캡스톤디자인(1)

Web Crawling, NLP. Word Cloud를 이용한 기사 통합 검색 시스템

목차

1. 개요 및 목적

1.1. 프로젝트 개발 동기

현재 살아가고 있는 지식 정보화 시대에서 사람들의 정보 습득은 주로 검색을 통해 이루어진다. 하지만 정보의 양이 워낙 방대하기 때문에 검색한 결과를 원래 목적에 맞게 받아들이는데 시간이 필요하고 이에 따라 검색에서 습득까지 불필요한 시간과 에너지가 소비된다.

기존에 원하는 키워드의 뉴스를 통합 검색하여 정보를 얻기까지 과정은 다음과 같다. 먼저 원하는 키워드를 검색하면 이 키워드를 포함하는 여러 뉴스 매체의 기사들이 제공이 된다. 이러한 뉴스들의 내용을 파악하기 위해 사용자는 하나씩 클릭하여 각 뉴스 내용을 읽어야한다. 이 중에는 중복되는 내용의 기사가 있을 수도 있고 그 중복으로 인해 원하는 키워드의 다른 여러 뉴스를 확인하기 위해서는 원하는 다른 뉴스가 나올 때까지 찾아보고 다시 그 기사를 클릭하여 확인해야한다. 이 과정은 걸리는 시간과 노력 면에서 비효율적이다.

이러한 이유들로 인하여 '원하는 키워드를 검색했을 때 최근 뉴스의 전체적인 정보를 먼저 대략적으로 파악하고 그 다음 원하는 뉴스의 상세 내용을 보게 할 수는 없을까'라는 생각에 이 프로젝트를 시작하게 되었다.

1.2. 프로젝트 개요

해당 프로젝트는 web crawling으로 뉴스 기사 데이터를 가져오고 주로 데이터 시각화에 많이 사용되는 Word Cloud를 사용하여 기사 내용을 요약한다. 이 과정에서 Word2Vec을 사용하여 Word Cloud의 성능을 향상시킨다. 해당 프로그램의 작동 과정을 살펴보면 다음과 같다.

먼저 사용자가 검색어를 입력하고 매체를 선택하면 해당 매체들의 검색 결과를 Word Cloud 형식으로 제공한다. 사용자는 해당 Word Cloud로 검색어에 대한 결과의 대략적인 내용과 흐름을 파악할 수 있다. 이 중 기사의 상세 내용이 궁금한 단어를 클릭하면 해당 단어가 들어가 있는 기사들의 제목과 링크를 제공해주게 된다.

1.3. 프로젝트 일정

3월	1주차: 프로젝트 개요 도출 2주차: 유사 서비스 검토 3주차: 기능 확정
4월	3월 4주차 ~ 4월 1주차: 소요 기술, 소요 자원 검토 4월 2주차 ~ 4월 4주차: 소요 기술 학습
5월	5월 1주차 ~ 5월 4주차: 뉴스 매체 페이지 분석 및 전처리

2. 배경

2.1. Word Cloud

워드 클라우드(Word Cloud)는 텍스트 데이터의 시각화를 위해 사용되는 도구이다. 특히 텍스트 데이터에서 가장 빈번하게 등장하는 단어들을 시각적으로 강조하여 표현하는 방식이다. 이는 텍스트의 상대적인 빈도를 시각적으로 이해하기 쉽게 해주어 텍스트 데이터의 특징을 파악하는데 도움이 된다. Word Cloud는 다음과 같은 과정으로 생성된다.

1. 텍스트 데이터 수집: Word Cloud를 생성하기 위해서는 분석하려는 텍스트 데이터가 필요하다. 이 데이터는 웹 문서, 뉴스 기사 등 다양한 자원에서 가져올 수 있다.
2. 전처리: 텍스트 데이터를 분석하기 전에 전처리 작업을 수행해야 한다. 이는 불필요한 문

자, 구두점, 불용어(“a”, “the”, “is”와 같이 빈번하게 등장하는 단어)를 제거하거나 형태소 분석을 통해 단어의 기본 형태로 변환하는 것과 같은 작업을 포함할 수 있다.

