Final Report

Team Winitech

(임도영, 정도, 김도형)

2023. 5. 1. ~ 2023. 6. 16.

목 차

최 종 보 고 서

2023년6월16일 금요일
Team Winitech (임도영, 정도, 김도형)

본 프로젝트는 ㈜위니텍과 경북대학교 K-Digital Training의 협력으로 진행되었습니다.

I. 서론
1. 기업 소개

㈜위니텍은 기술로 이기는 강한 기업이 되겠다는 뜻을 담아 1997년 추교관 대표가 설립. 재난 사고로 인한 피해를 최소화하는 통합긴급사태관리시스템(IEMS: Intergrated Emergency Management System)을 개발 및 구축하는 국내 1위의 독보적인 기업이다. 다양한 기술력을 바탕으로 119 긴급구조시스템의 핵심 역량을 보유한 기업으로 자리매김하였으며 2005년부터 해외 시장에 진출하여 동남 아시아, 아프리카, 중남미 등으로 영역을 넓혀가고 있다.

2. 프로젝트 배경

이번 프로젝트의 배경은 재난 관리 매뉴얼과 관련된 고객의 요청 사항을 수행 중 발생한 이슈를 해결하기 위함으로, 유사 항목 추출을 위한 자연어 처리 기반의 모델이 필요한 것으로 판단된다. 데이터는 국가법령정보센터(https://www.law.go.kr)에서 수집하여 사용할 예정이다.

이번 프로젝트는 2023년 5월 2일부터 6월 15일까지 진행되었다.

II. 프로젝트 개요
1. 팀 구성 및 업무 분장

	역할	주 업무
임도영	Project Manager	Web
정도	Team Member	Python
김도형	Team Member	Reports

2. 개발 환경

	버전
Python	3.9.13
scikit-learn	1.2.1
Windows	11 Pro (64-bit)

3. Work Breakdown Structure

1.0.0	<착수 및 프로젝트 관리>
1.1.0	프로젝트 관리
1.1.1	일간 회의
1.1.2	최종 회의
2.0.0	<계획서 작성>
2.1.0	프로젝트 계획서
2.1.1	배경 및 목적

2.1.2	프로젝트 일정
2.1.3	예상 이슈
2.1.4	WBS
2.1.5	최종 계획서
3.0.0	<데이터>
3.1.0	수집 및 전처리
3.1.1	데이터 수집
3.1.2	데이터 전처리
4.0.0	<모델링 및 검증>
4.1.0	모델링
4.1.1	머신 러닝 기반 모델링
~~4.1.2~~	~~딥 러닝 기반 모델링~~
4.1.3	모델 검증
5.0.0	<디자인>
5.1.0	메인 디자인
5.1.1	디자인 구성
5.1.2	디자인 컨펌
5.1.3	디자인 수정 보완
6.0.0	<Web 구현>
6.1.0	Web 구현 및 수정 보완
6.1.1	Web 구현
6.1.2	Web 수정 보완
7.0.0	<구현 및 수정 보완>
7.1.0	테스트
7.1.1	구현
7.1.2	최종 수정 보완

4. 일정

1 주 차	분석 기획 (프로젝트 계획 수립) ~ 5/5
2 주 차	데이터 준비 (자료 수집) ~ 5/12
3 주 차	데이터 정제 (전처리) ~ 5/19
4 주 차	모델 구현 (모델 구축) ~ 5/26
5 주 차	모델 구현 (모델 평가 및 검증) ~ 6/2
6 주 차	모델 구현 (페이지 구축) ~ 6/9

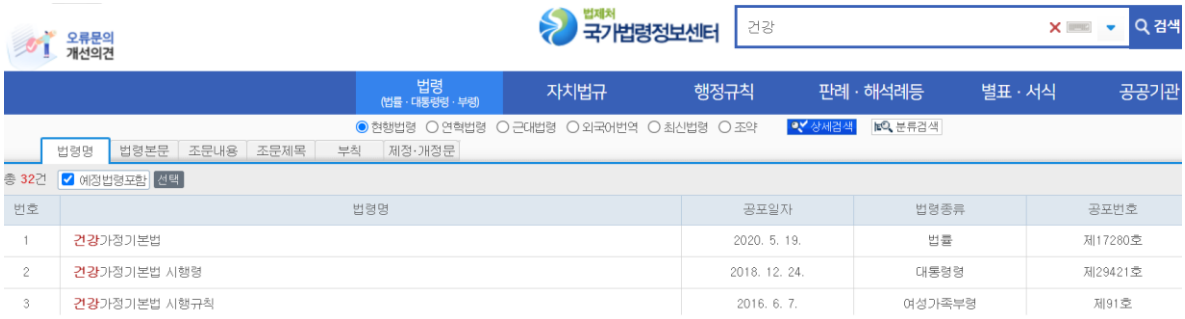
7 주 차	평가 및 전개 & 프로젝트 평가 및 보고 ~ 6/16

III. 프로젝트 과정

1. 데이터

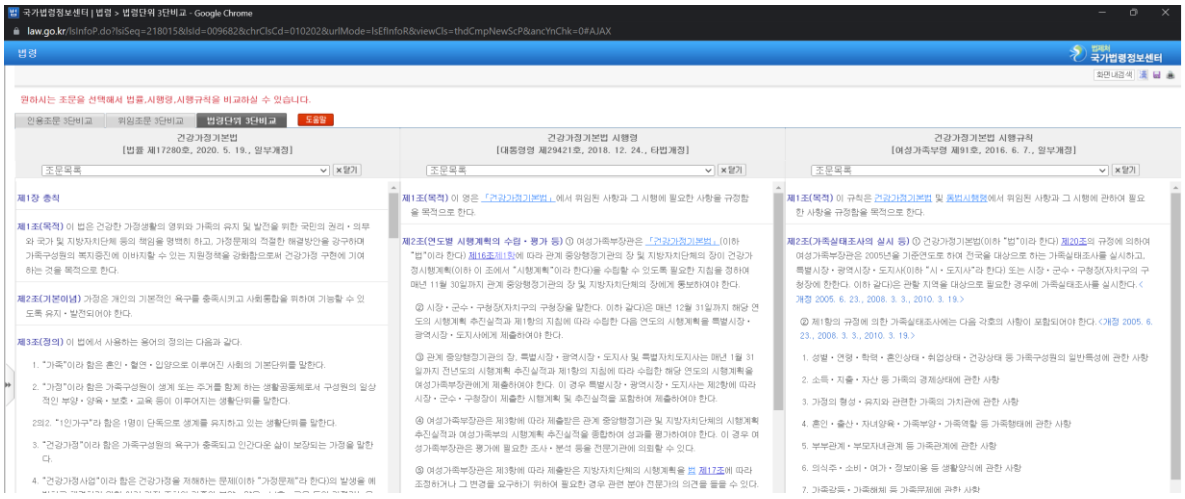
(1) 데이터 출처

데이터는 법제처에서 관리하는 국가법령정보센터(https://www.law.go.kr)에서 수집하였다.



(2) 데이터 수집 및 전처리

기본법, 시행령, 시행 규칙에 대한 비교가 필요하므로 크롤링 코드를 작성하여 각 법령 당 6개의 데이터를 수집하였다. 법령 전체와 조문 제목만 추출한 내용에 해당한다. 실제로 비교하게 되는 것은 조문 제목만이다. 하지만 크롤링 과정에서 이슈가 발생하였고 해결 방안을 모색하게 되었다.



