NAMAN THAKER

20BCE529

IRS PRACTICAL 8 META SEARCH

```
import nltk
import regex
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import euclidean_distances
from sklearn import preprocessing
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
Courpus1 = {'The Punisher is a fictional antihero appearing in American comic books publis
           'The character was created by writer Gerry Conway and artists John Romita Sr. a
           'The character is depicted as an Italian-American vigilante who employs murder,
           'Driven by the deaths of his wife and two children, who were killed by the mob
           'the Punisher wages a one-man war on crime using various weapons.',}
```

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(Courpus1)
print(vectorizer.get_feature_names())
print(X.toarray())
```

```
  0.         0.         0.         0.         0.26611804 0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.26611804 0.26611804 0.26611804 0.12680667 0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.26611804 0.         0.         0.
  0.         0.         0.26611804 0.        ]
 [0.         0.24241232 0.         0.         0.         0.30046383
  0.30046383 0.         0.         0.30046383 0.20122413 0.
  0.         0.         0.         0.         0.         0.30046383
  0.30046383 0.         0.         0.         0.         0.
  0.         0.         0.         0.30046383 0.         0.
  0.         0.20122413 0.24241232 0.         0.         0.
  0.         0.         0.         0.30046383 0.         0.
  0.         0.         0.         0.         0.         0.30046383
  0.24241232 0.         0.         0.         0.14317262 0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.2198971  0.1774116  0.2198971  0.14726765 0.         0.
  0.         0.         0.2198971  0.         0.         0.2198971
  0.         0.1774116  0.         0.         0.2198971  0.
  0.         0.         0.         0.1774116  0.         0.2198971
  0.         0.2198971  0.2198971  0.         0.         0.
  0.1774116  0.14726765 0.1774116  0.2198971  0.         0.2198971
  0.         0.         0.         0.         0.         0.2198971
  0.         0.1774116  0.         0.         0.         0.
  0.         0.         0.         0.         0.10478215 0.2198971
  0.2198971  0.         0.         0.         0.2198971  0.2198971
  0.         0.         0.         0.         0.         0.1774116
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.14425918 0.         0.
  0.         0.         0.         0.         0.28851836 0.
  0.21540491 0.         0.21540491 0.21540491 0.         0.
  0.         0.         0.         0.         0.21540491 0.
  0.21540491 0.         0.         0.         0.21540491 0.
  0.17378733 0.14425918 0.         0.         0.         0.
  0.21540491 0.21540491 0.         0.         0.21540491 0.
  0.21540491 0.17378733 0.         0.         0.21540491 0.
  0.         0.         0.         0.         0.20528319 0.
  0.         0.21540491 0.         0.         0.         0.
  0.         0.         0.         0.         0.21540491 0.17378733
  0.21540491 0.21540491 0.         0.21540491]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.26136134 0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.32395065 0.         0.         0.
  0.         0.         0.32395065 0.32395065 0.         0.
  0.26136134 0.         0.         0.         0.15436422 0.
  0.         0.         0.32395065 0.32395065 0.         0.
  0.32395065 0.32395065 0.         0.32395065 0.         0.
  0.         0.         0.         0.        ]]
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarr
  warnings.warn(msg, category=FutureWarning)
```

Euclidean Distance

```
euclidean_dist = euclidean_distances(X)
squared_euclidean = np.square(euclidean_dist)
print (squared_euclidean)
```

```
[[0.         1.89196424 1.79225892 1.74225576 1.96085117]
 [1.89196424 0.         1.73870149 1.76704757 1.82908412]
 [1.79225892 1.73870149 0.         1.6870098  1.8749137 ]
 [1.74225576 1.76704757 1.6870098  0.         1.93662324]
 [1.96085117 1.82908412 1.8749137  1.93662324 0.        ]]
```

Cosine Similarity