3. 단어 빈도 계산: 전처리된 텍스트 데이터에서 단어의 빈도를 계산한다. 즉, 각 단어가 텍스트 내에서 얼마나 자주 등장하는지 측정한다. 이를 통해 단어의 상대적 중요도를 파악할 수 있다.
4. Word Cloud 생성: 단어 빈도 계산이 완료되면 Word Cloud를 생성한다. 이는 단어의 빈도에 따라 단어를 시각적으로 표현한 그래픽이다. 빈도가 높은 단어는 크고 강조되게 표현되고 빈도가 낮은 단어는 작고 덜 강조되게 표현된다. 일반적으로 Word Cloud는 단어의 크기, 색상, 배치 등을 이용하여 시각적인 효과를 부여한다.

Wordcloud는 주제 분석, 텍스트 요약, 시각적인 표현을 통한 커뮤니케이션 등 다양한 분야에서 활용된다. 데이터 사이언스, 자연어 처리, 정보 검색, 마케팅 분석 등에서 텍스트 데이터의 시각화와 분석에 널리 사용된다.

D3 cloud는 D3.js(DS) 라이브러리를 사용하여 생성된 Word Cloud를 의미한다. D3.js는 데이터 시각화를 위한 자바스크립트 라이브러리로 웹 브라우저에서 동적으로 상호 작용하는 시각화를 만들기 위해 사용된다. D3 cloud는 Word Cloud와 마찬가지로 텍스트 데이터에서 가장 빈번하게 등장하는 단어를 시각적으로 표현하고 단어의 빈도에 따라 단어의 크기와 배치를 조정하여 시각화 한다. D3 cloud가 python 기반 Word Cloud와 다른 점은 D3 cloud는 Word Cloud에 대한 세부적인 커스터마이징이 가능하다는 점이다. 웹 페이지에서 사용자와의 상호작용, 애니메이션 효과, 클릭 이벤트 등과 같은 기능을 추가할 수 있다. 이는 시각화 렌더링 과정에서 SVG(scalable vector graphic element)를 사용하여 x, y 좌표로 이뤄진 단어들을 사용하기 때문이다.

해당 프로젝트는 python을 사용한 Word Cloud에 클릭 이벤트를 적용할 예정이기 때문에 D3 cloud의 작동과정을 참고하여 python을 이용한 클릭 이벤트가 적용 가능한 Word Cloud를 생성할 예정이다.

2.2. Word2Vec

Word2Vec은 단어를 벡터로 변환하는 도구로 단어의 의미와 관련성을 수학적으로 표현하는 기법이다. Word2Vec는 Neural Network 기반의 언어 모델로 주변 단어의 문맥을 이용하여 단어를 벡터로 표현한다. 이를 통해 단어 간 유사성을 계산하고 단어 간 관련성을 파악할 수 있다.

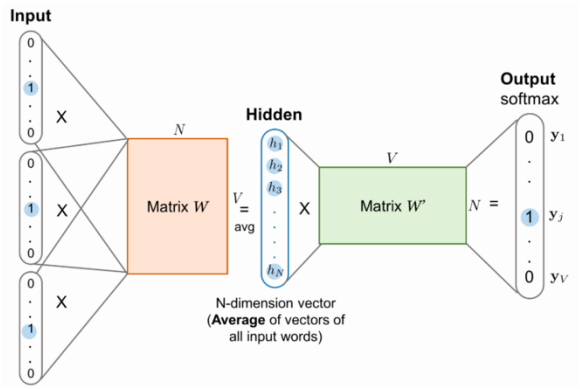
Word2Vec은 주로 두가지 방법으로 구현된다: CBOW(Continuous Bag of Words)와 Skip-gram이다.