(3) 데이터 크롤링 코드

<크롤링 코드: 전체 내용>

```
import csv
import time
import random
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
rd = random.randint(3,9)

# 크롬
driver = webdriver.Chrome(r"C:\workspace\chromedriver.exe")

# 경범죄 처벌법
driver.get("https://www.law.go.kr/lsSc.do?section=&menuId=1&subMenuId=15&tabMenuId=81&eventGubun=060101&query=%EA%B2%BD%EB%B2%94%EC%A3%84%20%EC%B2%98%EB%B2%8C%EB%B2%95#undefined")

time.sleep(2)
driver.implicitly_wait(10)

# 데이터 추출
sentence1 = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CLASS_NAME, 'scr_ctrl')))
time.sleep(rd)
st1 = list(sentence1.text)

with open('경범죄 처벌법.csv', 'w', encoding = 'utf-8') as f:
    for ct1 in st1:
        f.write(ct1)


# 경범죄 처벌법 시행령
```

```
driver.get('https://www.law.go.kr/lsSc.do?section=&menuId=1&subMenuId=15&tabMe
nuId=81&eventGubun=060101&query=%EA%B2%BD%EB%B2%94%EC%A3%84%20%EC%B2%98%EB%B2%
8C%EB%B2%95%20%EC%8B%9C%ED%96%89%EB%A0%B9#undefined')


time.sleep(2)

driver.implicitly_wait(10)


# 데이터 추출

sentence2 = WebDriverWait(driver, 10).until(

    EC.presence_of_element_located((By.CLASS_NAME, 'scr_ctrl')))

time.sleep(rd)

st2 = list(sentence2.text)


with open('경범죄 처벌법 시행령.csv', 'w', encoding = 'utf-8') as f:

    for ct2 in st2:

        f.write(ct2)


# 경범죄 처벌법 시행규칙

driver.get('https://www.law.go.kr/lsSc.do?section=&menuId=1&subMenuId=15&tabMe
nuId=81&eventGubun=060101&query=%EA%B2%BD%EB%B2%94%EC%A3%84%20%EC%B2%98%EB%B2%
8C%EB%B2%95%20%EC%8B%9C%ED%96%89%EA%B7%9C%EC%B9%99#undefined')


time.sleep(2)

driver.implicitly_wait(10)


# 데이터 추출

sentence3 = WebDriverWait(driver, 10).until(

    EC.presence_of_element_located((By.CLASS_NAME, 'scr_ctrl')))

time.sleep(rd)

st3 = sentence3.text


with open('경범죄 처벌법 시행규칙.csv', 'w', encoding = 'utf-8') as f:

    for ct3 in st3:
```

```
        f.write(ct3)

# 종료
driver.quit()
```

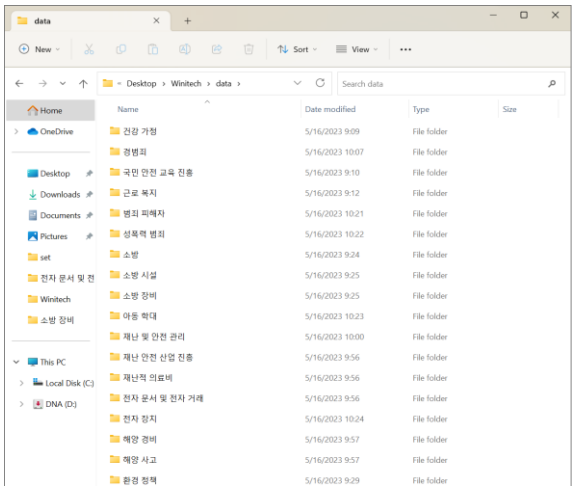
<크롤링 코드: 조문 내용>

```
import time
import random
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
rd = random.randint(3,9)

# 크롬
driver = webdriver.Chrome(r"C:\workspace\chromedriver.exe")

# 경범죄 처벌법
driver.get("https://www.law.go.kr/lsSc.do?section=&menuId=1&subMenuId=15&tabMenuId=81&eventGubun=060101&query=%EA%B2%BD%EB%B2%94%EC%A3%84%20%EC%B2%98%EB%B2%8C%EB%B2%95#undefined")

time.sleep(2)
driver.implicitly_wait(10)

# 데이터 추출
law1 =[]
sentence = WebDriverWait(driver, 10).until(
    EC.presence_of_all_elements_located((By.CLASS_NAME, 'bl')))
time.sleep(rd)
for i in sentence:
```

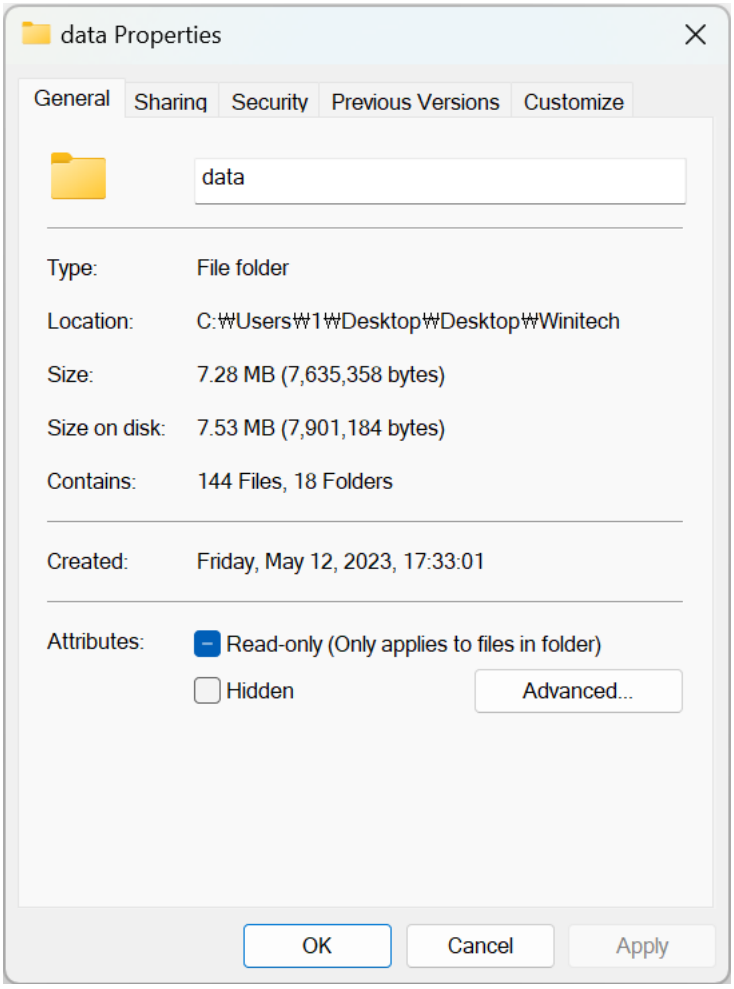
```
    law1.append(i.text)


with open('경범죄 처벌법 조문.csv', 'w', encoding = 'utf-8') as f:
    for ct1 in law1:
        f.write(ct1)
        f.write('\n')


# 경범죄 처벌법 시행령
driver.get('https://www.law.go.kr/lsSc.do?section=&menuId=1&subMenuId=15&tabMenuId=81&eventGubun=060101&query=%EA%B2%BD%EB%B2%94%EC%A3%84%20%EC%B2%98%EB%B2%8C%EB%B2%95%20%EC%8B%9C%ED%96%89%EB%A0%B9#undefined')


time.sleep(2)
driver.implicitly_wait(10)


# 데이터 추출
law2 =[]
sentence = WebDriverWait(driver, 10).until(
    EC.presence_of_all_elements_located((By.CLASS_NAME, 'bl')))
time.sleep(rd)
for i in sentence:
    law2.append(i.text)


with open('경범죄 처벌법 시행령 조문.csv', 'w', encoding = 'utf-8') as f:
    for ct2 in law2:
        f.write(ct2)
        f.write('\n')


# 경범죄 처벌법 시행규칙
driver.get('https://www.law.go.kr/lsSc.do?section=&menuId=1&subMenuId=15&tabMenuId=81&eventGubun=060101&query=%EA%B2%BD%EB%B2%94%EC%A3%84%20%EC%B2%98%EB%B2%8C%EB%B2%95%20%EC%8B%9C%ED%96%89%EA%B7%9C%EC%B9%99#undefined')
time.sleep(2)
```

