인공지능수학. **Ⅱ. 텍스트 자료의 표현과 분류**	고등학교 학번, 이름:	

0. INTRO

수와 수학 기호는 자료를 효과적으로 표현할 수 있는 도구이고, 인공지능이 다루는 자료는 수학을 이용하여 표현된다. 이번 단원에서는 텍스트 자료를 벡터라는 수학 개념을 이용하여 표현하고 처리하는 방법을 배운다.

0.1 목차

1. 벡터

I단원에서 벡터에 대해서 배웠다. 벡터는 주로 텍스트 자료를 수학적으로 표현하는 수단으로 사용된고 하였다. 이번 단원에서는 텍스트 자료를 벡터로 표현한 후, 벡터를 이용하여 텍스트 자료를 처리하려고 한다.

복습

n개의 수 $x_1, x_2, \cdots, x_n$을 괄호 속에 순서대로 나열하여 나타낸

$$\overrightarrow{X} = (x_1, x_2, \cdots, x_n)$$

을 **벡터**라고 한다. 이 때, $x_1, x_2, \cdots, x_n$을 벡터 $\overrightarrow{X}$의 성분이라고 하고, x_k를 벡터 $\overrightarrow{X}$의 k번째 성분이라고 한다.

[문제1] 벡터 $\overrightarrow{A}=(1,0,2,3,1)$에서 다음을 구하시오.

(1) 벡터 $\overrightarrow{A}$의 둘째 성분 (2) 벡터 $\overrightarrow{A}$의 넷째 성분

2. 텍스트 자료의 표현

컴퓨터는 단어나 문장과 같은 텍스트 자료를 글자 형태로 인식하지 못하기 때문에 이를 수 또는 수학 기호를 이용하여 표현해야 한다. 성별이나 혈액형과 같은 간단한 정형 텍스트 자료는 쉽게 숫자로 바꿀 수 있지만, 댓글이나 감상평과 같은 비정형 텍스트 자료는 수학 기호로 바꾸어야 컴퓨터가 인식할 수 있다. 이제 비정형 텍스트 자료를 수학적으로 표현하는 방법을 알아보자.

2.1 텍스트 자료 준비

예를 들어, 아래 말뭉치는 학생 4명(a1, a2, a3, a4)의 간단한 자기소개글이다.

a1 : 나는 수학을 잘하고 과학을 좋아해.
a2 : 나도 과학을 좋아하고 수학도 좋아하고 즐겨 해.
a3 : 난 물리는 좋지만 화학은 싫어해. 물리 만세!
a4 : 나는 요리를 좋아해. 특히 한식을 좋아해.

인공지능을 이용하지 않더라도 대략적으로 수학, 과학 분야를 좋아하는 a1, a2가 서로 비슷한 성향임을, 그리고 a4는 나머지 셋과 동떨어진 요리 이야기를 하고 있음을 알 수 있다. 이제 텍스트 자료를 수학적으로 처리하여도 이와 비슷한 결과를 얻을 수 있을지 알아보자.

2.2 텍스트 자료 전처리

자료를 본격적으로 처리하기에 앞서, 각각의 낱말을 기본 사전형 단어로 바꾸고, 불용어(관형사, 조사, 접미사, 특수 문자 등 문장에서 의미가 없는 불필요한 요소)를 제거, 문장 부호를 제거하고 영어의 경우 대문자를 소문자로 변환하며, 단수, 복수를 통일시키는 등의 전처리 작업을 거치면 아래와 같이 된다. 편의상 집합을 이용하여 표현하자.

A_1 = {나, 수학, 잘한다, 과학, 좋아한다}
A_2 = {나, 과학, 좋아한다, 수학, 즐기다}
A_3 = {나, 만세, 물리, 좋아한다, 화학, 싫어한다}
A_4 = {나, 요리, 좋아한다, 특히, 한식}

2.3 단어사전 생성

이제 집합 A_1, A_2, A_3, A_4의 합집합을 U라 하면,

U = {과학, 나, 만세, 물리, 수학, 싫어한다, 요리, 잘한다, 좋아한다, 즐기다, 특히, 한식, 화학}

이다. 이 집합 U를 네 텍스트 데이터 a1, a2, a3, a4로부터 구성한 **단어사전**이라고 한다.

2.4 텍스트 데이터의 벡터화

이제 각 데이터 a1, a2, a3, a4를 벡터로 나타내려 한다. 다음 표는 앞의 단어사전 U에 있는 단어를 순서대로 나열하고 세 집합 A_1, A_2, A_3의 각 단어가 사전 U에 속하면 1, 속하지 않으면 0으로 나타낸 것이다.

	과학	나	만세	물리	수학	싫어한다	요리	잘한다	좋아한다	즐기다	특히	한식	화학
A1	1	1	0	0	1	0	0	1	1	0	0	0	0
A2	1	1	0	0	1	0	0	0	2	1	0	0	0
A3	0	1	1	2	0	1	0	0	1	0	0	0	1
A4													

[문제2] 위 표에서 A4에 해당하는 부분을 스스로 채우시오.

이 표에서 세 집합 A_1, A_2, A_3에 대응하는 수를 순서대로 나열하면

$$\overrightarrow{A_1} = (1,1,0,0,1,0,0,1,1,0,0,0,0)$$
$$\overrightarrow{A_2} = (1,1,0,0,1,0,0,0,2,1,0,0,0)$$
$$\overrightarrow{A_3} = (0,1,1,2,0,1,0,0,1,0,0,0,1)$$

와 같이 벡터로 나타낼 수 있으며 이것은 단어사전 U의 각 단어가 세 데이터 a1, a2, a3에 속하는지 여부를 보여 준다.

[문제3] 어떤 두 텍스트 데이터를 구성하는 단어를 모은 집합을 각각 A, B라 하고 단어사전 U가 아래와 같다고 하자. A, B를 벡터 $\overrightarrow{A}, \overrightarrow{B}$로 나타내어라.

