import pandas as pd

import numpy as np

import string

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

sw = stopwords.words('english')

# read the csv files into pd

amazon\_small = pd.read\_csv('amazon\_small.csv')

google\_small = pd.read\_csv('google\_small.csv')

#Find all possible combinations of two data sets and read into a dataframe

amazon\_small['key'] = 1

google\_small['key'] = 1

data\_small = pd.merge(amazon\_small,google\_small,on='key').drop('key',axis=1)

data\_small.columns = ['idAmazon', 'amazon\_name', 'amazon\_description', 'amazon\_manufacturer', 'amazon\_price',

'idGoogleBase', 'google\_name', 'google\_description', 'google\_manufacturer', 'google\_price']

data\_small

def get\_vectors(\*strs):

# normalise the text by remove stopwords, punctuation from string

text = [(lemmatizer.lemmatize(w)).lower() for w in strs if (not w in sw) and (not w in string.punctuation)]

vectorizer = CountVectorizer(text)

vectorizer.fit(text)

return vectorizer.transform(text).toarray()

def get\_cosine\_sim(\*strs):

vectors = [t for t in get\_vectors(\*strs)]

return cosine\_similarity(vectors)

def get\_price\_sim(p1, p2):

p\_similarity = min(p1, p2)/max(p1, p2)

return p\_similarity

# replace Null with empty string

data\_small["amazon\_description"].replace(np.nan,"",inplace = True)

data\_small["google\_description"].replace(np.nan,"", inplace = True)

name\_similarity = []

des\_similarity = []

price\_similarity = []

final\_score = []

for i in range(len(data\_small)):

name\_sim = 0

des\_sim = 0

price\_sim = 0

amazon\_name = data\_small['amazon\_name'][i]

google\_name = data\_small['google\_name'][i]

name\_sim = get\_cosine\_sim(amazon\_name, google\_name)[0][1]

name\_similarity.append(name\_sim)

amazon\_description = data\_small['amazon\_description'][i]

google\_description = data\_small['google\_description'][i]

if not(amazon\_description == '') and not(google\_description == ''):

des\_sim = get\_cosine\_sim(amazon\_description, google\_description)[0][1]

des\_similarity.append(des\_sim)

price\_sim = get\_price\_sim(data\_small['amazon\_price'][i], data\_small['google\_price'][i])

price\_similarity.append(price\_sim)

score = name\_sim + des\_sim + price\_similarity

final\_score.append(score)

data\_small['name\_similarity'] = name\_similarity

data\_small['des\_similarity'] = des\_similarity

data\_small['price\_similarity'] = price\_similarity

data\_small['final\_score'] = final\_score

data\_small

price\_sim = get\_price\_sim(data\_small['amazon\_price'][1], data\_small['google\_price'][1])

price\_similarity.append(price\_sim)

price\_similarity

#Determine the matches

amazon\_id = []

google\_id = []

threshold = 2

for i in range(len(data\_small)):

if data\_small['final\_score'][i] >= threshold:

amazon\_id.append(data\_small["id\_amazon"][i])

google\_id.append(data\_small["id\_google"][i])

match = pd.DataFrame({'idAmazon': amazon\_id, 'idGoogleBase': google\_id})

match

truth = pd.read\_csv('amazon\_google\_truth\_small.csv')

t = len(truth)

tp = 0

fn = 0

fp = 0

for i in range(len(match)):

matchpair = 0

for j in range(len(truth)):

if match["idAmazon"][i] == truth["idAmazon"][j] and match['idGoogle'][i] == truth['idGoogleBase'][j]:

tp += 1

matchpair = 1

if not matchpair:

fp += 1

fn = t - tp

#print(tp)

#print(fp)

#print(fn)

precision = tp/(fp+tp)

print(precision)

recall = tp/(tp+fn)

print(recall)

---------------------------------------------------------------------------------------------------------------------------

pip install textdistance[extras]

---------------------------------------------------------------------------------------------------------------------------