```
print(cosine_similarity(X))
```

```
[[1.         0.05401788 0.10387054 0.12887212 0.01957441]
 [0.05401788 1.         0.13064926 0.11647622 0.08545794]
 [0.10387054 0.13064926 1.         0.1564951  0.06254315]
 [0.12887212 0.11647622 0.1564951  1.         0.03168838]
 [0.01957441 0.08545794 0.06254315 0.03168838 1.        ]]
```

Meta Search

```
import math

jud=int(input("Enter No Of Judges:"))
par=int(input("Enter No Of Participants"))

create_csv=[]

for i in range(par):
    temp=[]
    for j in range(jud):
        print("Enter Rank Given By Judge ",j+1," To Participant ",i+1," :")
        vote=input()
        if (vote==""):
            vote=math.inf
        else:
            vote=int(vote)
        temp.append(vote)
    create_csv.append(temp)
print(create_csv)
```

```
Enter No Of Judges:4
Enter No Of Participants5
Enter Rank Given By Judge  1  To Participant  1  :
1
Enter Rank Given By Judge  2  To Participant  1  :
2
Enter Rank Given By Judge  3  To Participant  1  :
5
Enter Rank Given By Judge  4  To Participant  1  :
4
Enter Rank Given By Judge  1  To Participant  2  :
2
Enter Rank Given By Judge  2  To Participant  2  :
1
Enter Rank Given By Judge  3  To Participant  2  :
4
Enter Rank Given By Judge  4  To Participant  2  :
3
Enter Rank Given By Judge  1  To Participant  3  :
3
Enter Rank Given By Judge  2  To Participant  3  :
4
Enter Rank Given By Judge  3  To Participant  3  :
1
Enter Rank Given By Judge  4  To Participant  3  :
2
Enter Rank Given By Judge  1  To Participant  4  :
4
```

```
Enter Rank Given By Judge  2  To Participant  4  :
3
Enter Rank Given By Judge  3  To Participant  4  :
2
Enter Rank Given By Judge  4  To Participant  4  :
1
Enter Rank Given By Judge  1  To Participant  5  :
5
Enter Rank Given By Judge  2  To Participant  5  :
5
Enter Rank Given By Judge  3  To Participant  5  :
3
Enter Rank Given By Judge  4  To Participant  5  :
5
[[1, 2, 5, 4], [2, 1, 4, 3], [3, 4, 1, 2], [4, 3, 2, 1], [5, 5, 3, 5]]
```

```
df = pd.DataFrame(create_csv)
df.to_csv('data.csv')
```

```
print(create_csv)
```

```
[[1, 2, 5, 4], [2, 1, 4, 3], [3, 4, 1, 2], [4, 3, 2, 1], [5, 5, 3, 5]]
```

```
print(df)
```

```
   0  1  2  3
0  1  2  5  4
1  2  1  4  3
2  3  4  1  2
3  4  3  2  1
4  5  5  3  5
```

```
Borda=df
Reciprocal=df
diff=0
for i in range(jud):
    Reciprocal[i]=Reciprocal[i].replace(math.inf,0)
Reciprocal=Reciprocal.values.tolist()

for i in range(jud):
    Borda[i]=Borda[i].replace(math.inf,0)
    data=Borda[i]
    a=0
    count=0
    for j in range(par):
        if data[j]!=0:
            a=a+(par-data[j]+1)
        else:
            count=count+1
    if (count):
        s = sum(range(1, par + 1))
        diff=(s-a)/count

    Borda[i]=Borda[i].replace(0,diff)
```

```
Borda=Borda.values.tolist()
```

```
print(Borda)
print(Reciprocal)
```

```
[[1, 2, 5, 4], [2, 1, 4, 3], [3, 4, 1, 2], [4, 3, 2, 1], [5, 5, 3, 5]]
[[1, 2, 5, 4], [2, 1, 4, 3], [3, 4, 1, 2], [4, 3, 2, 1], [5, 5, 3, 5]]
```

Borda Method

