 보통 다음 단계로 진행된다

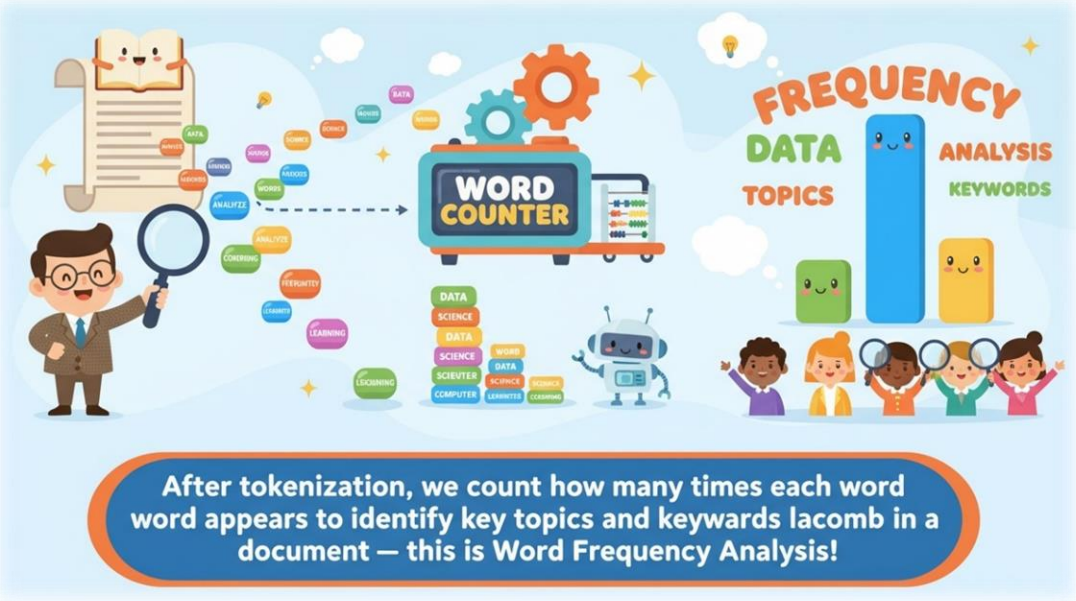
1. 형태소 분석 (Morphological Analysis) → 품사 정보까지 포함한 토큰화
2. 불용어 제거 + 품사 선택 (예: 명사, 동사만 사용)
3. 임베딩/모델 입력 (Word2Vec, BERT 등)

단어 빈도 분석

단어 빈도 분석

단어 빈도 분석(word frequency analysis)

- 문서 내에서 어떤 단어가 자주 등장하는지를 계산하여 주제나 핵심 키워드를 빠르게 파악하는 과정



- 예를 들어, 뉴스 기사나 대통령 연설문을 분석한다면,
  - '경제', '국민', '정책', '안전' 같은 단어가 자주 등장할 수 있음.
- 자주 등장하는 단어는 해당 문서의 핵심 관심사가 무엇인지 파악하는데 도움이 됨.

단어 빈도 분석

대통령 연설문 단어 빈도 분석 실습

- 대통령기록관: https://www.pa.go.kr/research/contents/speech/index.jsp
- 샘플 데이터 다운로드: 제103주년 3·1절 기념사
- 실습 코드:

  https://www.deepshark.org/courses/data_science/w/09_text_data_analysis#word_frequency_president_speech

감성 분석

감성 분석

감성 분석 (Sentiment Analysis)

- 텍스트(문장, 리뷰, 트윗 등)에 담긴 사람의 감정이나 의견을 자동으로 파악하는 기술
- 문장을 읽고 긍정인지(Positive), 부정인지(Negative), 혹은 중립인지(Neutral) 를 구분하는 작업
    - "I really love this movie. It was fantastic!" → 긍정적 감정 (Positive)
    - "This movie was boring and too long." → 부정적 감정 (Negative)

텍스트 데이터 분석에서의 역할

- 의견 요약 및 분류
    - 수많은 고객 리뷰나 SNS 댓글을 자동으로 요약하거나 긍·부정 비율로 시각화
- 의사결정 지원
    - 마케팅, 정치, 주식 시장 등 다양한 분야에서 대중의 정서를 수치화하여 데이터 기반 의사결정
- 자연어 처리(NLP) 파이프라인의 핵심 단계

