텍스트 분석 기법 (Text Analysis Techniques)

- 전처리 및 벡터화된 텍스트 데이터를 활용하여 특정 목적에 맞는 정보를 추출하거나 패턴을 발견하는 방법론
- 주로 감성 분석과 토픽 모델링이 대표적이며, 텍스트 데이터에서 감성 정보나 숨겨진 주제를 추출하여 비즈니스 의사결정이나 추가 분석에 활용

감성 분석 (Sentiment Analysis)

- **개념**: 텍스트에 담긴 주관적인 의견, 감정, 태도 등을 분석하여 긍정, 부정, 중립과 같은 감성 극성(sentiment polarity)을 판단하는 기술입니다.
- **적용**: 고객 리뷰 분석, 소셜 미디어 여론 분석, 브랜드 이미지 모니터링, 시장 동향 분석, 제품 피드백 분석 등 다양한 분야에서 활용됩니다.
- **방법**:
  - **어휘 기반 (Lexicon-based)**: 긍정/부정 단어 사전(lexicon)을 구축하고, 텍스트 내 단어들의 감성 점수를 합산하여 감성을 판단합니다.
  - **머신러닝 기반 (Machine Learning-based)**: 감성이 레이블링된 텍스트 데이터를 사용하여 분류 모델(로지스틱 회귀, SVM, 나이브 베이즈 등)을 학습시킵니다.
  - **딥러닝 기반 (Deep Learning-based)**: RNN, LSTM, BERT 등 딥러닝 모델을 사용하여 텍스트의 복잡한 문맥적 감성을 학습합니다.
- **주의 사항**:
  - **문맥 고려**: 어휘 기반 감성 분석은 단어의 문맥적 의미를 파악하기 어렵습니다. (e.g., "이 영화는 재미없지 않다" -> 긍정)
  - **도메인 특화**: 일반적인 감성 사전은 특정 도메인(e.g., 의료, 금융)의 감성을 정확히 반영하지 못할 수 있습니다.
- **구현 방법**: `NLTK`의 `VaderSentimentIntensityAnalyzer` (어휘 기반), `scikit-learn`의 분류 모델 (머신러닝 기반).

감성 분석 (머신러닝 기반)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# 1. 데이터 준비 (예시: 간단한 감성 데이터)
# 실제 데이터는 더 많은 양과 다양한 감성 표현을 포함해야 함
texts = [
    "이 영화 정말 최고다! 강력 추천합니다.", # 긍정
    "시간 낭비였어요. 너무 지루하고 재미없네요.", # 부정
    "그냥 볼만했어요. 특별히 좋지도 나쁘지도 않아요.", # 중립
    "인생 영화 등극! 다시 보고 싶어요.", # 긍정
    "최악의 경험. 돈 아까워요.", # 부정
    "나쁘지 않은데, 기대했던 것보다는 별로였어요.", # 부정 (중립에 가까움)
    "딱 평범하게 볼만한 것 같아요." # 중립
]
```

```
labels = [1, 0, 2, 1, 0, 0, 2] # 1: 긍정, 0: 부정, 2: 중립

df_sentiment = pd.DataFrame({'text': texts, 'sentiment': labels})

# 2. 텍스트 전처리 (간단화) 및 벡터화
# 실제로는 토큰화, 불용어 제거 등 더 복잡한 전처리 필요
vectorizer = TfidfVectorizer(stop_words='english', max_features=100)
X_vec = vectorizer.fit_transform(df_sentiment['text'])
y_sent = df_sentiment['sentiment']

X_train_sent, X_test_sent, y_train_sent, y_test_sent = train_test_split(X_vec,
y_sent, test_size=0.3, random_state=42, stratify=y_sent)

# 3. 모델 학습 (로지스틱 회귀)
sentiment_model = LogisticRegression(random_state=42, max_iter=1000)
sentiment_model.fit(X_train_sent, y_train_sent)

# 4. 예측 및 평가
y_pred_sent = sentiment_model.predict(X_test_sent)
print("--- 감성 분석 모델 평가 ---")
print(f"정확도: {accuracy_score(y_test_sent, y_pred_sent):.3f}")
print("분류 리포트:\n", classification_report(y_test_sent, y_pred_sent))
'''
정확도: 0.333
분류 리포트:
              precision    recall  f1-score   support