U = {달콤, 맛있다, 많다, 맵다, 수제, 양, 재방문, 정성, 추천, 치즈, 카레}

A = {달콤, 치즈, 양, 많다, 재방문},
B = {카레, 맵다, 수제, 정성, 추천, 맛있다}

3. 텍스트 자료의 처리

인공지능은 많은 양의 비정형 텍스트 데이터에서 주제어나 선호도 등을 찾아내는 데 이용된다. 특히, 텍스트 데이터에서 특정 단어의 출현 빈도는 그 자료의 성격을 판단하는 중요한 기준이 되기 때문에 주제어나 선호도 판단에 중요하게 쓰인다.

3.1 단어 빈도수

이제 a1, a2, a3, a4에 대해 집합으로 나타낸 텍스트 자료에 출현하는 단어 빈도수(Term Frequency, TF)를 구하여 표로 나타내어 보자.

	과학	나	만세	물리	수학	싫어한다	요리	잘한다	좋아한다	즐기다	특히	한식	화학
A1	1	1	0	0	1	0	0	1	1	0	0	0	0
A2	1	1	0	0	1	0	0	0	2	1	0	0	0
A3	0	1	1	2	0	1	0	0	1	0	0	0	1
A4	0	1	0	0	0	0	1	0	2	0	1	1	0
U	2	4	1	2	2	1	1	1	6	1	1	1	1

이를 벡터로 나타내면 아래와 같다.

$$\overrightarrow{A_1} = (1,1,0,0,1,0,0,1,1,0,0,0,0)$$
$$\overrightarrow{A_2} = (1,1,0,0,1,0,0,0,2,1,0,0,0)$$
$$\overrightarrow{A_3} = (0,1,1,2,0,1,0,0,1,0,0,0,1)$$
$$\overrightarrow{A_4} = (0,1,0,0,0,0,1,0,2,0,1,1,0)$$
$$\overrightarrow{U} = (2,4,1,2,2,1,1,1,6,1,1,1,1)$$

이로부터 a2, a4 중에서 각각 빈도수가 가장 큰 단어인 '좋아한다', '물리'가 각 데이터 a2, a4의 주제어임을 알 수 있다. 또한 말뭉치 U에서 빈도수가 가장 큰 단어가 '좋아한다', '나'이므로 이 말뭉치의 주제어는 '좋아한다', '나'임을 알 수 있다.

이처럼 비정형 텍스트 데이터에서 **단어 빈도수(Term Frequency, TF)**를 구하면 그 자료의 주제어를 찾을 수 있다.

개념

말뭉치 속의 한 단어 a에 대해, a가 담긴 문장을 A라 할 때,
(A에서 a의 **단어 빈도수** TF) = (A에 등장하는 a의 개수)

[문제4] A = '학교에서는 공동체 역량, 지식 정보 처리 역량, 자기 관리 역량 등을 배운다'에서 '역량'의 단어빈도수를 구하여라.

3.2 문서 빈도수, 상대도수

한편, 단어가 등장하는 총 문장의 개수도 의미 있는 지표가 된다. 앞서 등장한 예시에서 비록 단어 빈도수가 2이지만 한 데이터에만 등장한 '물리'보다는 단어 빈도수가 1이지만 모든 데이터에 골고루 등장한 '나'가 어떤 의미에서는 더 유용한 단어가 될 수도 있다.

이와 같이, 단어가 등장하는 총 문장의 개수를 **문서 빈도수(Document Frequency, DF)**라고 한다.

개념

말뭉치 속의 한 단어 a에 대해,
(a의 **문서 빈도수** DF) = (a가 등장하는 문장의 개수)

	과학	나	만세	물리	수학	싫어한다	요리	잘한다	좋아한다	즐기다	특히	한식	화학
a1	1	1	0	0	1	0	0	1	1	0	0	0	0
a2	1	1	0	0	1	0	0	0	2	1	0	0	0
a3	0	1	1	2	0	1	0	0	1	0	0	0	1
a4	0	1	0	0	0	0	1	0	2	0	1	1	0
u	2	4	1	2	2	1	1	1	6	1	1	1	1

예를 들어 이 예시에서 '과학'은 문장 a1, a2에 등장하므로 '과학'의 문서 빈도수 DF는 2이다.

한편, **DF가 높을수록 해당 단어가 많은 문서에 공통적으로 나타났다는 뜻이며 보편적인 단어라는 뜻**이다. 예를 들어, '과학', '나'의 경우, '나'의 문서 빈도수가 높지만 모든 데이터에 속하여 역설적으로 특수성이 사라지고, '과학'의 경우 a1, a2에는 속하고 a3, a4에는 속하지 않음으로서 a1, a2를 유의미하게 설명해주는 역할을 가진다는 것을 알 수 있다.

단어가 말뭉치 전체에 고루 등장하는 정도를 0과 1 사이의 숫자로 나타내려 한다. 이는 해당 단어의 문서 빈도수(DF)에 비례할 것이며,

$$0 < \text{문서 빈도수(DF)} \leq \text{(전체 문장의 개수)}$$

이므로, 단어가 말뭉치 전체에 고루 등장하는 정도는 DF/(전체 문장의 개수)로 정의하면 좋을 것이다. 이를 해당 단어의 **상대도수(Relative Frequency, RF)**라 한다.

개념

어떤 말뭉치의 전체 문장의 개수를 n, 그리고 a를 말뭉치 내의 한 단어라 할 때,
(단어 a의 **상대도수**) = (a가 등장하는 총 문장 개수) / (전체 문장의 개수) = **DF / n**

예를 들어, 위 예시에서 단어 '나', '수학', '물리'의 상대도수는 각각 아래와 같다.

('나'의 상대도수) = ('나'가 등장하는 총 문장 개수) / (전체 문장 개수) = 4/4 = 1
('수학'의 상대도수) = ('수학'이 등장하는 총 문장 개수) / (전체 문장 개수) = 2/4 = 0.5
('물리'의 상대도수) = ('물리'가 등장하는 총 문장 개수) / (전체 문장 개수) = 1/4 = 0.25

따라서, 상대도수가 높은 '나'가 '수학'보다, '수학'이 '물리'보다 말뭉치에 조금 더 고루 등장하는 단어라 할 수 있다.