1. CBOW: CBOW는 context 내 주변 단어들을 입력으로 사용하여 중심 단어를 예측하는 방식이다. CBOW는 여러 주변 단어들의 평균을 이용하여 중심 단어를 예측한다. 주로 데이터셋의 크기 작을 경우에 많이 사용한다.

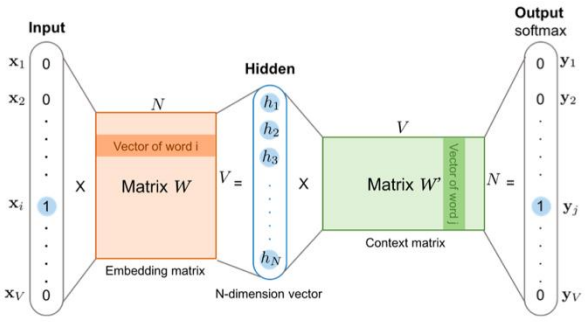
2. Skip-gram: Skip-gram은 CBOW의 반대 개념으로 중심 단어를 입력으로 사용하여 주변 단어를 예측하는 방식이다. Skip-gram은 중심 단어와 주변 단어를 한 쌍으로 사용하여 모델을 학습한다. 주로 데이터셋의 크기가 클 경우 잘 작동한다.

Word2vec을 학습하면 앞서 언급했듯이 단어의 유사성을 계산할 수 있다. 유사성은 벡터 간의 거리나 코사인 유사도(cosine similarity)로 측정된다. 학습된 Word2vec 모델을 사용하면 "king"과 "queen" 간의 관련성이 "man"과 "woman" 사이의 관련성과 유사한지 등으로 파악할 수 있다.

Word2vec는 자연어 처리(Natural Language Processing), 문서 분류, 문서 군집화, 정보 검색 등 다양한 응용 분야에서 활용된다. 단어 간 의미적 유사성, 단어의 특성을 활용하여 다양한 텍스트 기반 작업을 수행할 수 있다.



[그림 1. CBOW]



[그림 2. Skip-gram]

3. 설계 및 구현

3.1. 개발 환경

구분	상세 내용
OS	MacOS
IDE	Jupyter notebook, Visual Studio Code
개발 언어	Python
버전 관리	Git hub 예정

3.2. 개발 요구 명세: Requirement Specification

3.2.1. 기능적 요구사항

① 사용자는 검색어를 입력하고 뉴스 매체를 선택할 수 있어야 한다.
- 뉴스매체는 다수 선택 가능하다.

② 웹 크롤링을 통한 데이터셋 준비
- 검색어와 뉴스 매체가 주어졌을 때 해당 뉴스 매체들의 웹 사이트에 가서 해당 검색에 대한 기사 검색 결과를 크롤링해 제공해야한다.

③ 효율적인 Word Cloud를 위한 데이터셋의 1차 가공
- 데이터셋에 대한 형태소 분석이 필요하다.
- 형태소 분석이 완료된 데이터셋에 대해 불용어, 중요하지 않은 단어를 제외해야 한다.

④ 효율적인 Word Cloud를 위한 데이터셋의 2차 가공
- 형태소 분석과 불용어 처리가 완료된 데이터셋을 이용해 유사한 의미를 가진 단어들 간의 그룹화가 필요하다.
- Word2Vec을 이용해 단어 벡터간 유사성을 활용하여 그룹화한다.

⑤ Word Cloud에 클릭 이벤트를 적용하여 제공해야 한다.
- 사용자가 Word Cloud의 각 단어를 클릭할 수 있어야 한다.