           0       0.33      1.00      0.50         1
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         1

    accuracy                           0.33         3
   macro avg       0.11      0.33      0.17         3
weighted avg       0.11      0.33      0.17         3
'''
```

- **결과 해석**: `classification_report`를 통해 각 감성 클래스(긍정, 부정, 중립)에 대한 정밀도, 재현율, F1-점수 등을 확인하여 모델의 성능을 평가합니다.
- **장단점 및 대안**:
  - **장점**: 텍스트 데이터에서 주관적인 정보를 추출하여 비즈니스 의사결정에 활용할 수 있습니다.
  - **단점**: 문맥, 비유, 반어법 등을 이해하기 어렵고, 도메인 특화된 감성 분석이 필요할 수 있습니다.
  - **대안**: BERT, GPT 등 사전 학습된 언어 모델을 활용한 딥러닝 기반 감성 분석은 문맥 이해 능력이 뛰어나 더 높은 성능을 보입니다.

토픽 모델링 (Topic Modeling)

- **개념**: 대규모 텍스트 문서 집합에서 숨겨진 주제(Topic)를 발견하는 비지도 학습 기법입니다. 각 문서는 여러 주제의 혼합으로 구성되고, 각 주제는 특정 단어들의 확률 분포로 표현된다고 가정합니다.
- **적용**: 뉴스 기사 분류, 논문 주제 분석, 고객 문의 유형 분류 등 문서 집합의 구조를 이해하고 요약하는 데 사용됩니다.
  - 대량의 텍스트 데이터에서 핵심 주제를 자동으로 추출하고 분류, 문서 요약, 정보 검색.

- **대표 알고리즘**: **LDA (Latent Dirichlet Allocation, 잠재 디리클레 할당)**
  - **LDA 개념**: 각 문서가 특정 토픽들의 혼합으로 이루어져 있고, 각 토픽은 특정 단어들의 혼합으로 이루어져 있다는 가정 하에, 문서-토픽 분포와 토픽-단어 분포를 추론하는 확률적 생성 모델입니다.
  - **장점**: 문서 집합의 숨겨진 의미 구조를 발견하고, 각 문서가 어떤 주제를 다루는지 파악할 수 있습니다.
  - **단점**: 토픽의 개수(K)를 사전에 지정해야 하며, 토픽의 의미를 사람이 직접 해석해야 합니다.
- **주의 사항**:
  - **토픽 개수(K) 설정**: 최적의 토픽 개수를 찾는 것이 중요합니다. 일관성 점수(Coherence Score) 등을 활용하여 평가합니다.
  - **전처리 중요성**: 토픽 모델링의 결과는 텍스트 전처리(불용어 제거, 표제어 추출 등)의 품질에 크게 의존합니다.
- **구현 방법**: `gensim` 라이브러리 또는 `scikit-learn`의 `LatentDirichletAllocation`.
  - `pyLDAvis`의 경우 현재 환경에서 충돌로 인해 설치 불가 → 실습 미진행

토픽 모델링 (LDA)

```
# !pip install pyLDAvis
import gensim
from gensim.corpora import Dictionary
from gensim.models import LdaModel
from gensim.models.coherencemodel import CoherenceModel
# import pyLDAvis.gensim_models as gensimvis
# import pyLDAvis
import re

# 1. 텍스트 데이터 준비 (전처리된 텍스트)
# 실제로는 텍스트 전처리(토큰화, 불용어 제거 등)가 선행되어야 함
documents_lda = [
    "machine learning is a field of artificial intelligence",
    "deep learning is a subset of machine learning",
    "natural language processing uses machine learning techniques",
    "computer vision is another field of artificial intelligence",
    "data science combines statistics and machine learning",
    "neural networks are used in deep learning"
]

# 각 문서를 단어 리스트로 토큰화 (간단한 예시)
tokenized_docs = [doc.split() for doc in documents_lda]

# 2. 단어 사전 (Dictionary) 생성
dictionary = Dictionary(tokenized_docs)

# 3. 코퍼스 (Corpus) 생성 (Bag of Words 형태로 변환)
corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]

# 4. LDA 모델 학습
# num_topics: 토픽의 개수 (하이퍼파라미터)
# id2word: 단어 사전
# passes: 훈련 반복 횟수
# random_state: 재현성을 위한 시드
```

```
lda_model = LdaModel(corpus=corpus, num_topics=2, id2word=dictionary, passes=10,
random_state=42)

# 5. 토픽 확인
print("--- LDA 토픽 확인 ---")
for idx, topic in lda_model.print_topics(-1):
    print(f"Topic {idx}: {topic}")
'''
Topic 0: 0.100*"learning" + 0.078*"is" + 0.078*"of" + 0.056*"deep" + 0.056*"a" +
0.056*"intelligence" + 0.056*"field" + 0.055*"artificial" + 0.055*"machine" +
0.033*"are"
Topic 1: 0.091*"machine" + 0.090*"learning" + 0.053*"science" + 0.053*"combines" +
0.053*"techniques" + 0.053*"statistics" + 0.053*"uses" + 0.053*"and" +
0.053*"language" + 0.053*"data"
'''

# 6. 문서별 토픽 분포 확인
print("\n--- 문서별 토픽 분포 ---")
for i, doc in enumerate(corpus):
    print(f"Document {i}: {lda_model.get_document_topics(doc)}")
'''
Document 0: [(0, 0.9360359), (1, 0.06396413)]
Document 1: [(0, 0.93255156), (1, 0.067448504)]
Document 2: [(0, 0.0696858), (1, 0.9303142)]
Document 3: [(0, 0.9418144), (1, 0.05818556)]
Document 4: [(0, 0.06971005), (1, 0.9302899)]
Document 5: [(0, 0.9309207), (1, 0.06907922)]
'''