[문제5] 단어사전 U = {독일, 바게트, 소시지, 식도락, 여행, 프랑스}에 대해 문장 A, B와 단어사전 U의 벡터가 아래와 같다고 하자.

A = (1, 0, 1, 0, 1, 0)
B = (0, 1, 0, 2, 1, 1)
U = (1, 1, 1, 2, 2, 1)

'식도락', '여행'의 단어빈도수(TF), 문서빈도수(DF), 상대도수(RF)를 각각 구하고 그 의미를 정리하시오.

3.3 역문서 빈도수

상대도수가 높다는 것은 무엇을 의미할까? 위 예시에서 상대도수가 1인 단어 '나'는 모든 문장에 등장하는 가장 보편적인 단어 중 하나다. 반대로, 상대도수가 작은 단어는 그 단어를 포함하는 문장을 특별하게 해주는 역할을 한다. 예를 들어, 가장 낮은 상대도수 0.25를 갖는 단어 중 하나인 '물리'는 a3 문장만이 갖는 특징이 된다. 즉, 상대도수가 낮은 단어일수록 문장을 특별하게 하는 의미를 갖는다.

따라서, 데이터를 처리할 때, 상대도수보다는 상대도수의 역수가 중요할 것이다. 이를 **역문서 빈도수 (Inverse Document Frequency, IDF)**라 한다.

개념

a가 말뭉치 속의 한 단어일 때,
(a의 **역문서 빈도수** IDF) = 1 / (a의 상대도수) = (전체 문장 개수) / (a의 DF)

3.4 TF x IDF

역문서 빈도수(IDF)에서 설명하였듯, 일반적으로 모든 문장에서 자주 출현하는 단어는 중요도가 낮다고 판단하고, 특정한 문장에서만 자주 출현하는 단어는 중요도가 높다고 판단한다. 즉, 단어의 중요도는 TF, IDF에 비례한다. 그러면 **TF와 IDF의 곱**을 단어의 중요도라고 정의할 수 있을 것이다.

개념

a가 말뭉치 내의 단어, 문장 A에서의 a의 단어 빈도수를 TF, a의 역문서 빈도수를 IDF라 할 때,
TF x IDF 는 일반적으로 문장 A에서의 단어 a의 중요도를 말한다.

[문제6] 아래 표에 해당하는 말뭉치와 문장 a1, a2, a3, a4에서 물음에 답하시오.

	과학	나	만세	물리	수학	싫어한다	요리	잘한다	좋아한다	즐기다	특히	한식	화학
a1	1	1	0	0	1	0	0	1	1	0	0	0	0
a2	1	1	0	0	1	0	0	0	2	1	0	0	0
a3	0	1	1	2	0	1	0	0	1	0	0	0	1
a4	0	1	0	0	0	0	1	0	2	0	1	1	0
u	2	4	1	2	2	1	1	1	6	1	1	1	1

(1) a1에서 '나'의 TFxIDF는?

(2) a1에서 '수학'의 TFxIDF는?

(3) a1에서 '나', '수학' 중 중요도가 높은 단어는 무엇이라고 할 수 있을까?

한편, 인공지능이 다루는 문서에서 전체 문장의 개수 n은 일반적으로 매우 큰 수이므로, IDF = n/DF 역시 매우 큰 수가 되어 계산하기 불편하다. 따라서, IDF를 작은 값으로 만들기 위해 실제 인공지능에서는 로그를 이용하여,

$$\mathrm{IDF} = \log(\mathrm{n}/\mathrm{DF})$$

라고 정의한다.

4. 텍스트 자료의 표현 및 처리 실습

이번에는 데이터의 규모를 조금 더 키워서 파이썬 코딩의 힘을 빌려 텍스트 자료를 분석해보려 한다.

4.1 데이터 준비

여기서는 역시 4명의 학생 a1, a2, a3, a4의 자기소개글을 이용하되, 문장성분 간 띄어쓰기가 되어 있지 않은 한글에 비해 다루기 쉬운 영어 텍스트 자료를 이용하도록 한다.
(코드 가져오기 : https://cha-record.studio/인공지능수학-2단원-실습코드/)

In
```
a1 = 'I like math. I like science as well, especially physics.'
a2 = 'I prefer math and geometry to social science. And I don't like sports.'
a3 = 'I'm afraid but I don't like math nor science.'
a4 = 'I like cooking and my favorite food is Korean food.'
```

그리고 데이터 처리에 앞서 문장부호를 없애고 대문자를 소문자로 통일하는 전처리 작업을 하도록 한다.

In
```
data_list = [a1, a2, a3, a4]          # 각 데이터를 리스트 data_list에 담는다
mining_list = []                      # 전처리 작업을 거친 데이터를 담을 빈 리스트 준비
for data in data_list :               # data_list의 각 데이터 data에 대해 다음을 반복 수행
    data = data.replace(" ' ", "")    # 따옴표 제거
    data = data.replace(",","")       # 콤마 제거
    data = data.replace(".", "")      # 온점 제거
    data = data.lower()               # 대문자를 모두 소문자로 변환
    mining_list.append(data)          # mining_list에 전처리과저을 거친 data 담음

for new_data in mining_list :         # 전처리를 거친 모든 데이터를 출력
    print(new_data)
```

Out
```
i like math i like science as well especially physics
i prefer math and geometry to social science and i dont like sports
im afraid but i dont like math nor science
i like cooking and my favorite food is korean food
```

깔끔히 문장부호가 사라지고 모두 소문자로 변환되었다.

4.2 말뭉치 생성

이제 4개의 데이터를 모두 담은 말뭉치를 만드려고 한다. a1, a2, a3, a4를 모두 담은 리스트 corpus를 생성해 보자.