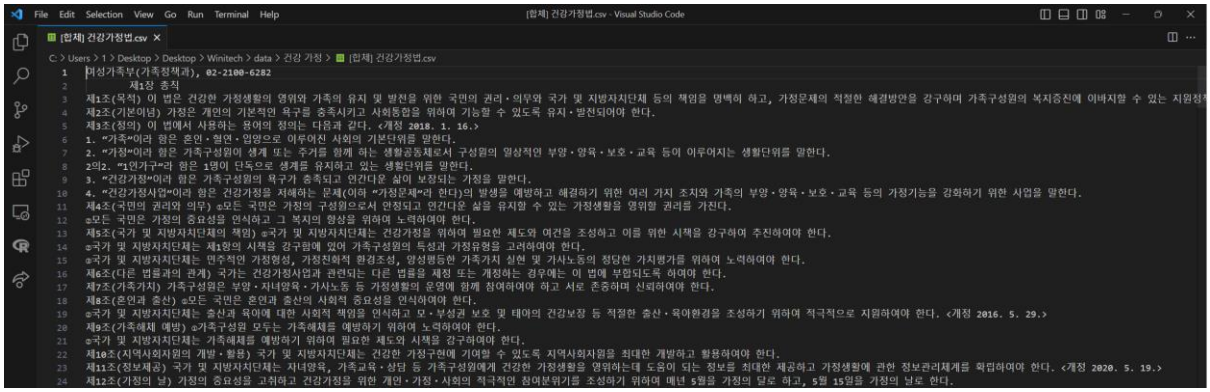
```
driver.implicitly_wait(10)

# 데이터 추출
law3 =[]
sentence = WebDriverWait(driver, 10).until(
    EC.presence_of_all_elements_located((By.CLASS_NAME, 'bl')))
time.sleep(rd)
for i in sentence:
    law3.append(i.text)

with open('경범죄 처벌법 시행규칙 조문.csv', 'w', encoding = 'utf-8') as f:
    for ct3 in law3:
        f.write(ct3)
        f.write('\n')

# 종료
driver.quit()
```

데이터는 소방, 재난, 안전 등 다양한 분야의 전체 18개 법령을 수집하였다.



Name	Date modified	Type	Size
[합체] 건강가정법	5/15/2023 16:53	Comma Separated V...	88 KB
files	5/15/2023 16:52	Windows Batch File	1 KB
건강가정법 기본법 조문만	5/15/2023 15:04	Comma Separated V...	29 KB
건강가정법 기본법	5/15/2023 15:04	Comma Separated V...	29 KB
건강가정법 시행규칙 조문만	5/15/2023 15:04	Comma Separated V...	9 KB
건강가정법 시행규칙	5/15/2023 15:04	Comma Separated V...	9 KB
건강가정법 시행령 조문만	5/15/2023 15:04	Comma Separated V...	7 KB
건강가정법 시행령	5/15/2023 15:04	Comma Separated V...	7 KB



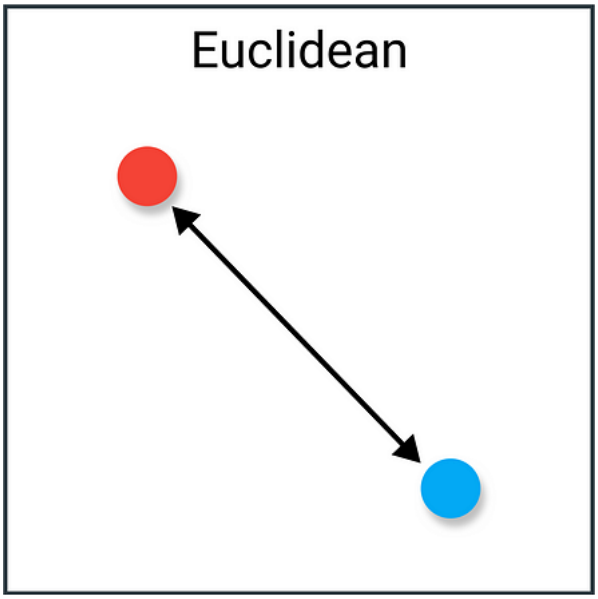


2. 모델

본 프로젝트의 주제는 자연어 처리 (Natural Language Processing, NLP) 기반 유사 항목 도출 모델 분석이다. 주제가 등장한 배경으로는 공공 기관의 재난 관련 솔루션 제공 기업으로서 재난 관리 매뉴얼과 관련된 프로젝트를 수행하였고 고객의 요청 사항 중 재난 관리 매뉴얼의 수직 비교에 대한 이슈가 있었다. 이에 대해 유사 항목 추출을 위한 자연어 처리 기반의 비교 모델이 필요한 것으로 판단하여 여러 가지 기법에 대한 정보가 필요하게 된 것이다. 그에 따라 Euclidean, Manhattan, Cosine 그리고 Jaccard의 4가지 방법으로의 접근을 모색하였다.

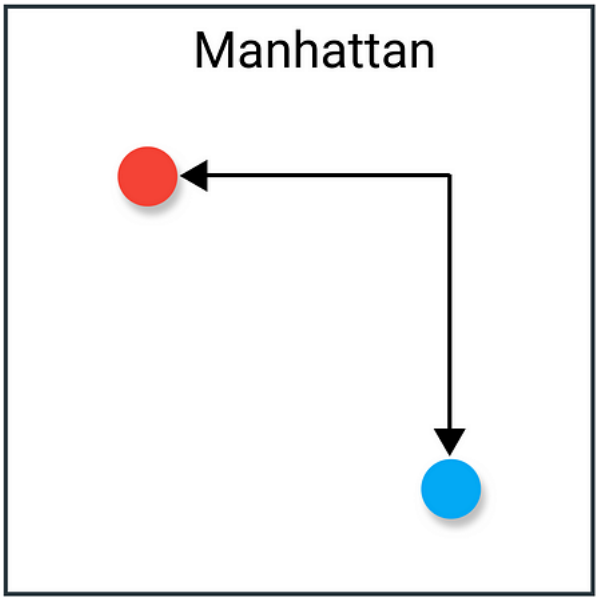
scikit-learn의 metrics 모듈의 euclidean_distances, manhattan_distances, cosine_similarity 그리고 NLTK의 metrics 모듈의 jaccard_distance를 활용하여 구현하였다. 먼저 비교하고자 하는 CSV 파일 2개에서 텍스트를 추출한다. 그런 다음 TfidfVectorizer를 통해 단어의 가중치를 조정한 BoW (Bag of Words) 벡터를 만든다. 그 후 해당하는 코드를 통해 유사도를 추출하게 된다.

1. Euclidean



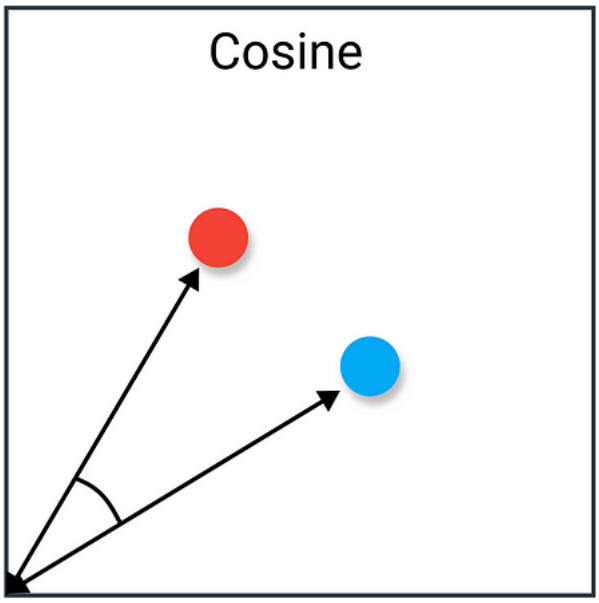
두 점 사이의 최단 거리를 구하는 방법
범위 없음 / 값이 작을수록 유사

2. Manhattan



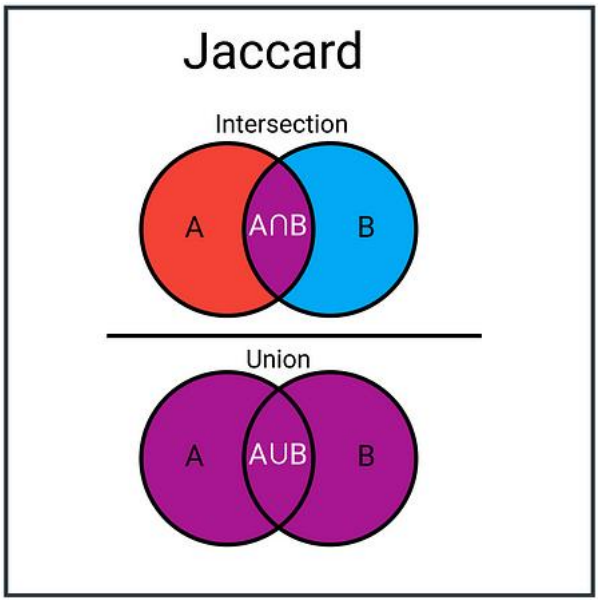
두 점 사이를 가로지르지 않고 갈 수 있는 최단 거리를 구하는 방법
값이 작을수록 유사