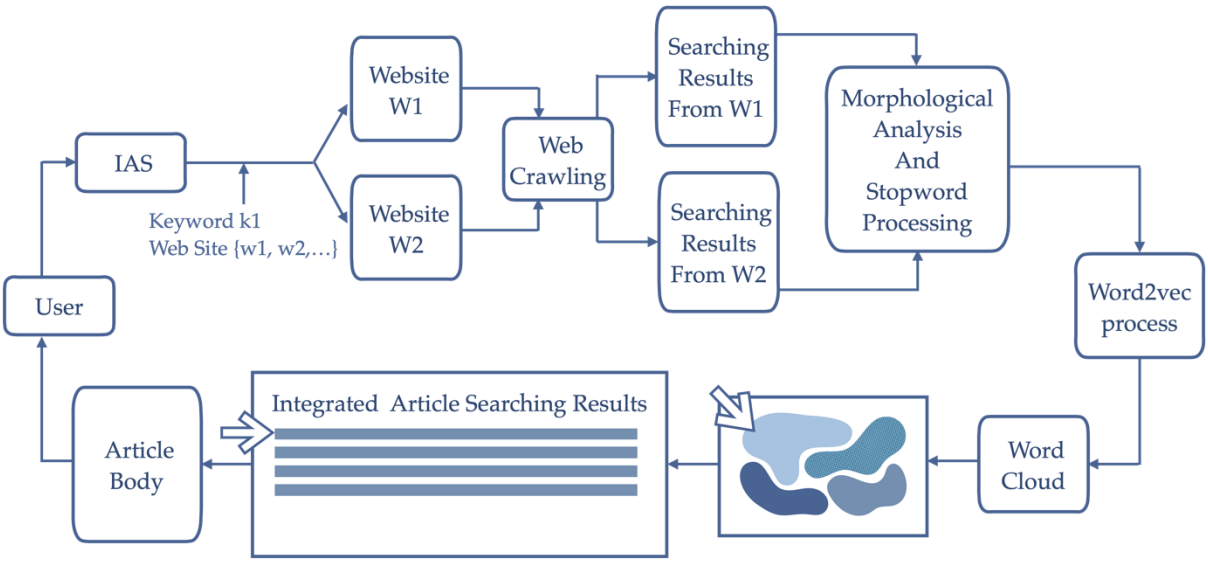
- 단어를 클릭했을 때 해당 단어가 포함된 기사들의 링크를 제공해야 한다.
- 해당 링크는 기사들의 제목을 보여주고 URL로 연결되는 형식으로 제공해야 한다.

3.2.2. 비기능적 요구사항

① 사용자가 검색을 했을 때 응답 시간은 3초 이내여야 한다.
- 웹 크롤링 시 응답 시간을 최소화 해야 한다.

② 사용자가 프로그램의 인터페이스를 부가적인 설명 없이도 이해하여 사용할 수 있도록 제공해야 한다.
- 이해하기 쉬운 인터페이스를 개발 해야 한다.

3.3. 개발 설계: Software Architecture



3.4. 구현 내용

3.4.1. Preprocessing: Dataset

데이터셋은 selenium을 통해 검색어를 선택한 뉴스매체에서 검색한 결과를 가져와 준비한다/

```
divs_to_remove = soup.find_all('div', {'class': 'image-area'})
for div in divs_to_remove:
    div.decompose()
divs_to_remove = soup.find_all('div', {'id': 'ad_tag'})
for div in divs_to_remove:
    div.decompose()
divs_to_remove = soup.find_all('div', {'class': 'iwmads is-loaded'})
for div in divs_to_remove:
    div.decompose()
```

```
article_body = []
for p_tag in soup.find_all('p', {'align': lambda x: x and 'justify' in x
    p_tag_following_text = p_tag.next_sibling.strip()
    if p_tag_following_text is not None:
        print(p_tag_following_text)
        article_body.append(p_tag_following_text)
    else:
        print("None")
```

Article body를 살펴보면 다음과 같다.

경찰이 경기 구리, 인천 등 수도권에서 전세사기를 벌인 일당에게 범죄단체조직죄를 적용해 검찰에 넘겼다.

