텍스트 전처리 (Text Preprocessing)

- 텍스트 데이터를 머신러닝 모델이나 텍스트 분석 알고리즘이 이해하고 처리할 수 있는 형태로 변환하는 과정
- 원시 텍스트 데이터: 노이즈(불필요한 문자, 기호 등)가 많고, 컴퓨터가 직접 이해하기 어려운 비정형적인 형태
- 효과적인 텍스트 전처리는 분석의 정확도와 효율성을 크게 향상

1. **토큰화 (Tokenization)**:
    - 텍스트를 의미 있는 최소 단위인 '토큰(Token)'으로 분리하는 과정
    - 토큰: 단어, 구, 문장 등
    - **종류**:
        - **단어 토큰화 (Word Tokenization)**: 문장을 단어 단위로 분리
        - **문장 토큰화 (Sentence Tokenization)**: 문서를 문장 단위로 분리
    - **도구**: `NLTK`의 `word_tokenize`, `sent_tokenize`, `spaCy`, `KoNLPy` (한국어) 등.
2. **불용어 제거 (Stop Word Removal)**:
    - 텍스트 분석에 큰 의미가 없지만 자주 등장하는 단어들을 제거하는 과정
        - e.g. '은', '는', '이', '가', 'the', 'a', 'is'
    - 이러한 단어들은 분석의 노이즈로 작용할 가능성이 높음
    - **도구**: `NLTK`의 `stopwords`, 직접 정의한 불용어 리스트.
3. **형태소 분석 (Morphological Analysis)**:
    - 한국어와 같은 교착어(Agglutinative language)의 특징
        - 단어가 어근과 접사, 조사 등이 결합된 형태
    - 형태소 분석: 단어를 의미를 가지는 최소 단위인 '형태소'로 분리 후 각 형태소의 품사(명사, 동사, 형용사 등)를 태깅하는 과정
    - **도구**: `KoNLPy` (Okt, Kkma, Komoran, Hannanum 등).
4. **표제어 추출 (Lemmatization) / 어간 추출 (Stemming)**:
    - 단어의 형태가 다르더라도 같은 의미를 가지는 단어들을 하나의 대표 형태로 통일하는 과정
    - **표제어 추출 (Lemmatization)**:
        - 단어의 원형(표제어, lemma)을 찾는 과정
        - 단어의 품사 정보를 활용하여 정확한 원형을 찾으므로, 결과 단어가 사전에 존재하는 형태
        - e.g. 'am', 'are', 'is' -> 'be'
    - **어간 추출 (Stemming)**:
        - 단어의 어간(stem)을 찾는 과정
        - 규칙 기반으로 단어의 접미사를 제거하여 어간을 찾으므로, 결과 단어가 사전에 존재하지 않는 형태일 수 있음
        - e.g. 'running', 'runs', 'ran' -> 'run'
    - **도구**: `NLTK`의 `WordNetLemmatizer`, `PorterStemmer`, `LancasterStemmer`.

적용 가능한 상황

- 모든 텍스트 마이닝 및 자연어 처리(NLP) 작업의 전처리 단계에서 필수적으로 수행됩니다.
- 텍스트 분류, 감성 분석, 토픽 모델링, 정보 검색 등 다양한 NLP 애플리케이션의 성능을 향상시킵니다.

구현 방법

`NLTK`와 `KoNLPy` 라이브러리를 주로 사용합니다.