3. Cosine



두 벡터의 사잇각을 코사인으로 구하는 방법
-1 이상1 이하/ 1에 가까울수록 유사

4. Jaccard



두 집합 사이의 유사도를 측정하는 방법
0 이상 1 이하/ 1에 가까울수록 유사

Picture	Method	Application	Features	Disadvantages	Formula
	**Euclidean Distance**	General distance measurement, Clustering, Classification, Regression	Measures the straight line distance between two points in n-dimensional space.	Sensitive to outliers, Can be affected by scale differences	O(n) Fast
	**Manhattan Distance**	Distance on grid networks, Routing algorithms, Image processing	Measures the distance between two points on a grid network, where movement is limited.	Ignores diagonal movement, not useful for high-dimensional data,	O(n) Fast
	**Cosine Similarity**	Text document clustering, Text analysis, Recommendation systems	Measures the cosine of the angle between two vectors	Ignores magnitude of vectors, Not useful for negative values or high degree of correlation data	O(n) Fast
	**Jaccard Similarity**	Set similarity measurement, Text analysis, recommendation systems	Measures the similarity between two sets by comparing their intersection and union.	Ignores magnitude of sets, May not be as useful for continuous data	O(n) Fast

4가지 방법은 각각의 특징에 따른 장단점이 존재한다. 1) Euclidean은 계산하기가 비교적 간단하며 효율적이고 계산 속도가 빠르다. 수치형 데이터에 적합하다. 2) Manhattan은 이상치에 상대적으로 덜 민감하다. 범주형 데이터에 적합하다. 3) Cosine은 NLP 작업에서 텍스트 문서의 유사성을 비교하는 데에 널리 쓰인다. 그리고 4) Jaccard는 직관적인 해석과 성능 효율성, 이진 데이터 처리에 대한 장점이 있다.

IV. 프로젝트 시각화

1. 모델 구현

먼저 모델 구현을 하기 위해, 프로젝트의 목적을 다시 이해할 필요가 있다. 매뉴얼은 표준 매뉴얼 -> 실무 매뉴얼 -> 행동 매뉴얼의 순서로 작성되며, 선행 매뉴얼의 기본 방침을 구체화 또는 매뉴얼 작성 기관의 담당 업무를 기술하는 형태이다. 문서를 공유하여 검토 단계를 거친 후 확정 배포 되면 그 다음 단계의 매뉴얼 작성에 참고하게

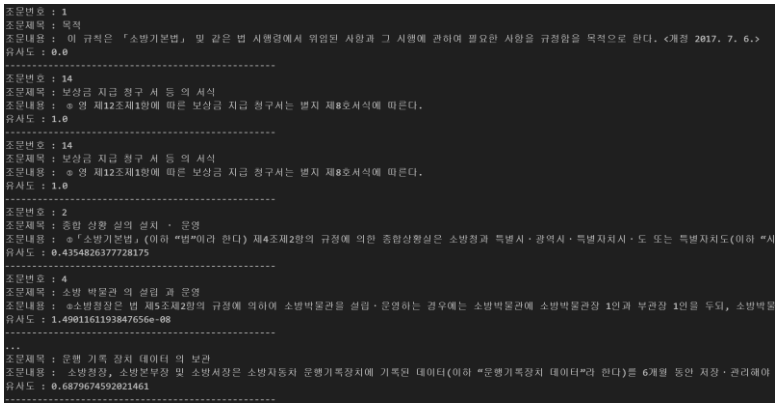
되는데, 상위 매뉴얼의 항목이 하위 매뉴얼에 누락 되거나 잘못 적용된 것이 없는지 비교 검토가 필요하게 된다. 문서 자체의 방대함, 대량의 매뉴얼 (표준 매뉴얼: 1, 실무 매뉴얼: 소관 기관별 작성 (소수), 행동 매뉴얼: 모든 관련 기관 작성 (다수))으로 검토가 곤란한 측면이 있었다.

구분	내용
위기관리 표준매뉴얼	▪ 국가적 차원에서 관리가 필요한 재난에 대하여 재난관리 체계와 관계 기관의 임무와 역할을 규정한 문서로 위기대응 실무매뉴얼의 작성 기준이 되며, 재난관리주관기관의 장이 작성 ▪ 다만, 다수의 재난관리주관기관이 관련되는 재난에 대해서는 관계 재난관리주관기관의 장과 협의하여 행정안전부장관이 위기관리 표준매뉴얼을 작성할 수 있음
위기대응 실무매뉴얼	▪ 위기관리 표준매뉴얼에서 규정하는 기능과 역할에 따라 실제 재난대응에 필요한 조치사항 및 절차를 규정한 문서로 재난관리주관기관의 장과 관계 기관의 장이 작성 ▪ 이 경우 재난관리주관기관의 장은 위기대응 실무매뉴얼과 위의 위기관리 표준매뉴얼을 통합하여 작성할 수 있음
현장조치 행동매뉴얼	▪ 재난현장에서 임무를 직접 수행하는 기관의 행동조치 절차를 구체적으로 수록한 문서로 위기대응 실무매뉴얼을 작성한 기관의 장이 지정한 기관의 장이 작성하되, 시장·군수·구청장은 재난유형별 현장조치 행동매뉴얼을 통합하여 작성할 수 있음 ▪ 다만, 현장조치 행동매뉴얼 작성 기관의 장이 다른 법령에 따라 작성한 계획·매뉴얼 등에 재난유형별 현장조치 행동매뉴얼에 포함될 사항이 모두 포함되어 있는 경우 해당 재난유형에 대해서는 현장조치 행동매뉴얼이 작성된 것으로 봄

매뉴얼	작 성 기 관	주 요 내 용
위기관리 표준매뉴얼	재난관리주관기관 (중앙부처)	재난관리체계 및 기관별 임무와 역할
위기대응 실무매뉴얼	주관기관 및 유관기관	재난대응에 필요한 조치사항 및 절차 규정
현장조치 행동매뉴얼	실무매뉴얼 작성기관의 장이 지정한 기관	재난현장 임무 수행기관의 행동절차 수록

그에 따라 문서 파일을 관리하고 있었으며 웹 기반의 관리 시스템을 개발하였다. 비교 검토 과정의 편의성 향상을 위해 항목 관리 시스템에 해당 기능을 도입하였다. 정의된 항목은 있었으나 규격화 되어 있지 않은 자유 형식의 문서로 인해 항목 비교 효율이 낮은 것이다. 그래서 이를 해결하고자 본 프로젝트를 진행하게 된 것이다.

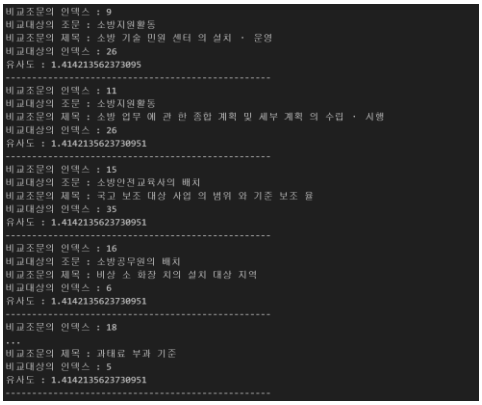
먼저 <방법 1>을 통해 유사도를 구해보고자 하였다. 결과와 코드 화면은 아래와 같다.




```python
import csv
import numpy as np
from konlpy.tag import Okt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import euclidean_distances

def preprocess_text(text):
    okt = Okt()
    tokens = okt.morphs(text)
    return tokens

# Extract text from CSV data
texts1 = []
for i in range(df1.shape[0]):
    texts1.append(df1.iloc[i, 1])

texts2 = []
for i in range(df2.shape[0]):
    texts2.append(df2.iloc[i, 1])

# Tokenize and preprocess texts
preprocessed_texts1 = [' '.join(preprocess_text(txt1)) for txt1 in texts1]
preprocessed_texts2 = [' '.join(preprocess_text(txt2)) for txt2 in texts2]

# Generate TF-IDF vectors for the corpus
vectorizer = TfidfVectorizer()
tfidf1 = vectorizer.fit_transform(preprocessed_texts1)
tfidf2 = vectorizer.transform(preprocessed_texts2)

# Calculate Euclidean distance for each pair of texts
distance_scores = euclidean_distances(tfidf1, tfidf2)

# Print the Euclidean distance scores
# for score in distance_scores:
#     print(score)
for distance in distance_scores:
    distance = list(distance)
    x = distance.index(min(distance))
    print(f'조문번호 : {df2.iloc[x,0]}')
    print(f'조문제목 : {preprocessed_texts2[x]}')
    print(f'조문내용 : {df2.iloc[x,2]}')
    print(f'유사도 : {min(distance)}')
    print("-"*50)
```

<방법 1>을 통한 유사도의 문제점은 어떤 조문과 어떤 조문을 비교하는지 알 수가 없었고 유사도의 결과를 확인할 방법이 없었다.