TextBlob을 이용한 간단한 감성분석

TextBlob

- 텍스트 데이터를 처리하기 위한 파이썬 라이브러리이다.
- 자연어 처리(NLP)에서 자주 수행되는 작업들을 간단하게 수행할 수 있도록 하는 직관적인 API를 제공

주요 기능

- 품사 태깅(Part-of-Speech Tagging)
- 명사구 추출(Noun Phrase Extraction)
- 감성 분석(Sentiment Analysis)
- 문서 분류(Classification)
- 번역(Translation) 등

실습 코드

- https://www.deepshark.org/courses/data_science/w/09_text_data_analysis#textblob

딥러닝 모델을 이용한 감성 분석

BERT 소개

- BERT (Bidirectional Encoder Representations from Transformers)
- Google이 2018년에 발표한 언어 모델
- 문맥을 양방향(앞뒤 문장 모두)으로 이해하는 Transformer 기반 구조를 사용
- 문장의 의미나 어휘 간 관계를 깊이 파악

Roberta 소개

- Robustly Optimized BERT Approach
- 2019년 Facebook AI(현재 Meta AI)가 발표한 BERT 모델의 개선 버전
- Roberta 실습 코드

https://www.deepshark.org/courses/data_science/w/09_text_data_analysis#roberta

주의:
GPU가 없을 경우
오래 걸릴 수 있음
ㅠㅠ

워드 클라우드 시각화

워드클라우드 시각화 (1/2)

워드클라우드 시각화 (Word Cloud)

- 텍스트 데이터에서 단어의 등장 빈도를 시각적으로 표현하는 방법
- 단어의 크기, 굵기, 색상 등을 이용하여 등장 빈도가 높은 단어일수록 강조되게 표현
- 긴 문서의 핵심 주제나 주요 키워드를 한눈에 파악

워드클라우드를 사용하는 이유

- 요약 효과: 대량의 텍스트를 읽지 않아도 주요 단어를 직관적으로 확인
- 탐색적 분석(EDA): 본격적인 분석 이전에 데이터의 전반적 특징을 파악하는 데 유용
- 커뮤니케이션: 분석 결과를 시각적으로 전달하여 비전문가도 쉽게 이해
- 전처리 점검: 불용어 제거와 형태소 분석이 잘 되었는지 시각적으로 확인

워드클라우드 시각화 (2/2)

장점과 한계

- 시각적으로 아름답고 직관적이지만,
- 문맥이나 부정 표현(예: "좋지 않다")을 반영하기 어렵움.
- 정밀 분석보다는 탐색적 시각화나 발표용 요약에 적합

사용 도구

- Python: wordcloud, matplotlib, konlpy를 조합하여 사용
- R: wordcloud, tm, tidytext 패키지
- 시각화 툴: Tableau, Power BI, Flourish 등
- 웹 도구: Voyant Tools, D3.js, Echarts

한국어 분석 시 주의사항

- 한국어는 조사가 많고 어미 변형이 다양하기 때문에 형태소 분석을 반드시 수행
- 형태소 분석기를 이용하며, 의미 없는 조사나 접속사는 불용어로 제거

워드클라우드 시각화 실습

- **wordcloud 패키지 설치 (pip install wordcloud)**
- **운영체제별 한글 폰트 위치**
  - Windows : C:/Windows/Fonts/malgun.ttf (맑은 고딕)
  - macOS : /Library/Fonts/AppleGothic.ttf (애플고딕)
  - Linux/Colab : /usr/share/fonts/truetype/nanum/NanumGothic.ttf (나눔고딕)

실습 코드:
https://www.deepshark.org/courses/data_science/w/09_text_data_analysis#wordcloud

폰트 이름	라이선스	특징	다운로드
나눔고딕 (Nanum Gothic)	무료 (NAVER OFL)	깔끔하고 가독성이 좋아 워드클라우드나 웹 프로젝트에 자주 사용	NAVER
카페24 써라운드 (Cafe24 Surround)	무료 상업적 사용 가능	귀엽고 부드러운 곡선체 감성적인 디자인에 적합	카페24
배민 도현체 (BME DoHyeon)	무료 상업적 사용 가능	굵고 임팩트 있는 제목용 폰트	배민

수고하셨습니다 ..^^..