```
ans=[]
for i in range(par):
    a=0
    data=Borda[i]
    for j in range(jud):
        a=a+(par-data[j]+1)
    ans.append(a)
```

```
print(ans)
```

```
[12, 14, 14, 14, 6]
```

Reciprocal

```
ans=[]
for i in range(par):
    a=0
    data=Reciprocal[i]
    for j in range(jud):
        if(data[j]==0):
            continue
        a=a+(1/data[j])
    ans.append(a)
```

```
print(ans)
```

```
[1.95, 2.0833333333333335, 2.083333333333333, 2.083333333333333, 0.9333333333333333]
```

Condercet

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of
```

```
from itertools import combinations
```

```
df = pd.read_csv("condercet.csv")
```

```
df
```

	Document	J1	J2	J3	J4
0	P1	1	2	5	4
1	P2	2	1	4	3
2	P3	3	4	1	2
3	P4	4	3	2	1
4	P5	5	5	3	5

```
document_list = df.iloc[:,0].tolist()
```

```
document_list
```

```
['P1', 'P2', 'P3', 'P4', 'P5']
```

```
pairs = list(combinations(document_list, 2))
```

```
pairs
```

```
[('P1', 'P2'),
 ('P1', 'P3'),
 ('P1', 'P4'),
 ('P1', 'P5'),
 ('P2', 'P3'),
 ('P2', 'P4'),
 ('P2', 'P5'),
 ('P3', 'P4'),
 ('P3', 'P5'),
 ('P4', 'P5')]
```

```
print(df[df["Document"]=="P1"].index.values[0])
```

```
0
```

```
df["J1"][0]
```

```
1
```

```
combinations = {}
for i in pairs:
    item_1 = i[0]
    item_2 = i[1]
    index_1 = df[df["Document"] == item_1].index.values[0]
    index_2 = df[df["Document"] == item_2].index.values[0]
    for j in df.columns[1:]:
        if j not in combinations:
            combinations[j] = []
        if df[j][index_1] < df[j][index_2]:
            combinations[j].append(item_1)
        elif df[j][index_1] > df[j][index_2]:
            combinations[j].append(item_2)
        else:
            combinations[j].append("-")
```

```
combinations
```

```
{'J1': ['P1', 'P1', 'P1', 'P1', 'P2', 'P2', 'P2', 'P3', 'P3', 'P4'],
 'J2': ['P2', 'P1', 'P1', 'P1', 'P2', 'P2', 'P2', 'P4', 'P3', 'P4'],
 'J3': ['P2', 'P3', 'P4', 'P5', 'P3', 'P4', 'P5', 'P3', 'P3', 'P4'],
 'J4': ['P2', 'P3', 'P4', 'P1', 'P3', 'P4', 'P2', 'P4', 'P3', 'P4']}
```

```
pair_df = pd.DataFrame(combinations)
pair_df
```

	J1	J2	J3	J4
0	P1	P2	P2	P2
1	P1	P1	P3	P3
2	P1	P1	P4	P4

```
pair_df["Pairs"] = pairs
```

```
pair_df = pair_df[["Pairs", "J1", "J2", "J3", "J4"]]
```

```
pair_df
```