그래서 <방법 2>를 도입하게 되었다.


비교조문의 인덱스 : 9
비교대상의 조문 : 소방지원활동
비교조문의 제목 : 소방 기술 민원 센터 의 설치 · 운영
비교대상의 인덱스 : 26
유사도 : 1.414213562373095

비교조문의 인덱스 : 11
비교대상의 조문 : 소방지원활동
비교조문의 제목 : 소방 업무 에 관 한 종합 계획 및 세부 계획 의 수립 · 시행
비교대상의 인덱스 : 26
유사도 : 1.4142135623730951

비교조문의 인덱스 : 15
비교대상의 조문 : 소방안전교육사의 배치
비교조문의 제목 : 국고 보조 대상 사업 의 범위 와 기준 보조 율
비교대상의 인덱스 : 35
유사도 : 1.4142135623730951

비교조문의 인덱스 : 16
비교대상의 조문 : 소방공무원의 배치
비교조문의 제목 : 비상 소 화장 치의 설치 대상 지역
비교대상의 인덱스 : 6
유사도 : 1.4142135623730951

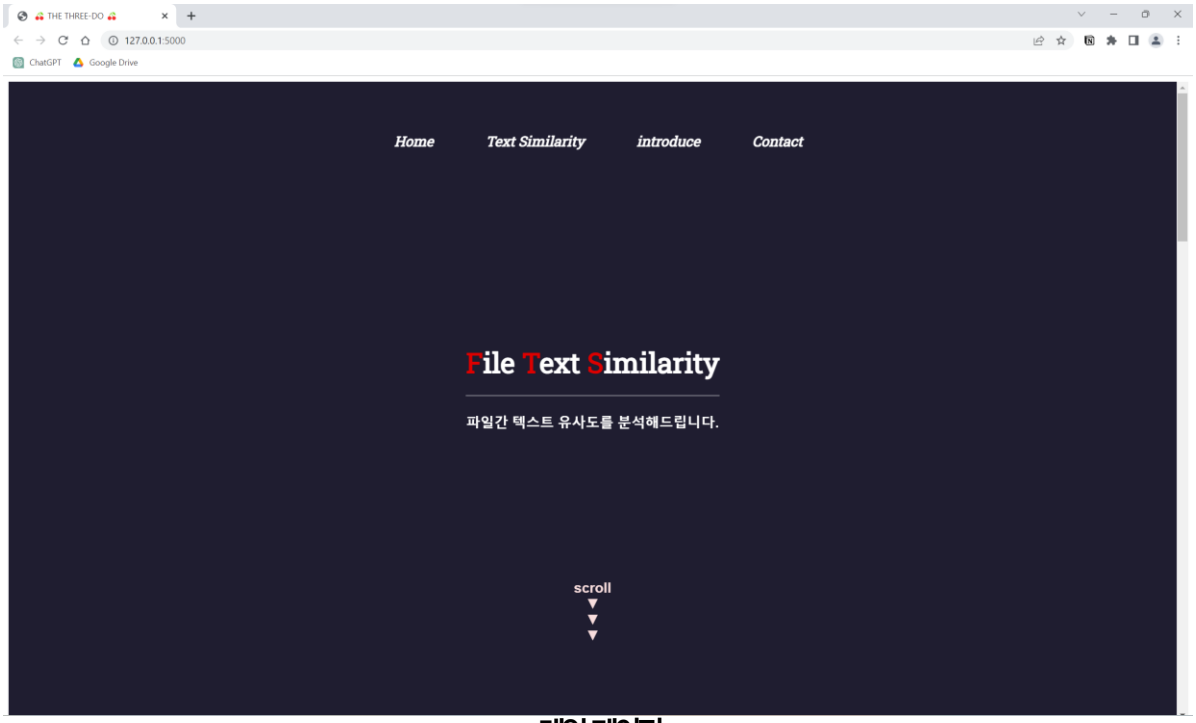
비교조문의 인덱스 : 18
...
비교조문의 제목 : 과태료 부과 기준
비교대상의 인덱스 : 5
유사도 : 1.4142135623730951

```python
import csv
import numpy as np
from konlpy.tag import Okt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import euclidean_distances

def preprocess_text(text):
    okt = Okt()
    tokens = okt.morphs(text)
    return tokens

# Extract text from CSV data
texts1 = []
for i in range(fire2.shape[0]):
    texts1.append(fire2.iloc[i, 0])

texts2 = []
for i in range(fire1.shape[0]):
    texts2.append(fire1.iloc[i, 0])

# Tokenize and preprocess texts
preprocessed_texts1 = [' '.join(preprocess_text(txt1)) for txt1 in texts1]
preprocessed_texts2 = [' '.join(preprocess_text(txt2)) for txt2 in texts2]

# Generate TF-IDF vectors for the corpus
vectorizer = TfidfVectorizer()
tfidf1 = vectorizer.fit_transform(preprocessed_texts1)
tfidf2 = vectorizer.transform(preprocessed_texts2)

# Calculate Euclidean distance for each pair of texts
distance_scores = euclidean_distances(tfidf1, tfidf2)

# Print the Euclidean distance scores


for i in range(len(distance_scores)):
    score = distance_scores[i]
    score = list(score)
    x = score.index(max(score))
    print(f'비교조문의 인덱스 : {fire2.iloc[i,1]}')
    print(f'비교대상의 조문 : {fire1.iloc[x,0]}')
    print(f'비교조문의 제목 : {preprocessed_texts1[i]}')
    print(f'비교대상의 인덱스 : {fire1.iloc[x,1]}')
    print(f'유사도 : {max(score)}')
    print("-"*50)

# 유클리디안 유사도 결과 저장
with open("유클리디안 유사도.txt", "w", encoding="utf-8") as output_file:
    for i in range(len(distance_scores)):
        score = distance_scores[i]
        score = list(score)
        x = score.index(max(score))
        output_file.write(f'비교조문의 인덱스 : {fire2.iloc[i,1]}')
        output_file.write(f'비교대상의 조문 : {fire1.iloc[x,0]}')
        output_file.write(f'비교조문의 제목 : {preprocessed_texts1[i]}')
        output_file.write(f'비교대상의 인덱스 : {fire1.iloc[x,1]}')
        output_file.write(f'유사도 : {max(score)}')
        output_file.write("\n")
```