구리경찰서는 31일 전세사기 총책 ㄱ씨, 대부중개업체 직원 ㄴ씨, 명의대여자 ㄷ씨 등 20명에게 사기 등 혐의를 적용해 사건을 검찰에 송치했다. 이 중 전세사기에 적극적으로 가담한 14명에게는 범죄단체조직죄가 적용됐다.

ㄱ씨는 지난 2020년 10월부터 2022년 10월까지 부동산 컨설팅사무실을 운영하며 동시진행 및 무자본갭투자 방식으로 오피스텔을 매입한 뒤 전세계약을 해 세입자에게서 전세보증금을 가로챈 혐의를 받고 있다. 이들의 피해자는 900여명으로 현재까지 파악된 피해 전세보증금은 2500억원에 이른다.

ㄱ씨는 오피스텔 등이 새로 건설되면 바로 세입자를 구해 전세 보증금을 받아 건물을 매입하는 '동시진행 및 무자본 갭투자'로 보유 주택 수를 늘린 것으로 조사됐다. ㄱ씨가 세운 부동산 컨설팅사무실 직원들은 분양대행사를 상대로 무자본갭투자를 영업하는 영업사원, 이들을 교육하고 수수료를 정산하는 부장, 영업교육 총괄 이사 등으로 나뉘어 있던 것으로 파악됐다. ㄱ씨 일당은 건축주가 내건 분양 성공 수수료를 받기도 했다.

ㄱ씨는 자신의 명의 사용이 어렵게 되자 대부업체를 통해 명의대여자를 모으기도 했다. ㄴ씨 등 대부업체 직원 2명은 자신들에게 돈을 빌리러 온 ㄷ씨 등 4명에게 명의대여를 해주면 돈을 벌 수 있게 해주겠다는 식으로 전세사기에 가담하게 한 것으로 나타났다.

경찰은 ㄱ씨가 운영한 부동산 컨설팅사무실 직원뿐 아니라 명의대여자를 소개해준 대부업체 직원, 명의대여자 등이 조직적으로 범행을 저지른 것으로 보고 범죄단체조직죄를 적용했다.

경찰은 ㄱ씨 일당의 오피스텔 등을 중개한 공인중개사 6명에게는 공인중개사법 위반 혐의를 적용했다. 이들 공인중개사는 법정 수수료 이상을 받고 중개행위를 한 것으로 파악됐다. 다만 공인중개사들이 ㄱ씨 일당과 함께 전세사기를 벌인 부분은 입증하지 못해 사기 및 범죄단체조직죄는 적용하지 못했다.
경찰은 추가 전세사기 피해자가 있는지 찾는 한편, ㄱ씨 등이 보유한 자산을 추적해 몰수·추징할 예정이다.

이승욱 기자

3.4.2. Preprocessing: Kkma

태깅을 통해 품사를 구분하고 동사와 명사만 추출하였다. Stopword를 지정하여 추가적인 불용어를 처리해주었다. 이 부분은 Okt, hannanum, komoran도 동일하다.

```
tagging_kkma = []
tagging_okt = []
tagging_hannanum = []
tagging_komoran = []

tagged_result = [tagging_kkma, tagging_okt, tagging_hannanum, tagging_ko
lib = [kkma, okt, hannanum, komoran]
for i in range(4):
    tagged_result[i] += lib[i].pos(article)
```

```
tagging_kkma
```

```
[('경찰', 'NNG'),
 ('이', 'JKS'),
 ('경기', 'NNG'),
 ('구리', 'NNG'),
 (',', 'SP'),
 ('인천', 'NNG'),
 ('등', 'NNB'),
 ('수도권', 'NNG'),
 ('에서', 'JKM'),
 ('전세', 'NNG'),
 ('사기', 'NNG'),
 ('를', 'JKO'),
 ('벌이', 'VV'),
 ('ㄴ', 'ETD'),
 ('일당', 'NNG'),
 ('에게', 'JKM'),
 ('범죄', 'NNG'),
 ('단체', 'NNG'),
 ('조직', 'NNG'),
```

```
preprocess_kkma = []
preprocess_okt = []
preprocess_hannanum = []
preprocess_komoran = []

preprocess_result = [preprocess_kkma, preprocess_okt,
                     preprocess_hannanum, preprocess_komoran]
lib = [kkma, okt, hannanum, komoran]

for i in range(4):
    for j, k in tagged_result[i]:
        if (k=='Noun' or k=='NNG' or k=='NNP' or 'N' or k=='Verb' or k==
            preprocess_result[i].append(j)
```

```
preprocess_kkma
```

```
'등',
'수도권',
'에서',
'전세',
'사기',
'를',
'벌이',
'ㄴ',
'일당',
'에게',
'범죄',
'단체',
'조직',
'죄',
'를',
'적용',
'하',
'어',
'검찰',
'에',
```