	Pairs	J1	J2	J3	J4
0	(P1, P2)	P1	P2	P2	P2
1	(P1, P3)	P1	P1	P3	P3
2	(P1, P4)	P1	P1	P4	P4
3	(P1, P5)	P1	P1	P5	P1
4	(P2, P3)	P2	P2	P3	P3
5	(P2, P4)	P2	P2	P4	P4
6	(P2, P5)	P2	P2	P5	P2
7	(P3, P4)	P3	P4	P3	P4
8	(P3, P5)	P3	P3	P3	P3
9	(P4, P5)	P4	P4	P4	P4

```
win_loss_1 = []
win_loss_2 = []
for i in range(pair_df.shape[0]):
    count_item_1 = 0
    count_item_2 = 0
    tie = 0
    count_ties = 0

    item_1 = pair_df["Pairs"][i][0]
    item_2 = pair_df["Pairs"][i][1]

    row = pair_df.iloc[i, 1:]
    for j in row:
        if j == item_1:
            count_item_1 += 1
        else:
            count_item_2 += 1
    print("Item: " + str(item_1) + " Count: " + str(count_item_1))
    print("Item: " + str(item_2) + " Count: " + str(count_item_2))
```

```
        out_str_1 = ""
        out_str_2 = ""

        if item_1 == "-" or item_2 == "-":
            tie = 1
        out_str_1 = str(count_item_1) + ":" + str(count_item_2) + ":" + str(tie)
        out_str_2 = str(count_item_2) + ":" + str(count_item_1) + ":" + str(tie)

        print("Output String 1: ", out_str_1)
        print("Output String 2: ", out_str_2)

        print("\n\n")
        win_loss_1.append(out_str_1)
        win_loss_2.append(out_str_2)
```

```
Item: P1 Count: 2
Item: P4 Count: 2
Output String 1:  2:2:0
Output String 2:  2:2:0



Item: P1 Count: 3
Item: P5 Count: 1
Output String 1:  3:1:0
Output String 2:  1:3:0



Item: P2 Count: 2
Item: P3 Count: 2
Output String 1:  2:2:0
Output String 2:  2:2:0



Item: P2 Count: 2
Item: P4 Count: 2
Output String 1:  2:2:0
Output String 2:  2:2:0



Item: P2 Count: 3

Item: P5 Count: 1
Output String 1:  3:1:0
Output String 2:  1:3:0



Item: P3 Count: 2
Item: P4 Count: 2
Output String 1:  2:2:0
Output String 2:  2:2:0
```

```
Item: P3 Count: 4
Item: P5 Count: 0
Output String 1:  4:0:0
Output String 2:  0:4:0



Item: P4 Count: 4
Item: P5 Count: 0
Output String 1:  4:0:0
Output String 2:  0:4:0
```

```
win_loss_1
```

```
['1:3:0',
 '2:2:0',
 '2:2:0',
 '3:1:0',
 '2:2:0',
 '2:2:0',
 '3:1:0',
 '2:2:0',
 '4:0:0',
 '4:0:0']
```

```
win_loss_2
```

```
['3:1:0',
 '2:2:0',
 '2:2:0',
 '1:3:0',
 '2:2:0',
 '2:2:0',
 '1:3:0',
 '2:2:0',
 '0:4:0',
 '0:4:0']
```

```
document_matrix = np.empty((len(document_list), len(document_list)), dtype = "<U5")
```

```
document_matrix
```

```
array([['', '', '', '', ''],
       ['', '', '', '', ''],
       ['', '', '', '', ''],
       ['', '', '', '', ''],
       ['', '', '', '', '']], dtype='<U5')
```

```
counter_win_loss_1 = 0
counter_win_loss_2 = 0
for i in range(document_matrix.shape[0]):
    for j in range(i, document_matrix.shape[1]):
        if i == j:
            document_matrix[i][j] = "-"
        else:
            document_matrix[i][j] = win_loss_1[counter_win_loss_1]