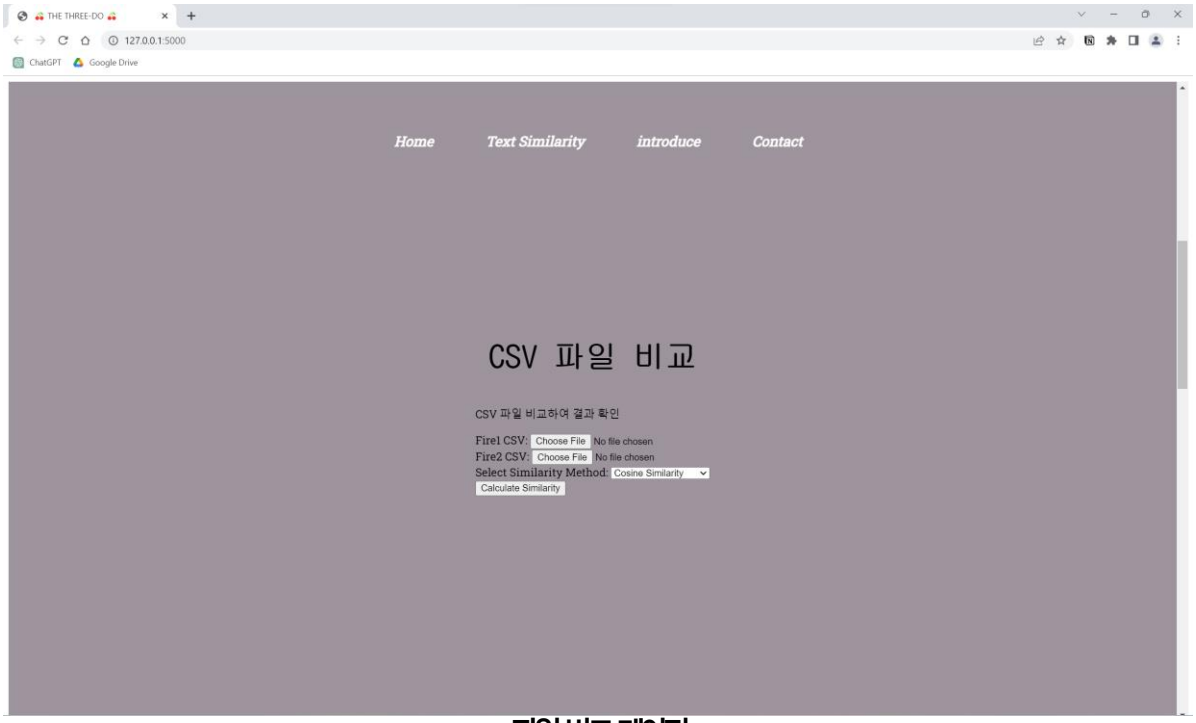
<방법 2>를 통해 비교하고자 하는 조문과 비교 대상의 인덱스와 내용을 알 수 있었고 직관적으로 파악할 수 있게 되었다.

2. Web 구현

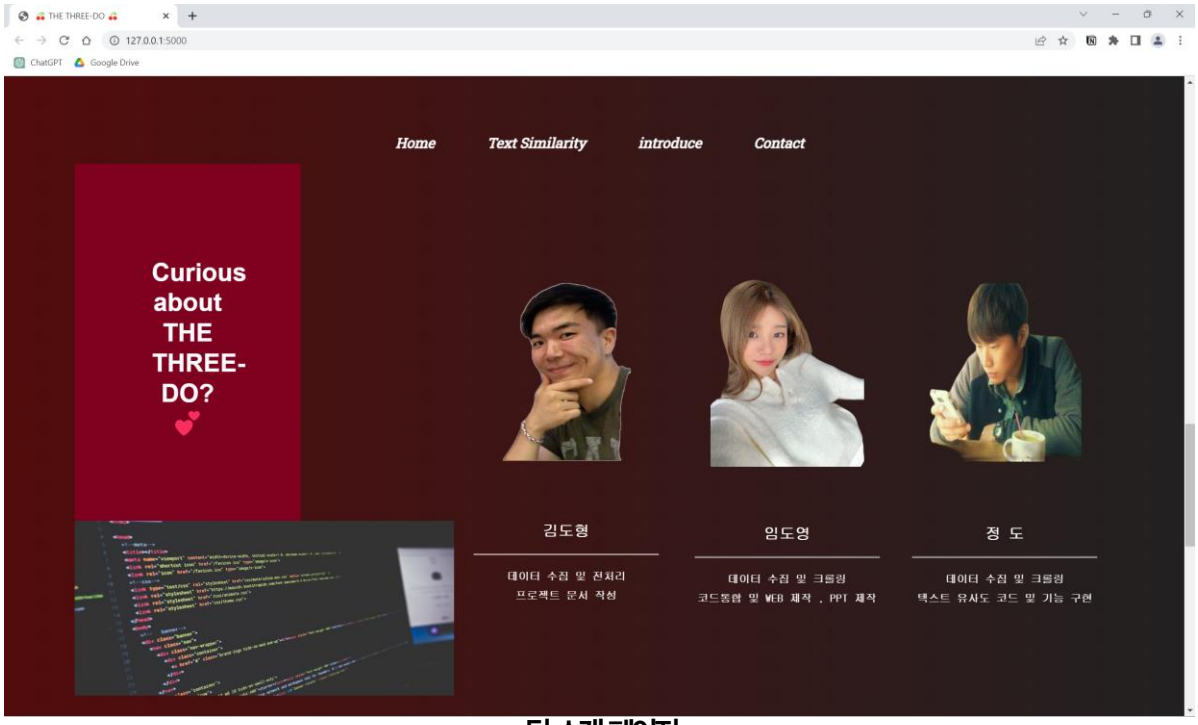
Python 기반의 Web Framework인 Flask를 통해 Web 구현을 시도하였다. HTML (HyperText Markup Language)로 뼈대를 구성하고 CSS (Cascading Style Sheets)로 디자인을 구성하였다.