# 7. 토픽 일관성 (Coherence Score) 평가
# cv: 일관성 측정 방법
coherence_model_lda = CoherenceModel(model=lda_model, texts=tokenized_docs,
dictionary=dictionary, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print(f"\nCoherence Score: {coherence_lda:.3f}") # 0.367

# 8. 토픽 시각화 (pyLDAvis)
# pyLDAvis.enable_notebook() # Jupyter Notebook 환경에서 사용
# vis = gensimvis.prepare(lda_model, corpus, dictionary)
# pyLDAvis.display(vis) # 웹 브라우저에서 시각화
```

- **결과 해석**:
  - `print_topics()`: 각 토픽을 구성하는 주요 단어들과 그 단어들의 확률을 보여줍니다. 이를 통해 각 토픽이 어떤 의미를 가지는지 사람이 해석해야 합니다.
  - `get_document_topics()`: 각 문서가 어떤 토픽에 얼마나 기여하는지(토픽 분포)를 보여줍니다.
  - **Coherence Score**: 토픽의 의미론적 일관성을 측정하는 지표입니다. 점수가 높을수록 토픽이 더 의미 있고 해석하기 쉽다고 판단합니다.
- **장단점 및 대안**:
  - **장점**: 대규모 문서 집합에서 숨겨진 주제를 자동으로 발견하고, 문서의 구조를 이해하는 데 도움을 줍니다.
  - **단점**: 토픽의 개수를 사전에 지정해야 하고, 토픽의 의미를 사람이 직접 해석해야 합니다.

- **대안**: NMF(Non-negative Matrix Factorization), LSA(Latent Semantic Analysis) 등 다른 토픽 모델링 기법이나, BERTopic과 같이 딥러닝 임베딩을 활용한 토픽 모델링은 더 풍부한 토픽을 추출할 수 있습니다.

토픽 모델링 (Scikit-Learn 기반)

```
import re
import numpy as np
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import CountVectorizer

# 1. 텍스트 데이터 준비
documents_lda = [
    "machine learning is a field of artificial intelligence",
    "deep learning is a subset of machine learning",
    "natural language processing uses machine learning techniques",
    "computer vision is another field of artificial intelligence",
    "data science combines statistics and machine learning",
    "neural networks are used in deep learning"
]

# 2. 벡터화 (CountVectorizer)
# - TF-IDF가 아니라 단어 빈도를 써야 LDA에서 확률 해석이 가능
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(documents_lda)

# 3. LDA 모델 학습
# n_components: 토픽 개수
lda_model = LatentDirichletAllocation(
    n_components=2,
    random_state=42,
    learning_method='batch',
    max_iter=20
)
lda_model.fit(X)

# 4. 단어 사전 (피처 이름)
terms = vectorizer.get_feature_names()

# 5. 토픽별 주요 단어 출력
def print_topics(model, feature_names, n_top_words=10):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx}: ", end="")
        top_features = topic.argsort()[:-n_top_words - 1:-1]
        print(" + ".join([f"{feature_names[i]} ({topic[i]:.2f})" for i in
top_features]))

print("--- LDA 토픽 확인 ---")
print_topics(lda_model, terms, n_top_words=10)
'''
Topic 0: artificial (2.48) + field (2.48) + intelligence (2.48) + learning (1.95)
+ deep (1.50) + computer (1.50) + vision (1.50) + neural (1.49) + networks (1.49)
```

```
+ used (1.49)
Topic 1: learning (5.05) + machine (4.26) + natural (1.50) + techniques (1.50) +
processing (1.50) + uses (1.50) + language (1.50) + deep (1.50) + combines (1.50)
+ statistics (1.50)
'''

# 6. 문서별 토픽 분포 확인
doc_topic_distr = lda_model.transform(X)

print("\n--- 문서별 토픽 분포 ---")
for i, topic_probs in enumerate(doc_topic_distr):
    print(f"Document {i}: {[(j, round(p, 3)) for j, p in
enumerate(topic_probs)]}")
'''
Document 0: [(0, 0.698), (1, 0.302)]
Document 1: [(0, 0.1), (1, 0.9)]
Document 2: [(0, 0.066), (1, 0.934)]
Document 3: [(0, 0.915), (1, 0.085)]
Document 4: [(0, 0.076), (1, 0.924)]
Document 5: [(0, 0.883), (1, 0.117)]
'''

# 7. 토픽 일관성(Coherence) 유사 지표로 대체
# scikit-learn에는 gensim의 CoherenceModel이 없음.
# 대신 각 문서의 가장 높은 토픽 확률 평균을 간단한 "일관성" 근사값으로 사용 가능.
coherence_like = np.mean(np.max(doc_topic_distr, axis=1))
print(f"\nApprox. Coherence (mean dominant topic prob): {coherence_like:.3f}") #
0.876
```