전처리한 결과 시각화를 위해 Word Cloud를 사용하였다.

```
word_cloud_kkma = WordCloud(font_path = 'NanumBarunGothic', max_font_siz

# 사이즈 설정 및 화면에 출력
plt.figure(figsize=(5,4))
plt.imshow(word_cloud_kkma)
plt.axis('off')
```

```
(-0.5, 399.5, 199.5, -0.5)
```

3.4.3. Preprocessing: Okt

```
word_cloud_okt = WordCloud(font_path = 'NanumBarunGothic', max_font_size

# 사이즈 설정 및 화면에 출력
plt.figure(figsize=(5,4))
plt.imshow(word_cloud_okt)
plt.axis('off')
```

```
(-0.5, 399.5, 199.5, -0.5)
```

3.4.4. Preprocessing: Hannanum

```
word_cloud_hannanum = WordCloud(font_path = 'NanumBarunGothic', max_font

# 사이즈 설정 및 화면에 출력
plt.figure(figsize=(5,4))
plt.imshow(word_cloud_hannanum)
plt.axis('off')
```

```
(-0.5, 399.5, 199.5, -0.5)
```

3.4.5. Preprocessing: Komoran

```
word_cloud_komoran = WordCloud(font_path = 'NanumBarunGothic', max_font_

# 사이즈 설정 및 화면에 출력
plt.figure(figsize=(5,4))
plt.imshow(word_cloud_komoran)
plt.axis('off')
```

```
(-0.5, 399.5, 199.5, -0.5)
```

4. 개선 사항 및 향후 계획

4.1.1. 개선 사항

현재 분석 완료된 뉴스 매체는 한겨레 하나이므로 나머지 뉴스 매체에 대한 분석이 추가적으로 필요하다. 또한 전처리 역시 조금 더 개선이 되어야한다. 불용어 처리의 경우 기사마다 불용어로 처리되는 단어들이 다른데 현재는 다른 임베딩을 사용하고 있지 않아서 이 부분에서 어려움을 겪고 있었다. 이 부분은 후에 임베딩 과정에서 유사도가 높은 단어끼리 그룹화를 하고 상위 단어들을 Word Cloud에 포함하는 과정에서 해결이 될 것이라고 예상하고 있다. 따라서 전처리 향상 과정 중에서 Word Cloud 생선 전 Word2Vec 임베딩을 사용하여 유사한 데이터끼리 그룹화를 할 예정이며 클릭이 가능한 Word Cloud를 파이썬으로 구현할 예정이다.

4.1.2. 향후 계획

6월	전처리, UI 완료
7월	Word2Vec을 이용한 전처리 향상 완료
8월	Clickable Word Cloud 구현 완료
9월	수정 및 테스트

5. 참고 문헌

[1] 'Natural Language Processing with Transformers; Building Language Applications with Hugging Face', Lewis Tunstall, Leandro von Werra & Thomas Wolf, O'Reilly

[2] 'Word Cloud Explorer: Text Analytics Based on Word Clouds', Heimerl, Florian. 2014 47th Hawaii international Conference on System Sciences ISBN