**Lab 4**

Họ và tên: Nguyễn Khánh Nhật

MSSV:41.01.104.080

Lớp: CNTT A

Phần 1: Content Based

Thuật toán : Sử dụng phương pháp lọc theo nội dung dựa trên thuật toán tdidf cũng như thư viện stopwords loại bỏ những từ thông dụng trong tiếng anh để tìm ra độ tương đồng giữa các item trong văn bản

Giải thích code :

import pandas as pd //import thư viện pandas bộ công cụ để phân tích và chỉnh sửa dữ liệu

from sklearn.feature\_extraction.text import TfidfVectorizer // import thư viện tdidf

from sklearn.metrics.pairwise import linear\_kernel //import thư viện linear\_kernel

from nltk.corpus import stopwords //import thư viện stopwords

class ContentEngine(object):

def \_\_init\_\_(self, input\_file):

"""

read the input file and calculate tfidf matrix

Arguments

- input\_file (text) : input file contains two columns: id and description

"""

self.df\_input = pd.read\_csv(input\_file) // đọc dữ liệu từ file csv

print(self.df\_input.head(10)) //In ra 10 dòng đầu tiên

self.tfidf\_vectorizer = TfidfVectorizer(stop\_words=stopwords.words('english')) // khởi tạo đối tượng là vecto tfidf nếu trong đối tượng này có những từ ngữ thông dụng trong tiếng anh sẽ bị xóa bỏ

self.tfidf\_matrix = self.tfidf\_vectorizer.fit\_transform(self.df\_input["description"]) //khởi tạo ma trận self.tfidf

print(self.tfidf\_matrix) //In ra ma trận self.tfidf

def predict(self, id, topK):

"""

predict similarity, given id of the item

Arguments

- id (int) : id of the item in the data file

- topK (int) : top-K similar items to retrieve

"""

index = self.df\_input[self.df\_input['id'] == id].index.tolist()[0] //Lấy ra index trong văn bản so sánh vs tham số id truyền vào

simscore\_matrix = linear\_kernel(self.tfidf\_matrix[index:index+1], self.tfidf\_matrix)[0]//tính độ tương đồng của ma trận bằng công thức cosine

print(simscore\_matrix) // In ra ma trận tương đồng vừa tìm được

related\_docs\_indices = [index\_sim for index\_sim in simscore\_matrix.argsort()[::-1] if index\_sim != index] // sắp xếp lại ma trận theo thứ tự giảm dần của độ tương đồng

print("Input: {}".format(self.df\_input.iloc[index]["description"])) //In ra description để tìm top k giống nhau

for index\_sim in related\_docs\_indices[:topK]:

print("\t{}\t{}".format(index\_sim, self.df\_input.iloc[index\_sim]["description"]))

#==========================================================================

if \_\_name\_\_ == "\_\_main\_\_":

ce = ContentEngine("./sample-data.csv") // khởi tạo đối tượng vs constructor là file csv

ce.predict(id=4, topK=5) //thực hiện hàm predict vs id =4 và topK=5

Phần 2: CollaborativeFiltering

Thuật toán: Sử dụng phương pháp lọc cộng tác cùng thuật toán phân ra ma trận để đánh giá tính ngầm định của sản phẩm

Giải thích:

import numpy as np //import thư viện nu,py

class CollaborativeFiltering():

def \_\_init\_\_(self, R, K, alpha, beta, iterations):

"""

Perform matrix factorization to predict empty

entries in a matrix.

Arguments

- R (ndarray) : user-item rating matrix

- K (int) : number of latent dimensions

- alpha (float) : learning rate

- beta (float) : regularization parameter

"""

self.R = R

self.num\_users, self.num\_items = R.shape

self.K = K

self.alpha = alpha

self.beta = beta

self.iterations = iterations

def train(self):

# Initialize user and item latent feature matrice //khởi tạo ma trận đối tượng tìm ẩn

self.P = np.random.normal(scale=1./self.K, size=(self.num\_users, self.K))// khởi tạo ngẫy nhiên ma trận p theo user

self.Q = np.random.normal(scale=1./self.K, size=(self.num\_items, self.K))// khởi tạo ngẫu nhiên ma trận q theo item

# Initialize the biases

self.b\_u = np.zeros(self.num\_users)// khởi tạo ma trận với gán các num\_users giá trị =0

self.b\_i = np.zeros(self.num\_items) )// khởi tạo ma trận với gán các num\_items giá trị =0

self.b = np.mean(self.R[np.where(self.R != 0)])// gán giá trị mean các phần tử khác 0

# Create a list of training samples

self.samples = [

(i, j, self.R[i, j])

for i in range(self.num\_users)

for j in range(self.num\_items)

if self.R[i, j] > 0

]

# Perform stochastic gradient descent for number of iterations

training\_process = []

for i in range(self.iterations):

np.random.shuffle(self.samples)

self.sgd()

mse = self.mse()

training\_process.append((i, mse))

if (i+1) % 10 == 0:

print("Iteration: %d ; error = %.4f" % (i+1, mse))

return training\_process

def mse(self):// Hàm tính tổng sai số trung bình

"""

A function to compute the total mean square error

"""

xs, ys = self.R.nonzero() // lấy các dòng đánh giá khác 0

predicted = self.full\_matrix() // dự đoán kết quả đánh giá của toàn ma trận

error = 0

for x, y in zip(xs, ys):

error += pow(self.R[x, y] - predicted[x, y], 2)

return np.sqrt(error)

def sgd(self):

"""

Perform stochastic graident descent

"""

for i, j, r in self.samples:

# Computer prediction and error// dự đoán và sai số

prediction = self.get\_rating(i, j)

e = (r - prediction)

# Update biases // cập nhật lại độ chênh lệch

self.b\_u[i] += self.alpha \* (e - self.beta \* self.b\_u[i])

self.b\_i[j] += self.alpha \* (e - self.beta \* self.b\_i[j])

# Update user and item latent feature matrices // Cập nhật ma trận tính năng tiềm ẩn của người dùng và item

self.P[i, :] += self.alpha \* (e \* self.Q[j, :] - self.beta \* self.P[i,:])

self.Q[j, :] += self.alpha \* (e \* self.P[i, :] - self.beta \* self.Q[j,:])

def get\_rating(self, i, j):

"""

Get the predicted rating of user i and item j //Lấy rating dự đoán của user I và item j

"""

prediction = self.b + self.b\_u[i] + self.b\_i[j] + self.P[i, :].dot(self.Q[j, :].T)

return prediction

def full\_matrix(self):

"""

Computer the full matrix using the resultant biases, P and Q //Tính đầy đủ ma trận sử dụng kết quả phân tách ,P và Q để tìm ra kết quả

"""

return self.b + self.b\_u[:,np.newaxis] + self.b\_i[np.newaxis:,] + self.P.dot(self.Q.T)

#========================================================================================

if \_\_name\_\_ == "\_\_main\_\_":

R = np.array([

[5, 3, 0, 1],

[4, 0, 0, 1],

[1, 1, 0, 5],

[1, 0, 0, 4],

[0, 1, 5, 4],

])

cf = CollaborativeFiltering(R, K=2, alpha=0.1, beta=0.01, iterations=30)// khởi tạo ma trận

cf.train()

print(cf.full\_matrix())// In kết quả ma trận