주의사항

- **언어 특성 고려**: 한국어와 영어는 언어적 특성이 다르므로, 각 언어에 맞는 전처리 기법과 도구를 사용해야 합니다. (e.g., 한국어는 형태소 분석이 필수적)
- **분석 목적 고려**: 전처리 수준은 분석 목적에 따라 달라질 수 있습니다. 예를 들어, 감성 분석에서는 부정어를 제거하지 않아야 합니다.
- **대소문자 통일, 특수문자 제거**: 일반적으로 텍스트를 소문자로 통일하고, 불필요한 특수문자나 숫자 등을 제거하는 과정도 함께 수행합니다.

```
# !pip install nltk konlpy
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer

# NLTK 데이터 다운로드 (최초 1회 실행)
# nltk.download('punkt')
# nltk.download('stopwords')
# nltk.download('wordnet')

# KoNLPy (한국어 형태소 분석기)
from konlpy.tag import Okt

# 1. 원시 텍스트
text_en = "Natural language processing (NLP) is a field of artificial intelligence
that gives computers the ability to understand human language."
text_ko = "자연어 처리는 인공지능의 한 분야로, 컴퓨터가 인간의 언어를 이해할 수 있도록
하는 기술입니다."

# 2. 토큰화 (Tokenization)
# 영어 단어 토큰화
tokens_en = word_tokenize(text_en)
print("--- 영어 단어 토큰화 ---")
print(tokens_en)
'''
['Natural', 'language', 'processing', '(', 'NLP', ')', 'is', 'a', 'field', 'of',
'artificial', 'intelligence', 'that', 'gives', 'computers', 'the', 'ability',
'to', 'understand', 'human', 'language', '.']
'''

# 한국어 형태소 분석 (KoNLPy)
okt = Okt()
tokens_ko = okt.morphs(text_ko) # 형태소 단위로 토큰화
print("\n--- 한국어 형태소 분석 (Okt) ---")
```

```
print(tokens_ko)
'''
['자연어', '처리', '는', '인공', '지능', '의', '한', '분야', '로', ',', '컴퓨터',
'가', '인간', '의', '언어', '를', '이해', '할', '수', '있도록', '하는', '기술', '입니
다', '.']
'''

# 3. 불용어 제거 (Stop Word Removal)
# 영어 불용어 제거
stop_words_en = set(stopwords.words('english'))
filtered_tokens_en = [word for word in tokens_en if word.lower() not in
stop_words_en and word.isalpha()] # 알파벳만 남기고 소문자 변환 후 불용어 제거
print("\n--- 영어 불용어 제거 ---")
print(filtered_tokens_en)
'''
['Natural', 'language', 'processing', 'NLP', 'field', 'artificial',
'intelligence', 'gives', 'computers', 'ability', 'understand', 'human',
'language']
'''

# 한국어 불용어 (직접 정의)
stop_words_ko = ['은', '는', '이', '가', '을', '를', '에', '의', '하다', '이다', '있
습니다', '합니다']
filtered_tokens_ko = [word for word in tokens_ko if word not in stop_words_ko]
print("\n--- 한국어 불용어 제거 ---")
print(filtered_tokens_ko)
'''
['자연어', '처리', '인공', '지능', '한', '분야', '로', ',', '컴퓨터', '인간', '언어',
'이해', '할', '수', '있도록', '하는', '기술', '입니다', '.']
'''

# 4. 표제어 추출 (Lemmatization) / 어간 추출 (Stemming)
# 영어 표제어 추출
lemmatizer = WordNetLemmatizer()
lemmas_en = [lemmatizer.lemmatize(word) for word in filtered_tokens_en]
print("\n--- 영어 표제어 추출 ---")
print(lemmas_en)
'''
['Natural', 'language', 'processing', 'NLP', 'field', 'artificial',
'intelligence', 'give', 'computer', 'ability', 'understand', 'human', 'language']
'''

# 영어 어간 추출
stemmer = PorterStemmer()
stems_en = [stemmer.stem(word) for word in filtered_tokens_en]
print("\n--- 영어 어간 추출 ---")
print(stems_en)
'''
['natur', 'languag', 'process', 'nlp', 'field', 'artifici', 'intellig', 'give',
'comput', 'abil', 'understand', 'human', 'languag']
'''

# 한국어는 형태소 분석 과정에서 이미 어간/원형에 가까운 형태로 분리되므로 별도의 표제어/
어간 추출은 잘 사용되지 않음.
```

```
# 필요시 품사 태깅 후 명사만 추출하거나 동사/형용사의 원형을 찾는 방식으로 활용.
nouns_ko = okt.nouns(text_ko) # 명사만 추출
print("\n--- 한국어 명사 추출 (Okt) ---")
print(nouns_ko)
'''
['자연어', '처리', '인공', '지능', '분야', '컴퓨터', '인간', '언어', '이해', '수', '기
술']
'''
```

결과 해석 방법

- 각 전처리 단계의 출력 결과를 통해 원시 텍스트가 어떻게 정제되고 변환되는지 확인할 수 있습니다.
- 토큰화는 텍스트를 분석 가능한 단위로 나누고, 불용어 제거는 의미 없는 단어를 걸러냅니다.
- 형태소 분석, 표제어/어간 추출은 단어의 다양한 형태를 통일하여 분석의 일관성을 높입니다.