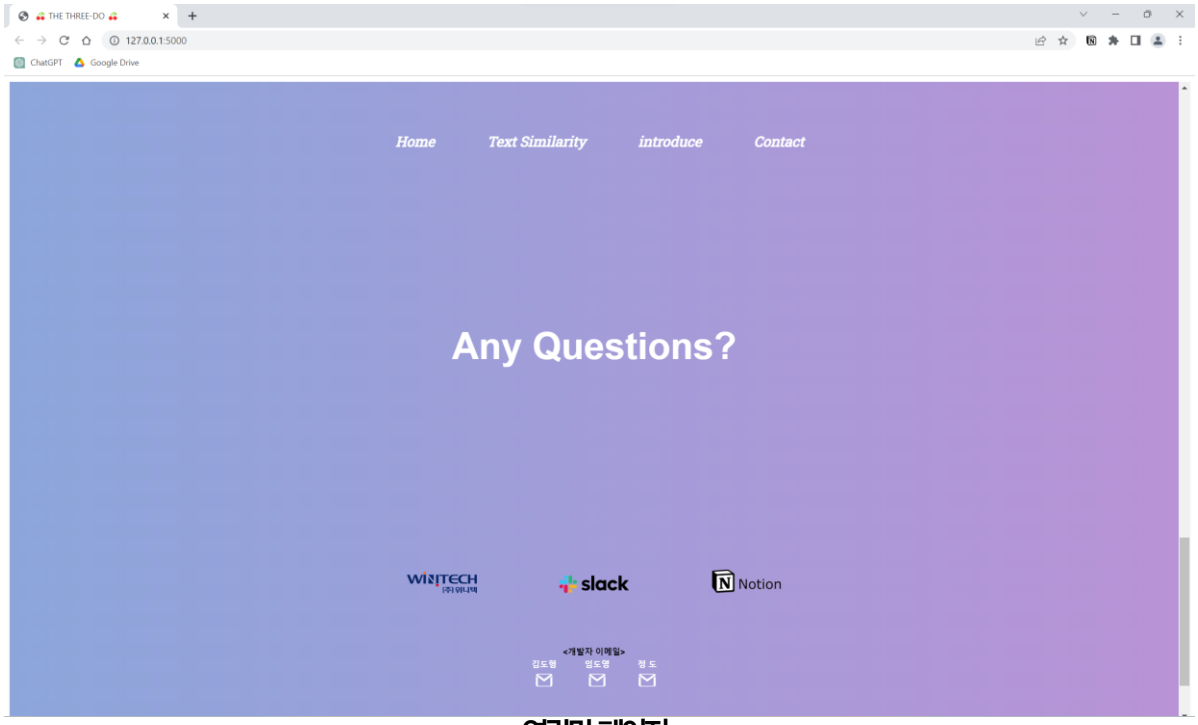
<메인 페이지>



<파일 비교 페이지>



<팀 소개 페이지>

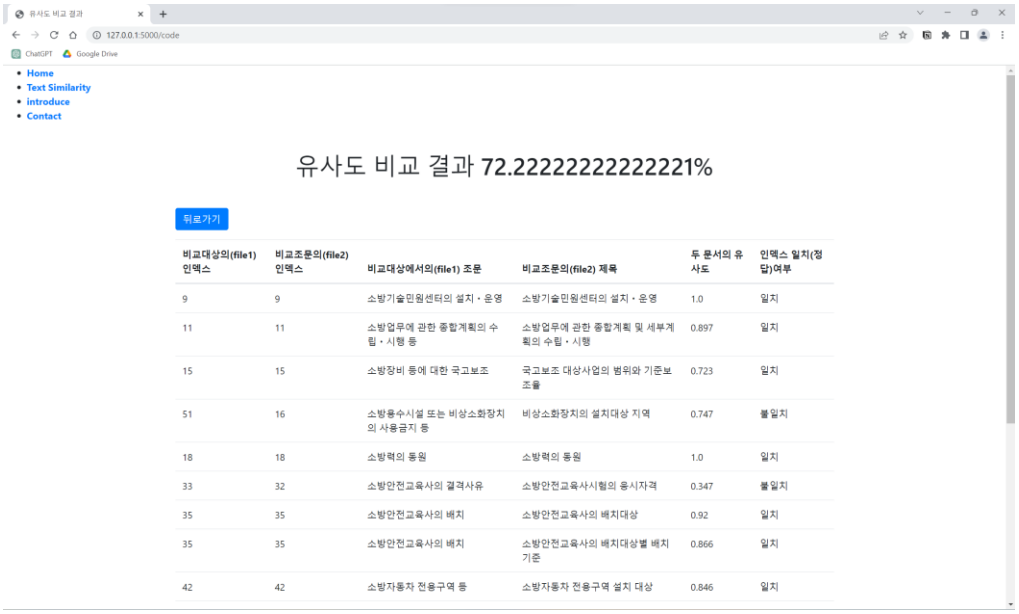


<연락망 페이지>

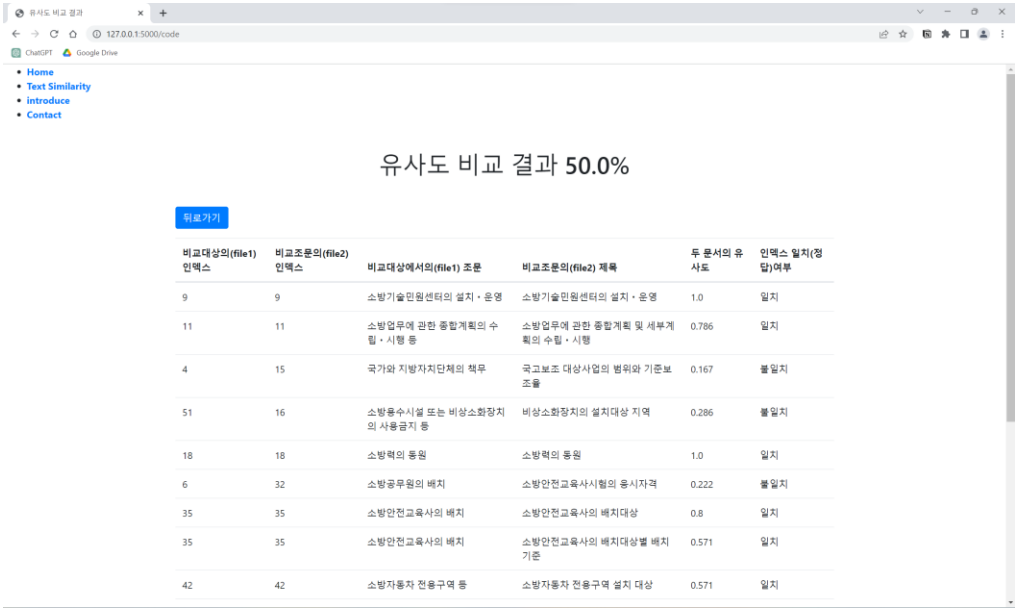
V. 프로젝트 결과

1. 결과

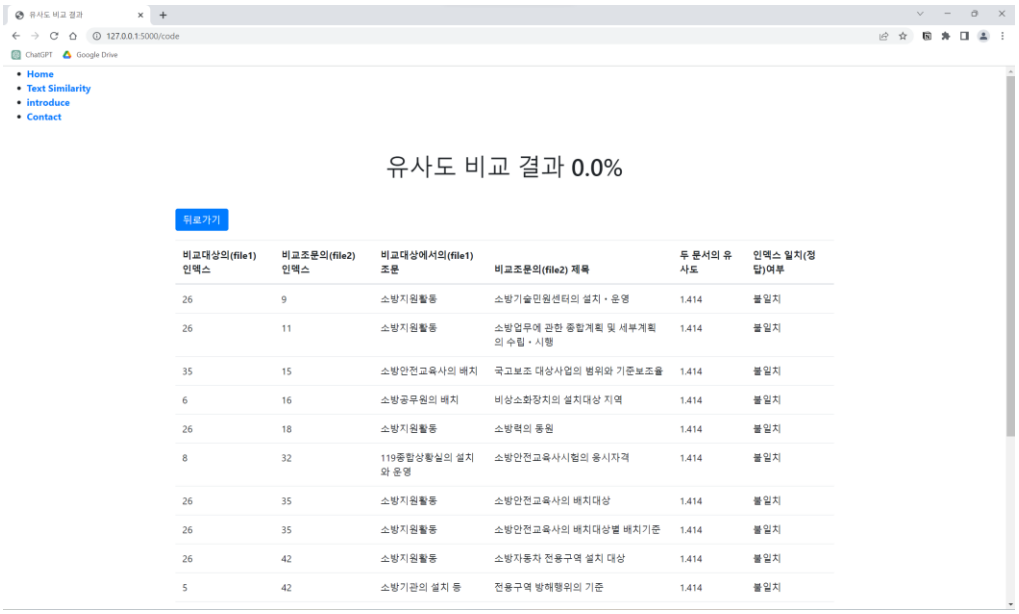
크롤링한 법령들의 기본법, 시행령, 시행 규칙의 3 가지 파일에 대해 <기본법 x 시행령>, <시행령 x 시행 규칙>, <기본법 x 시행 규칙>의 2 가지 조합을 구성하여 4 가지 유사도에 대해 테스트를 진행하였다. 모두 60 건의 테스트를 진행하였다.