# Program to measure similarity between

# two sentences using cosine similarity.

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

# X = input("Enter first string: ").lower()

# Y = input("Enter second string: ").lower()

X = "1. arthur's theme 2. axel f 3. love theme ?out of africa? 4. raiders march 5. the way he makes me feel 6. somewhere out there 7. cantina band 8. princess leia's theme 9. against all odds 10. can you read my mind 11. the windmills of your mind/"

Y = "contains the following songs: 1. arthur's theme 2. axel f 3. love theme out of africa 4. raiders march 5. the way he makes me feel 6. somewhere out there 7. cantina band 8. princess leia's theme 9. against all odds 10. can you read my mind 11. the windmills of your mind/the summer knows (theme from summer of '42)"

# tokenization

X\_list = word\_tokenize(X)

Y\_list = word\_tokenize(Y)

# sw contains the list of stopwords

sw = stopwords.words('english')

l1 =[];l2 =[]

# remove stop words from string

X\_set = {lemmatizer.lemmatize(w) for w in X\_list if not w in sw}

Y\_set = {lemmatizer.lemmatize(w) for w in Y\_list if not w in sw}

# form a set containing keywords of both strings

rvector = X\_set.union(Y\_set)

for w in rvector:

if w in X\_set: l1.append(1) # create a vector

else: l1.append(0)

if w in Y\_set: l2.append(1)

else: l2.append(0)

c = 0

# cosine formula

for i in range(len(rvector)):

c+= l1[i]\*l2[i]

cosine = c / float((sum(l1)\*sum(l2))\*\*0.5)

print("similarity: ", cosine)

---------------------------------------------------------------------------------------------------------------------------

#from collections import Counter

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

def get\_cosine\_sim(\*strs):

vectors = [t for t in get\_vectors(\*strs)]

return cosine\_similarity(vectors)

def get\_vectors(\*strs):

text = [t for t in strs if not w in sw]

vectorizer = CountVectorizer(text)

vectorizer.fit(text)

return vectorizer.transform(text).toarray()

get\_cosine\_sim(X, Y)[0][1]

---------------------------------------------------------------------------------------------------------------------------

import string

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

from nltk.corpus import stopwords

sw = stopwords.words('english')

sentences = ["phillip keveren - 501056 - hollywood volume 1", "hollywood volume 1"]

def clean\_string(text):

text = ''.join ([word for word in text if word not in string.punctuation])

text = text.lower()

text = ' '.join ([word for word in text.split() if word not in sw])

return text

cleaned = list(map(clean\_string, sentences))

vectorizer = CountVectorizer().fit\_transform(cleaned)

vectors = vectorizer.toarray()

csim = cosine\_similarity(vectors)

def cosine\_sim\_vectors(vec1, vec2):

vec1 = vec1.reshape(1, -1)

vec2 = vec2.reshape(1, -1)

return cosine\_similarity(vec1, vec2)[0][0]

cosine\_sim\_vectors(vectors[0], vectors[1])

---------------------------------------------------------------------------------------------------------------------------

lemmatizer = WordNetLemmatizer()

sw = stopwords.words('english')

X = "1. arthur's theme 2. axel f 3. love theme ?out of africa? 4. raiders march 5. the way he makes me feel 6. somewhere out there 7. cantina band 8. princess leia's theme 9. against all odds 10. can you read my mind 11. the windmills of your mind/"

Y = "contains the following songs: 1. arthur's theme 2. axel f 3. love theme out of africa 4. raiders march 5. the way he makes me feel 6. somewhere out there 7. cantina band 8. princess leia's theme 9. against all odds 10. can you read my mind 11. the windmills of your mind/the summer knows (theme from summer of '42)"

#X = "phillip keveren - 501056 - hollywood volume 1"

#Y = "hollywood volume 1"

#X = X.lower()

#Y = Y.lower()

def get\_vectors(\*strs):

# normalise the text by remove stopwords, punctuation from string

text = [(lemmatizer.lemmatize(w)).lower() for w in strs if (not w in sw) and (not w in string.punctuation)]

vectorizer = CountVectorizer(text)

vectorizer.fit(text)

return vectorizer.transform(text).toarray()

def get\_cosine\_sim(\*strs):

vectors = [t for t in get\_vectors(\*strs)]

return cosine\_similarity(vectors)

get\_cosine\_sim(X, Y)[0][1]