In

```
corpus = data_list    # data_list를 변수 corpus에 재할당
corpus
```

Out

```
['i like math i like science as well especially physics',
 'i prefer math and geometry to social science and i dont like sports',
 'im afraid but i dont like math nor science',
 'i like cooking and my favorite food is korean food']
```

4.3 벡터화

이제 각 데이터를 벡터로 바꾸자. 그러기 위해 단어사전을 만들자. 다행히 파이썬 패키지 scikit-learn에서는 번거로운 과정 없이 바로 단어사전을 생성하여 주므로 이를 이용해 보자.

In

```
from sklearn.feature_extraction.text import CountVectorizer        # 패키지 임포트

vect = CountVectorizer()        # 클래스 인스턴스 생성 및 CounterVectorizer() 함수를 vect에 할당
vect.fit(corpus)                  # corpus를 전체집합으로 하여 단어사전 생성

vect_dict = vect.vocabulary_    # 생성된 단어사전 vect.vocabulary_를 vect_dict라 함.

word_list = []                   # 단어사전 속 단어를 담을 리스트 생성
index_list = []                 # 단어사전 속 단어들을 넘버링하고 그 번호를 담을 리스트 생성

for tup in sorted(vect_dict.items()) :  # vect_dict.items()의 각 튜플 tup에 대해
                    word_list.append(tup[0])  # 튜플의 첫 번째 성분인 단어 뽑아내어 word_list에.
                    index_list.append(tup[1]) # 튜플의 두 번째 성분인 번호 뽑아내어 index_list에.
print(sorted(vect_dict.items()), end=' ')    # 보기좋게 출력
```

Out

```
[('afraid', 0), ('and', 1), ('as', 2), ('but', 3), ('cooking', 4), ('don', 5), ('dont', 6), ('especially', 7), ('favorite', 8), ('food', 9), ('geometry', 10), ('is', 11), ('korean', 12), ('like', 13), ('math', 14), ('my', 15), ('nor', 16), ('physics', 17), ('prefer', 18), ('science', 19), ('social', 20), ('sports', 21), ('to', 22), ('well', 23)]
```

이와 같이 단어사전이 생성되었다.

이제 이 단어사전을 기반으로 각 문장을 벡터로 변환하자.

In

```
vectors = []                                    # 변환된 벡터를 담을 리스트 준비

u = a1 + ' ' + a2 + ' ' + a3 + ' ' + a4         # 전체 문장 u 생성 (a1, a2, a3, a4를 이은 문장)

for i in corpus :                               # 각 문장에 대해 다음을 반복 수행
    a = vect.transform([i]).toarray()           # 단어사전을 기반으로 벡터로 자동변환된 것을 a라 함.
    vectors.append(a)                           # 변환된 벡터 a를 vectors 리스트에 담음.

u_ = vect.transform([u]).toarray()              # 전체문장 u도 벡터로 변환
vectors.append(u_)

namelist = ['A1', 'A2', 'A3', 'A4', 'U' ]       # 출력 때 보기좋게 문자열 생성
print('---------- 단어 사전 ----------')
print(sorted(vect_dict.items()), end = ' ')     # 단어사전 출력
print("""

""")
print('---------- 벡터 ----------')
for name, vector in zip(namelist, vectors) :    # 변환된 벡터와 그 기호를 짝지어 출력
    print(name, ':', vector)
```

Out

```
---------- 단어 사전 ----------
[('afraid', 0), ('and', 1), ('as', 2), ('but', 3), ('cooking', 4), ('don', 5), ('dont', 6), ('especially', 7), ('favorite', 8), ('food', 9), ('geometry', 10), ('is', 11), ('korean', 12), ('like', 13), ('math', 14), ('my', 15), ('nor', 16), ('physics', 17), ('prefer', 18), ('science', 19), ('social', 20), ('sports', 21), ('to', 22), ('well', 23)]


---------- 벡터 ----------
A1 : [[0 0 1 0 0 0 0 1 0 0 0 0 0 2 1 0 0 1 0 1 0 0 0 1]]
A2 : [[0 2 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 1 1 1 1 1 0]]
A3 : [[1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0]]
A4 : [[0 1 0 0 1 0 0 0 1 2 0 1 1 1 0 1 0 0 0 0 0 0 0 0]]
U : [[1 3 1 1 1 1 1 1 1 2 1 1 1 5 3 1 1 1 1 3 1 1 1 1]]
```

가독성을 위해 표를 이용해 표현하면 아래와 같다.

In

```
import pandas as pd                    # 표 생성을 위한 pandas 라이브러리 임포트

# 표의 헤더를 word, A1, A2, A3, A4, U로 하고 각 단어의 단어 빈도수를 내용으로 하는 표 생성
con_dict = {
    'word' : word_list,
    'A1' : vectors[0][0],
    'A2' : vectors[1][0],
    'A3' : vectors[2][0],
    'A4' : vectors[3][0],
    'U' : vectors[4][0]
}
a = pd.DataFrame(con_dict)

a
```

Out

	word	A1	A2	A3	A4	U
0	afraid	0	0	1	0	1
1	and	0	2	0	1	3
2	as	1	0	0	0	1
3	but	0	0	1	0	1
4	cooking	0	0	0	1	1
5	don	0	0	1	0	1
6	dont	0	1	0	0	1
7	especially	1	0	0	0	1
8	favorite	0	0	0	1	1
9	food	0	0	0	2	2
10	geometry	0	1	0	0	1
11	is	0	0	0	1	1
12	korean	0	0	0	1	1
13	like	2	1	1	1	5
14	math	1	1	1	0	3
15	my	0	0	0	1	1
16	nor	0	0	1	0	1
17	physics	1	0	0	0	1
18	prefer	0	1	0	0	1
19	science	1	1	1	0	3
20	social	0	1	0	0	1
21	sports	0	1	0	0	1
22	to	0	1	0	0	1
23	well	1	0	0	0	1

Note. 각 벡터의 성분이 곧 해당 단어의 단어 빈도수(TF, Term Frequency)임을 잊지 말자.

4.4 문서 빈도수(DF) 출력

이제 텍스트 데이터에서 각 단어의 문서 빈도수(DF)를 출력해 보자.