장단점 및 대안

- **장점**:
  - 텍스트 데이터의 노이즈를 줄이고, 분석의 정확도와 효율성을 높입니다.
  - 모델이 텍스트의 의미를 더 잘 파악할 수 있도록 돕습니다.
- **단점**:
  - 언어별 특성과 분석 목적에 따라 적절한 전처리 기법을 선택하고 구현해야 하는 복잡성이 있습니다.
  - 과도한 전처리는 중요한 정보 손실로 이어질 수 있습니다.
- **대안**:
  - **정규 표현식 (Regular Expression)**: 특정 패턴의 문자열을 찾거나 제거하는 데 유용합니다.
  - **Byte Pair Encoding (BPE)**, **WordPiece**: 서브워드(subword) 토큰화 기법으로, OOV(Out-Of-Vocabulary) 문제에 강건하며, 딥러닝 기반 NLP 모델에서 널리 사용됩니다.

전처리 파이프라인 생성

```
set_stopwords = set(stopwords.words('english')) # 영어로 된 불용어를 가져다 집합으로
구성
# 아래의 함수는 리뷰글을 받아서, 쓸데없는 부분을 없애는 것이다.
def preprocessing(review, remove_stopwords=True):
    review_text = BeautifulSoup(review, 'html5lib').get_text() # html 태그제거
    review_text = re.sub("[^a-zA-Z]", " ", review_text)
    # 위의 문장은 subtitution, 즉 대체하라는 의미이다.
    # a-z, A-Z를 제외하고(제외한다는 표현은 ^ 로 표현) 나머지들은 모두 두 번째 항목인
빈칸으로 대체
    # 세번째 항목으로 주어진 review_text 항목에서 제거해라!
    # 이제부터 불용어를 제거하는 루틴을 돌립니다.
    if remove_stopwords: # 만약 이 함수를 쓸 때에 stopwords를 제거하라고 한다면
        words = review_text.split() # 현재 HTML제거하고, 알파벳으로만 구성된 리뷰글을
쪼개서 리스트로 만듦
        # 위의 split의 결과는 만약에 입력 리뷰글이 "I am a boy"였다면 [I, am, a, boy]
로 변환, words라는 리스트에 저장
        # words 리스트의 하나하나 보면서 만약에 하나가 set_stopwords에 속하지 않으면
```

```
불용어가 아니다.
        words = [w for w in words if not w in set_stopwords]
        # 예를 들어 [I, am, a, boy]였고, 불용어가 am, a였다면, 결과로 [I, boy]
        review_text = ' '.join(words)
        # 결과적으로 "I boy"가 됨
    review_text = review_text.lower() # 전부 소문자로 대체
    return review_text
```

시각화해서 확인하기

1. 히스토그램

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.hist(review_len_by_words, bins=50, alpha=0.5, color='r')
plt.hist(review_len_by_alphabet, bins=50, alpha=0.5, color='g')
plt.yscale('log',nonpositive='clip') # 음수는 잘라준다.
plt.title("Review Length Histogram")
plt.xlabel("Review Length")
plt.ylabel("Number of Reviews")
plt.show()
```

2. 통계적 수치

```
import numpy as np
print("최대 단어수를 가지는 문장은 몇개의 단어인가?",
np.max(review_len_by_words))
print("최소 단어수를 가지는 문장은 몇개의 단어인가?",
np.min(review_len_by_words))
print("평균적으로 몇개의 단어를 가지는가?", np.mean(review_len_by_words))
print("문장에 있는 단어들 수의 표준편차는?", np.std(review_len_by_words))
print("문장의 단어수들의 중간값은?", np.median(review_len_by_words))
print("문장의 하위 10% 길이는?", np.percentile(review_len_by_words, 10,
interpolation='nearest'))
```

- boxplot으로 확인해보기

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))
axes[0].boxplot([review_len_by_words], showmeans=True)
axes[1].boxplot([review_len_by_alphabet], showmeans=True)
plt.tight_layout()
plt.show()
```

3. wordcloud 그려보기

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

wordcloud = WordCloud(stopwords = STOPWORDS, background_color='black',
width=800, height=600).generate(' '.join(reviews))

plt.figure(figsize=(15,10))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

4. ylabel 데이터 확인해보기

```
import seaborn as sns
import matplotlib.pyplot as plt
sentiment = imdb_pd['sentiment'].value_counts()
print(sentiment)
fig, axe = plt.subplots(ncols=1)
fig.set_size_inches(6,3)
sns.countplot(x = imdb_pd['sentiment'])
plt.show()
```