            print("\nWin_Loss_1: ", type(win_loss_1[counter_win_loss_1]))

            document_matrix[j][i] = win_loss_2[counter_win_loss_2]
            print("\nWin_Loss_2: ", win_loss_2[counter_win_loss_2])

            print("\nDocument Matrix: \n", document_matrix)

            counter_win_loss_1 += 1
            counter_win_loss_2 += 1
```

```
 ['2:2:0' '2:2:0' '' '' '']
 ['2:2:0' '' '' '' '']
 ['1:3:0' '' '' '' '']]

Win_Loss_1:  <class 'str'>

Win_Loss_2:  2:2:0

Document Matrix:
 [['-' '1:3:0' '2:2:0' '2:2:0' '3:1:0']
 ['3:1:0' '-' '2:2:0' '2:2:0' '']
 ['2:2:0' '2:2:0' '' '' '']
 ['2:2:0' '2:2:0' '' '' '']
 ['1:3:0' '' '' '' '']]

Win_Loss_1:  <class 'str'>

Win_Loss_2:  1:3:0

Document Matrix:
 [['-' '1:3:0' '2:2:0' '2:2:0' '3:1:0']
 ['3:1:0' '-' '2:2:0' '2:2:0' '3:1:0']
 ['2:2:0' '2:2:0' '' '' '']
 ['2:2:0' '2:2:0' '' '' '']
 ['1:3:0' '1:3:0' '' '' '']]

Win_Loss_1:  <class 'str'>

Win_Loss_2:  2:2:0

Document Matrix:
 [['-' '1:3:0' '2:2:0' '2:2:0' '3:1:0']
 ['3:1:0' '-' '2:2:0' '2:2:0' '3:1:0']
 ['2:2:0' '2:2:0' '-' '2:2:0' '']
 ['2:2:0' '2:2:0' '2:2:0' '' '']
 ['1:3:0' '1:3:0' '' '' '']]

Win_Loss_1:  <class 'str'>

Win_Loss_2:  0:4:0

Document Matrix:
 [['-' '1:3:0' '2:2:0' '2:2:0' '3:1:0']
 ['3:1:0' '-' '2:2:0' '2:2:0' '3:1:0']
 ['2:2:0' '2:2:0' '-' '2:2:0' '4:0:0']
 ['2:2:0' '2:2:0' '2:2:0' '' '']
 ['1:3:0' '1:3:0' '0:4:0' '' '']]

Win_Loss_1:  <class 'str'>

Win_Loss_2:  0:4:0

Document Matrix:
 [['-' '1:3:0' '2:2:0' '2:2:0' '3:1:0']
 ['3:1:0' '-' '2:2:0' '2:2:0' '3:1:0']
 ['2:2:0' '2:2:0' '-' '2:2:0' '4:0:0']
 ['2:2:0' '2:2:0' '2:2:0' '-' '4:0:0']
 ['1:3:0' '1:3:0' '0:4:0' '0:4:0' '']]
```

```
document_matrix
```

```
array([['-', '1:3:0', '2:2:0', '2:2:0', '3:1:0'],
       ['3:1:0', '-', '2:2:0', '2:2:0', '3:1:0'],
       ['2:2:0', '2:2:0', '-', '2:2:0', '4:0:0'],
       ['2:2:0', '2:2:0', '2:2:0', '-', '4:0:0'],
       ['1:3:0', '1:3:0', '0:4:0', '0:4:0', '-']], dtype='<U5')
```

```
final_df = pd.DataFrame(columns=["Document", "Win", "Lose", "Tie"])
```

```
final_df["Document"] = np.array(document_list).T
```

```
final_df
```

	Document	Win	Lose	Tie
0	P1	NaN	NaN	NaN
1	P2	NaN	NaN	NaN
2	P3	NaN	NaN	NaN
3	P4	NaN	NaN	NaN
4	P5	NaN	NaN	NaN

```
for i in range(document_matrix.shape[0]):
    win = 0
    loss = 0
    tie = 0
    for j in range(document_matrix.shape[1]):
        if i == j:
            continue
        values = document_matrix[i][j][:-2].split(":")
        item_1 = int(values[0])
        item_2 = int(values[1])
        if item_1 > item_2:
            win += 1
        elif item_1 < item_2:
            loss += 1
        else:
            tie += 1
    final_df["Win"][i] = win
    final_df["Lose"][i] = loss
    final_df["Tie"][i] = tie

final_df
```

	Document	Win	Lose	Tie
0	P1	1	1	2
1	P2	2	0	2
2	P3	1	0	3
3	P4	1	0	3
4	P5	0	4	0

✓ 0s completed at 2:45 PM