In

```
df_list = []     # 문서 빈도수 담을 리스트 출력
for k in range(0, len(word_list)) :    # 각 단어마다 그 문서빈도수를 리스트 df_list에 담음
    df_list.append(len(vectors)-1)     # vectors에는 U도 있으므로 이를 제외하기 위해 1을 뺌.

for i in range(0, len(vectors[0][0])) :
    for vector in vectors :           # 각 벡터에서 아래를 반복 수행
        if vector[0][i] == 0 :
            df_list[i] -= 1           # 해당 단어가 든 데이터가 존재할 때마다 1씩 뺌.

print(df_list)
```

Out

```
[1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 3, 1, 1, 1, 1, 3, 1, 1, 1, 1]
```

4.5 상대도수 출력

이제 각 단어들의 상대도수를 구해 리스트 relative_freq_list에 담아보자.

In

```
# 문장 개수를 n에 할당. vectors에는 문장 벡터 뿐 아니라 전체문장 u도 있으므로 1빼야 문장개수가 됨.
n = len(vectors) - 1

relative_freq_list = [ df/n for df in df_list ]        # 상대도수 df/n을 담은 리스트

relative_freq_dict = {
    'word' : word_list,
    'Relative_Freq' : relative_freq_list
}

temp = pd.DataFrame(relative_freq_dict)              # 표로 출력
temp
```

Out

	word	Relative_Freq
0	afraid	0.25
1	and	0.50
2	as	0.25
3	but	0.25
4	cooking	0.25

5	don	0.25
6	dont	0.25
7	especially	0.25
8	favorite	0.25
9	food	0.25
10	geometry	0.25
11	is	0.25
12	korean	0.25
13	like	1.00
14	math	0.75
15	my	0.25
16	nor	0.25
17	physics	0.25
18	prefer	0.25
19	science	0.75
20	social	0.25
21	sports	0.25
22	to	0.25
23	well	0.25

실제로 모든 문장이 등장하는 'like'의 상대도수가 1, 한 문장에만 등장하는 단어 'food'의 상대도수가 0.25 임을 확인할 수 있다.

4.6 역문서 빈도수(IDF) 출력

다음은 역문서 빈도수를 출력하는 과정이다.

In

```
idf_list = [ 1/relative_freq for relative_freq in relative_freq_list]

idf_dict = {
    'word' : word_list,
    'Relative_Freq' : relative_freq_list,
    'Inverse DF' : idf_list
}

temp = pd.DataFrame(idf_dict)
temp
```

Out

	word	Relative_Freq	Inverse DF
0	afraid	0.25	4.000000
1	and	0.50	2.000000
2	as	0.25	4.000000
3	but	0.25	4.000000

4	cooking	0.25	4.000000
5	don	0.25	4.000000
6	dont	0.25	4.000000
7	especially	0.25	4.000000
8	favorite	0.25	4.000000
9	food	0.25	4.000000
10	geometry	0.25	4.000000
11	is	0.25	4.000000
12	korean	0.25	4.000000
13	like	1.00	1.000000
14	math	0.75	1.333333
15	my	0.25	4.000000
16	nor	0.25	4.000000
17	physics	0.25	4.000000
18	prefer	0.25	4.000000
19	science	0.75	1.333333
20	social	0.25	4.000000
21	sports	0.25	4.000000
22	to	0.25	4.000000
23	well	0.25	4.000000

4.7 TF x IDF 출력

비슷한 방식으로 문장 a1에서의 TFxIDF 까지 함께 출력하면 아래와 같다.

In

```
tfidf_list = [tf*idf for tf, idf in zip(vectors[0][0], idf_list)]

tfidf_dict = {
    'word' : word_list,
    'TF' : vectors[0][0],
    'IDF' : idf_list,
    'TFxIDF' : tfidf_list
}

temp = pd.DataFrame(tfidf_dict)

temp
```

Out

	word	TF	IDF	TFxIDF
0	afraid	0	4.000000	0.000000
1	and	0	2.000000	0.000000
2	as	1	4.000000	4.000000
3	but	0	4.000000	0.000000

4	cooking	0	4.000000	0.000000
5	don	0	4.000000	0.000000
6	dont	0	4.000000	0.000000
7	especially	1	4.000000	4.000000
8	favorite	0	4.000000	0.000000
9	food	0	4.000000	0.000000
10	geometry	0	4.000000	0.000000
11	is	0	4.000000	0.000000
12	korean	0	4.000000	0.000000
13	like	2	1.000000	2.000000
14	math	1	1.333333	1.333333
15	my	0	4.000000	0.000000
16	nor	0	4.000000	0.000000
17	physics	1	4.000000	4.000000
18	prefer	0	4.000000	0.000000
19	science	1	1.333333	1.333333
20	social	0	4.000000	0.000000
21	sports	0	4.000000	0.000000
22	to	0	4.000000	0.000000
23	well	1	4.000000	4.000000

이를 통해 **문장 a1에서 TFxIDF가 가장 높은 as, especially, physics, well이 가장 중요도가 높은 단어라 할 수 있다.** 한 발 더 나아가, as, especially, well은 문장을 수식하는 역할에 불과하다는 것을 고려할 때 정말 중요도가 높은 단어는 physics로서, 실제로 물리를 좋아한다고 응답한 학생은 a1밖에 없다는 것을 생각하면 어느 정도 유의미한 결과를 얻었다고 할 수 있겠다.

[문제7] 문장 a4에서 가장 중요도가 높은 단어는 무엇이라고 할 수 있을지 미리 예상해 보자. 그리고 코드를 이용하여 a4에서 실제로 중요도가 가장 높은 단어를 추출해 보고 자신의 예상과 비교해 보자.
(힌트 : 15쪽의 tfidf_dict 딕셔너리의 키 'TF'의 값을 어떻게 바꾸면 될까?)

