クレジット:
UTokyo Online Educationデータマイニング入門 2018 森　純一郎

東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

課題4 テキストデータ分析

Q1

n-次元ベクトル空間における、任意の2つのベクトル、$\vec{x} = (x_1, x_2, \ldots, x_n)$、$\vec{y} = (y_1, y_2, \ldots, y_n)$、の間のcos類似度 $cos(X, Y)$ は以下のように定義されます。

$$cos(X, Y) = \frac{\vec{x} \cdot \vec{y}}{\|x\|_2 \|y\|_2} = \frac{\Sigma_{i=1}^{n} x_i y_i}{\sqrt{\Sigma_{i=1}^{n} x_i^2} \sqrt{\Sigma_{i=1}^{n} y_i^2}}$$

入力ベクトル $\vec{x}, \vec{y}$ をそれぞれ `NumPy` の配列として引数で受け取り、それらのベクトル間のcos類似度を計算して返す関数 `cos_sim` を完成させてください。

In [ ]:
```
import numpy as np
```

In [ ]:
```
def cos_sim(vec1, vec2):

```

Q2

"course_list.csv"ファイルには以下のように各行に講義名のテキストデータが含まれています。以下では、このファイルを読み込み、各講義のベクトルを作成し、講義間の類似度を求めるコードを実装します。

```
## course_list.csvファイル
...
行動神経科学
認知心理学
性格心理学
スポーツ栄養学
グローバル教養特別演習I（１３）
認知行動障害論
スポーツバイオメカニクス
...
```

まず必要なモジュールをimportします。

In [ ]:
```
import json
import csv
import numpy as np
import pandas as pd
```

Q2.1

"course_list.csv"ファイルを1行ずつ**順番に**読み込み、その各行を要素とするリストを作成して返す `course_list` 関数を完成さてください。作成されたリストは変数 `courses` で受け取ります。以降の処理では、リスト `courses` のインデックスをその要素（講義名）のIDとして扱います。

In [ ]:
```
def course_list():
```

Q2.2

"keyword_list.csv"ファイルには以下のように各行に1単語が含まれています。

```
## keyword_list.csvファイル
演習
特殊
講義
科学
研究
工学
社会
コース
日本
文化
...
```

"keywor_list.csv"ファイルを1行ずつ**順番に**読み込み、その各行を要素とするリストを作成して返す `vocab_list` 関数を完成さてください。作成されたリストは変数 `vocab` で受け取ります。以降の処理では、リスト `vocab` のインデックスをその要素（単語）のIDとして扱います。リスト `vocab` は以降の処理における語彙となります。

In [ ]:
```
def vocab_list():
```

Q2.3

リスト `courses` と `vocab` を引数で受け取り、単語のID（リスト `vocab` のその単語のインデックス）をキー、その単語のDF（Document Frequency)を値とする辞書を作成して返す `count_df` 関数を作成してください。作成された辞書は変数 `df` で受け取ります。この場合、ある単語のDF値はその単語を講義名に含む講義数に対応します。

In [ ]:
```
def count_df(courses, vocab):
```

Q2.4.1

リスト vocab の各単語を次元とする講義ベクトルを考えます。講義ベクトルの長さはリスト vocab の長さと等しく、リスト vocab のインデックス i の単語 vocab[i] が講義名に含まれる時、講義ベクトルの i 番目の要素は 1 、含まれなければ 0 とします。

以下では、リスト courses と vocab を引数で受け取り、リスト courses の各講義のベクトルを行、リスト vocab の各単語を列とした NumPy の行列を作成して返す lec_word_matrix 関数を完成させてください。作成した講義-単語行列は、講義（行）の講義名に単語（列）が含まれていれば、その要素が1であるような行列です。

In [ ]:
```
def lec_word_matrix(courses, vocab)
```

Q2.4.2

Q2.4.1で作成した講義-単語行列の各要素を、その講義の講義名に単語が含まれるか否かの1or0ではなく、その講義の講義名に単語が何回含まれるか（TF: Term Frequency)で表した行列を作成して返す lec_word_tf_matrix 関数を完成させてください。

In [ ]:
```
def lec_word_tf_matrix(courses, vocab)
```

Q2.4.3

Q2.4.2で作成した講義-単語行列の各要素（講義 i の単語 j のTF_{ij}）にその単語のIDF値を掛けたTFIDF値を要素とする行列を作成して返す lec_word_tfidf_matrix 関数を完成させてください。作成した行列は変数 tfidf_matrix で受け取ります。

ここで、講義 i 、単語 j のTFIDF値は以下のように定義されます。

```
TFIDF=TF_ij*log(すべての講義数/単語jのDF)=TF_ij*log(len(courses)/df[j])
```

log の計算には np.log() を使用してよいです。

In [ ]:
```
def lec_word_tfidf_matrix(courses, vocab, df):
```

Q2.5

Q2.4.3で作成した講義-単語行列を元に、入力の講義に対してcos類似度に基づいて他のすべての講義との類似度を計算し、類似する講義名をキー、その類似度を値とした辞書を返す以下の `find_similar_course` 関数を完成させてください。その際、入力の講義および類似度が0の講義は辞書に含めないようにしてください。cos類似度の計算にはQ1で作成した関数を使ってもよいです。

In [ ]:
```
def find_similar_course(target, matrix, courses):

```