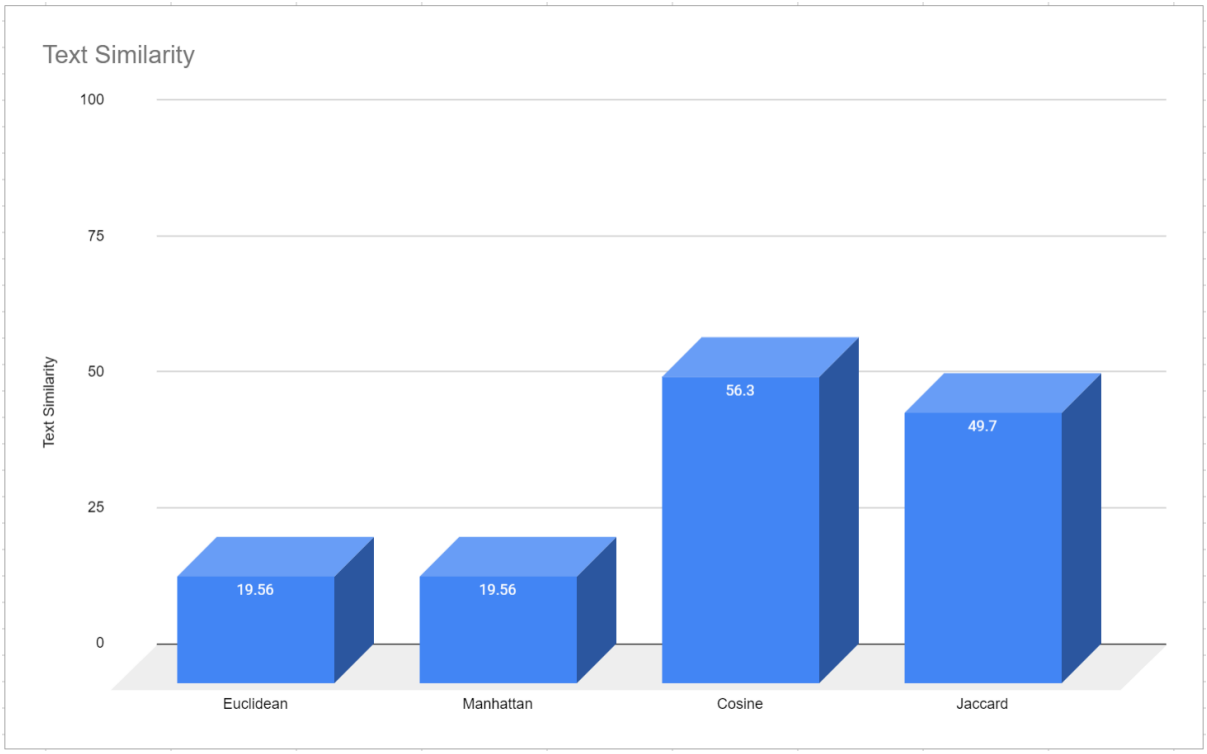


비교대상의(file1) 인덱스	비교조문의(file2) 인덱스	비교대상에서의(file1) 조문	비교조문의(file2) 제목	두 문서의 유사도	인덱스 일치(정답)여부
9	9	소방기술민원센터의 설치·운영	소방기술민원센터의 설치·운영	1.0	일치
11	11	소방업무에 관한 종합계획의 수립·시행 등	소방업무에 관한 종합계획 및 세부계획의 수립·시행	0.897	일치
15	15	소방장비 등에 대한 국고보조	국고보조 대상사업의 범위와 기준보조율	0.723	일치
51	16	소방용수시설 또는 비상소화장치의 사용금지 등	비상소화장치의 설치대상 지역	0.747	불일치
18	18	소방력의 동원	소방력의 동원	1.0	일치
33	32	소방안전교육사의 결격사유	소방안전교육사시험의 응시자격	0.347	불일치
35	35	소방안전교육사의 배치	소방안전교육사의 배치대상	0.92	일치
35	35	소방안전교육사의 배치	소방안전교육사의 배치대상별 배치기준	0.866	일치
42	42	소방자동차 전용구역 등	소방자동차 전용구역 설치 대상	0.846	일치



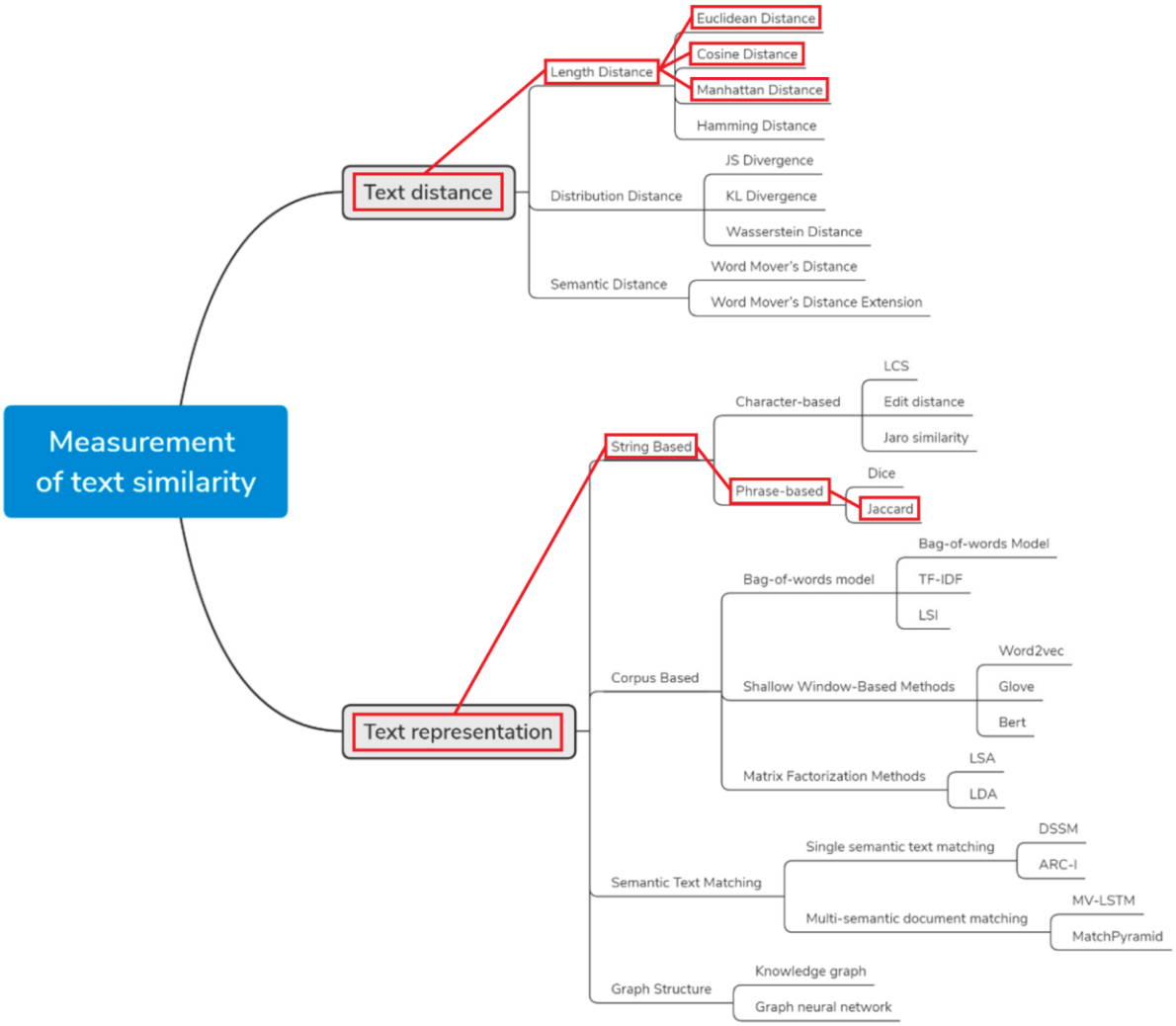
비교대상의(file1) 인덱스	비교조문의(file2) 인덱스	비교대상에서의(file1) 조문	비교조문의(file2) 제목	두 문서의 유사도	인덱스 일치(정답)여부
9	9	소방기술민원센터의 설치·운영	소방기술민원센터의 설치·운영	1.0	일치
11	11	소방업무에 관한 종합계획의 수립·시행 등	소방업무에 관한 종합계획 및 세부계획의 수립·시행	0.786	일치
4	15	국가와 지방자치단체의 책무	국고보조 대상사업의 범위와 기준보조율	0.167	불일치
51	16	소방용수시설 또는 비상소화장치의 사용금지 등	비상소화장치의 설치대상 지역	0.286	불일치
18	18	소방력의 동원	소방력의 동원	1.0	일치
6	32	소방공무원의 배치	소방안전교육사시험의 응시자격	0.222	불일치
35	35	소방안전교육사의 배치	소방안전교육사의 배치대상	0.8	일치
35	35	소방안전교육사의 배치	소방안전교육사의 배치대상별 배치기준	0.571	일치
42	42	소방자동차 전용구역 등	소방자동차 전용구역 설치 대상	0.571	일치





그 결과, 적게는 0%, 높게는 72%의 유사도 결과를 나타내었으며 평균적으로 Cosine: 56.30%, Jaccard: 49.70%, Manhattan: 19.56%, Euclidean: 19.56%의 순서로 Cosine 이 가장 높은 유사도를 나타낸다는 것을 알 수 있었다.

2. 한계 및 보완점



논문 출처: Jiapeng Wang & Yihong Dong (2020). Measurement of Text Similarity: A Survey, *11*(9), 421. https://doi.org/10.3390/info11090421

텍스트 유사도를 비교하는 방법은 굉장히 많았으나 이번 프로젝트에서 선택한 방법은 4 가지에 국한되었다. 기간 또한 한 달 반 밖에 되지 않았고 데이터의 표본 수 또한 작았다고 생각한다. 다시 기회와 시간이 주어진다면, 더 많은 질 좋은 데이터를 많이 수집하여, 다양한 유사도 방법으로 진행하고자 한다. 또한 웹사이트 배포도 이루어진다면 이번 프로젝트를 실질적으로 활용할 수 있다는 측면에서 더욱 좋을 것이다.

3. 기대 효과

이번 프로젝트를 통해 텍스트 유사도에 대한 테스트와 웹사이트 구축 등을 시도하였다. 다양한 방법에 따라 다양한 결과가 나온다는 것을 알 수 있었고 이번 결과를 국가법령정보센터의 서비스에 활용하여, 이용자에게 보다 직관적이고 알기 쉬운 서비스를 제공할 수 있을 것으로 기대한다.