5. 인공지능의 텍스트 분류 1 - 여러 가지 유사도

5.1 유사도

인공지능을 이용하여 텍스트 자료를 분석하기 위해서는 텍스트들 사이의 유사한 정도를 수학적으로 표현해야 한다. 두 텍스트 사이의 유사한 정도를 수학적인 방법으로 수치화한 것을 유사도라고 하는데 유클리드 유사도, 코사인 유사도, 자카드 유사도 등이 많이 사용된다.

5.2 유클리드 유사도

그 중 유클리드 유사도는 아래와 같이 정의한다.

개념

두 텍스트 P와 Q를 나타내는 벡터를 각각

$$P = (p_1, p_2, \cdots, p_n),\ Q = (q_1, q_2, \cdots, q_n)$$

라 할 때, P와 Q의 **유클리드 유사도**를 기호로 $L(P,Q)$와 같이 나타내고,

$L(P,Q) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2}$ 과 같이 정한다.

[문제8] 두 텍스트 P와 Q를 나타내는 벡터가 각각 P=(4,1), Q=(1,5)일 때, P와 Q의 유클리드 유사도 L(P,Q)를 구하여라.

두 텍스트의 유클리드 유사도는 텍스트를 나타내는 벡터의 성분에 의해 결정되는 두 점 사이의 거리와 같다. 그러므로 두 점 사이의 거리가 가까울수록 유클리드 유사도의 값이 작아진다.

따라서, **유클리드 유사도의 값이 0에 가까울수록 두 텍스트가 유사하다고 판단**한다.

[문제9] 세 텍스트 P, Q, R를 나타내는 벡터가 다음과 같을 때, 유클리드 유사도를 이용하여 가장 유사한 텍스트 두 개를 고르시오.

P = (1,0,1), Q = (0,1,2), R = (1,1,0)

5.3 코사인 유사도

코사인 유사도는 아래와 같이 정의한다.

개념

두 텍스트 P와 Q를 나타내는 벡터를 각각

$$P = (p_1, p_2, \cdots, p_n),\ Q = (q_1, q_2, \cdots, q_n)$$

일 때, P와 Q의 **코사인 유사도**를 기호로 $C(P,Q)$와 같이 나타내고,

$$C(P,Q) = \frac{p_1q_1+p_2q_2+\cdots+p_nq_n}{\sqrt{p_1^2+p_2^2+\cdots+p_n^2}\sqrt{q_1^2+q_2^2+\cdots+q_n^2}}$$ 과 같이 정한다.

[문제10] 두 텍스트 P와 Q를 나타내는 벡터가 각각 P=(2,1), Q=(1,3)일 때, P와 Q의 코사인 유사도 C(P,Q)를 구하여라.

두 텍스트 P와 Q를 나타내는 벡터 $\overrightarrow{P}, \overrightarrow{Q}$가 이루는 각의 크기를 $\theta\,(0^\circ \le \theta \le 90^\circ)$라 할 때, 두 텍스트 P, Q의 코사인 유사도 C(P,Q)는 $\cos\theta^\circ$와 같음이 알려져 있다.
이와 같은 이유에서 C(P,Q)를 코사인 유사도라고 부른다. 이 때, $0 \le C(P,Q) \le 1$이다.

두 벡터가 이루는 각의 크기가 작을수록 두 텍스트가 유사하다고 할 수 있으므로 **코사인 유사도의 값이 1에 가까울수록 두 텍스트가 유사하다고 판단**한다.

[문제11] 세 텍스트 P,Q,R를 나타내는 벡터가 다음과 같을 때, 코사인 유사도를 이용하여 가장 유사한 텍스트 두 개를 고르시오.

P = (1,1,0), Q = (2,0,0), R = (1,0,2)

5.4 자카드 유사도

자카드 유사도는 아래와 같이 정의한다.

개념

두 텍스트 P와 Q를 나타내는 벡터를 각각

$$P = (p_1, p_2, \cdots, p_n),\ Q = (q_1, q_2, \cdots, q_n)$$

일 때, P와 Q의 **자카드 유사도**를 기호로 $J(P,Q)$와 같이 나타내고,

$$J(P,Q) = \frac{n(P \cap Q)}{n(P \cup Q)} = \frac{n(P \cap Q)}{n(P)+n(Q)-n(P \cap Q)}$$ 과 같이 정한다.

자카드 유사도는 0 이상 1 이하의 값을 가진다. 두 집합에 공통으로 포함되는 원소가 많을수록 두 텍스트가 유사하므로 **자카드 유사도의 값이 1에 가까울수록 두 텍스트가 유사하다고 판단**한다.

[문제12] 세 텍스트 P,Q,R를 구성하는 단어의 집합이 다음과 같을 때, 자카드 유사도를 이용하여 가장 유사한 텍스트 두 개를 고르시오.

P = {빨강, 노랑, 파랑}, Q = {빨강, 초록, 보라}, R = {초록, 파랑, 보라}

6. 인공지능의 텍스트 분류 2 - 유사도 분석을 이용한 텍스트 분류

6.1 상호 유사도 분석

텍스트 자료 전체를 특성이 비슷한 자료들끼리 몇 개의 그룹으로 묶는 방법을 유사도 분석이라고 한다.
여기서는 코사인 유사도를 이용하여 유사도 분석을 하는 방법을 알아보자.

다음과 같이 5개의 텍스트로 이루어진 말뭉치와 단어사전이 있다.

t1 : 사과를 좋아하고 바나나도 좋아한다.
t2 : 사과를 좋아하고, 바나나는 사과보다 더 좋아한다.
t3 : 사과는 좋아하지만, 바나나는 싫어한다.
t4 : 사과와 바나나를 모두 싫어한다.
t5 : 사과를 싫어하고 바나나도 싫어한다.

단어사전 U = {바나나, 사과, 싫다, 좋다}

[문제13] t1, t2, t3, t4, t5를 단어사전 U를 이용해 벡터 $\overrightarrow{T_1}, \overrightarrow{T_2}, \overrightarrow{T_3}, \overrightarrow{T_4}, \overrightarrow{T_5}$로 만들어라.

여기서 두 텍스트 t1, t2의 코사인 유사도 C(t1, t2)를 계산하여 소수점 아래 셋째 자리에서 반올림하면,

$$\text{C(t1, t2)} = \frac{1\times2+1\times1+2\times2+0\times0}{\sqrt{1^2+1^2+2^2+0^2}\sqrt{2^2+1^2+2^2+0^2}} = \frac{7\sqrt{6}}{18} = 0.95$$

와 같다. 같은 방법으로 모든 텍스트들 사이의 코사인 유사도를 구하여 다음과 같이 상호 유사도 행렬을 만들 수 있다.

	t1	t2	t3	t4	t5
t1	1	0.95	0.82	0.47	0.33
t2		1	0.83	0.58	0.41
t3			1	0.87	
t4					
t5					

[문제14] 위 상호 유사도 행렬의 나머지 빈 칸을 채우시오.

[문제15] 위 상호 유사도 행렬을 이용하여 텍스트 t1, t2, t3, t4, t5 중 가장 유사한 두 텍스트를 고르고 그 이유를 설명하시오.

6.2 분류

이제 이 표에서 유사도가 0.95로 가장 큰 t1, t2를 묶어서 g1라 놓고, 이와 비슷하게 큰 유사도 0.94를 갖는 t4, t5를 묶어서 g2라 한 뒤, g1, g2, t3의 상호 유사도 표를 만들어 보자.

이 때, C(g1, t3)은 C(t1, t3), C(t2, t3)중 큰 값으로 정한다. 즉,

$$C(t1, t3) = 0.82, \quad C(t2, t3) = 0.83$$

이므로, C(g1, t3) = 0.83 이다.

[문제16] g1, g2, t3의 상호 유사도 행렬을 완성하시오.

	g1	g2	t3
g1	1		0.83
g2		1	
t3	0.83		1

마지막으로, 이 표에서 유사도가 가장 큰 g2, t3을 묶어서 g3으로 놓으면 처음 5개의 텍스트 자료를 다음과 같이 2개의 유사 그룹으로 분류할 수 있다.

g1	g3
t1 : 사과를 좋아하고 바나나도 좋아한다. t2 : 사과를 좋아하고 바나나는 사과보다 더 좋아한다.	t3 : 사과는 좋아하지만, 바나나는 싫어한다. t4 : 사과와 바나나를 모두 싫어한다. t5 : 사과를 싫어하고, 바나나도 싫어한다.

이처럼 유사도 분석을 이용하여 주어진 텍스트 자료를 몇 개의 그룹으로 분류할 수 있고, 그 결과를 분석하여 의미 있는 정보를 얻을 수 있다.

[문제17] 2쪽의 자기소개글 a1, a2, a3, a4에 대하여 코사인 유사도를 이용하여 서로 유사한 텍스트끼리 두 그룹으로 분류하고, 그 결과를 최초의 추측과 비교해 보시오.

7. 인공지능의 텍스트 분류 실습

7.1 유클리드 유사도 분석 및 시각화

교재 9쪽~16쪽의 텍스트 자료를 그대로 갖고 유클리드 유사도 분석을 해 보려고 한다. 먼저 아래 코드는 종전과 같다. 헷갈리면 9~16쪽의 실습에서 그대로 이어서 진행하면 된다.

In (코드 가져오기 : https://cha-record.studio/인공지능수학-2단원-실습코드/)

```
import pandas as pd

a1 = 'I like math. I like science as well, especially physics.'
a2 = "I prefer math and geometry to social science. And I don't like sports."
a3 = "I'm afraid but I don't like math nor science."
a4 = 'I like cooking and my favorite food is Korean food.'

data_list = [a1, a2, a3, a4]
mining_list = []
for data in data_list :
    data = data.replace("'", "")
    data = data.replace(",", "")
    data = data.replace(".", "")
    data = data.lower()
    mining_list.append(data)

corpus = [a1, a2, a3, a4] = [new_data for new_data in mining_list]

from sklearn.feature_extraction.text import CountVectorizer

vect = CountVectorizer()
vect.fit(corpus)

vect_dict = vect.vocabulary_

word_list = []
index_list = []

vectors = []

u = a1 + ' '+ a2 + ' '+ a3 + ' '+ a4

for i in corpus :
    a = vect.transform([i]).toarray()
    vectors.append(a)

vectors
```

Out

```
[array([[0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 1, 0, 1, 0,
         0, 0, 1]], dtype=int64),
 array([[0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
         1, 1, 0]], dtype=int64),
 array([[1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
         0, 0, 0]], dtype=int64),
 array([[0, 1, 0, 0, 1, 0, 0, 0, 1, 2, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
         0, 0, 0]], dtype=int64)]
```

벡터를 담은 리스트에서 array([])와 같은 불필요한 기호 표현을 삭제하고 리스트를 조금 더 간단히 하자.

In

```
vectors_list = []
for vector_array in vectors :
    vectors_list.append(list(vector_array[0]))

vectors_list.pop()    # 23쪽 코드가 아니라 9~16쪽 코드를 그대로 이용하는 경우 이 명령을 추가

[v1,v2,v3,v4] = vectors_list

print(v1,v2,v3,v4)
```

Out

```
[0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1]
[0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0]
[1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 1, 0, 0, 0, 1, 2, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
```

이제 군더더기 없이 벡터들이 출력되었다. 유클리드 유사도를 함수로 만들자.

In

```
import math

# Euclidean Similarity e(p,q)
def e(p,q) :
    val = 0
    for k in range(len(p)):
        val += (p[k]-q[k])*(p[k]-q[k])
    val = math.sqrt(val)
    return round(val,2)
```

이렇게 만든 유클리드 유사도 함수를 이용해 시범삼아 벡터 (1,0), (0,1)의 유클리드 유사도와 v1, v2의 유클리드 유사도를 출력해 보자.

In

```
print(e([1,0],[0,1]))
e(v1, v2)
```

Out

```
1.41
3.87
```

정상적으로 출력됨을 확인할 수 있다.

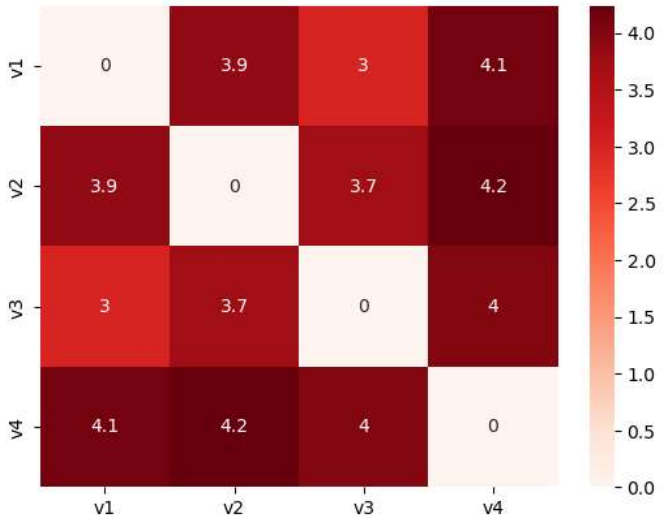
이제 유클리드 유사도를 이용한 상호 유사도 행렬을 출력하고 이를 히트맵으로 시각화해 보자.

In

```
import seaborn as sns
import matplotlib.pyplot as plt

euclidean_m = []
temp = []

for a in vectors_list :
    for b in vectors_list :
        temp.append(e(a,b))
    euclidean_m.append(temp)
    temp = []

print('Euclidean Similarity Table :')
for k in euclidean_m:
    print(k)

labels = ['v1', 'v2', 'v3', 'v4']

df = sns.heatmap(euclidean_m, cmap='Reds', annot = True,
                 xticklabels = labels, yticklabels = labels)

plt.show()
```

Out

Euclidean Similarity Table :
[0.0, 3.87, 3.0, 4.12]
[3.87, 0.0, 3.74, 4.24]
[3.0, 3.74, 0.0, 4.0]
[4.12, 4.24, 4.0, 0.0]



이 결과에 따르면 각 자료들은 뚜렷하게 비슷한 것도, 뚜렷하게 다른 것도 없지만 그나마 벡터 v1, v3이 유사하다는 결론을 얻을 수 있다. 실제 텍스트 자료 a1, a3을 고려할 때, 이 결과를 어떻게 받아들일 수 있을지 생각해 보자.

7.2 코사인 유사도 분석 및 시각화

비슷한 방식으로 코사인 유사도 분석을 해 보자. 코사인 유사도 함수는 아래와 같이 만들 수 있다.

In

```
# Cosine Similarity c(p,q)
def c(p,q) :
    val = 0
    upp = 0
    down_1 = 0
    down_2 = 0
    for k in range(len(p)) :
        upp += p[k]*q[k]
        down_1 += p[k]*p[k]
        down_2 += q[k]*q[k]
    down_1 = math.sqrt(down_1)
    down_2 = math.sqrt(down_2)
    val += upp
    val /= down_1*down_2
    return round(val,2)
```

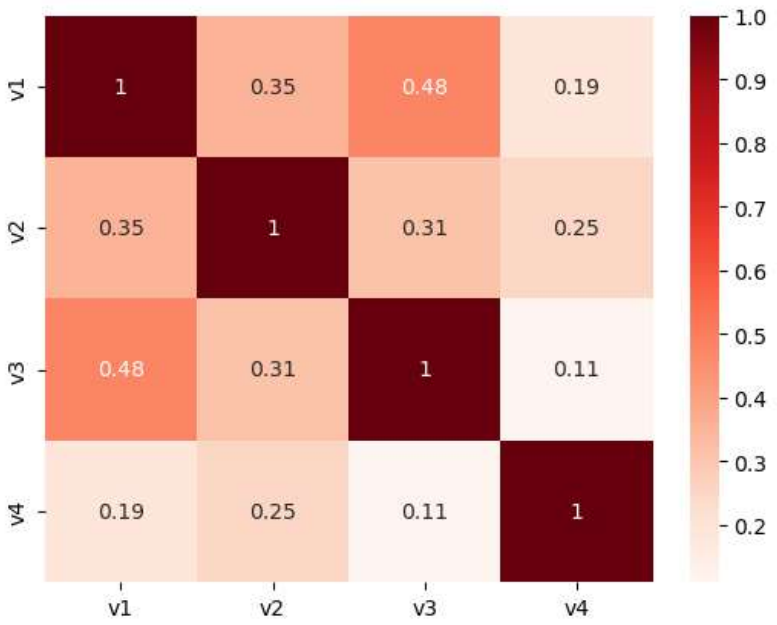
상호 유사도 행렬과 그 시각화는 아래와 같다.

In

```
cosine_m = []
temp = []


for a in vectors_list :
    for b in vectors_list :
        temp.append(c(a,b))
    cosine_m.append(temp)
    temp = []

print('Cosine Similarity Table :')
for k in cosine_m:
    print(k)

labels = ['v1', 'v2', 'v3', 'v4']

df = sns.heatmap(cosine_m, cmap='Reds', annot = True,
                 xticklabels = labels, yticklabels = labels)

plt.show()
```

Out

Cosine Similarity Table :
[1.0, 0.35, 0.48, 0.19]
[0.35, 1.0, 0.31, 0.25]
[0.48, 0.31, 1.0, 0.11]
[0.19, 0.25, 0.11, 1.0]



역시 그나마 a1, a3이 비슷하다는 결론을 얻었다. a1은 수학, 과학을 좋아하는 학생이고, a3은 수학, 과학을 싫어하는 학생임을 감안할 때 아쉬운 결과다. 이처럼 유사도에만 의존한 텍스트 분석은 문맥을 고려하지 못하고 빈도수에 의존하는 분석으로서 분명한 한계를 갖는다. Ⅳ단원에서는 이러한 한계를 극복하고 실제 인간의 지능과 비슷하게 추론하는 인공지능을 만드는 데 쓰이는 수학적 원리를 배우게 된다.