Ranked Retrieval System

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd

df = pd.read_excel('/content/drive/MyDrive/IR-Ranked-Retrieval--main/ConceptualCaptionsDataset.xlsx',header = None)

first_column = df.iloc[:10, 0]
print(first_column)
```

```
0                           a very typical bus station
1    sierra looked stunning in this top and this sk...
2    young confused girl standing in front of a war...
3    interior design of modern living room with fir...
4      cybernetic scene isolated on white background .
5    gangsta rap artist attends sports team vs play...
6    the jetty : different types of plants to estab...
7     traditional ornamental floral paisley bandanna .
8    # of the sports team skates against sports tea...
9    by geographical feature category or in the cit...
Name: 0, dtype: object
```

```
print(df.shape)
```

```
(30000, 2)
```

```
N = 2000
```

Printing Documents

```
documents = list()
for i in range(N):
  documents.append(df.iloc[i, 0])
  print(df.iloc[i, 0])
```

```
a delicate ring adorned with your favorite initials
the sand on the dunes
a statue of monarch is unveiled at a ceremony
sparkling shiny gold uppercase or capital letter i in a 3d illustration with a golden color rough textured metallic surface and ancie
a colourful assortment of various sizes of fans displayed for sale at an outdoor market
tourists disembark from a helicopter along the area
glowing lettering with shamrock on a dark green background .
it 's so humid these little guys are growing in my hotel room .
a view of the beach in ft .
vector illustration of a minimalistic forest and house vector
folk rock artist and actor at the premiere
portrait of a laughing man wearing glasses
actor arrives at the premiere of season .
cake for an avid gardener .
empty blue seat of a football stadium
actor and tv personality compete monday onfamily tv program .
rake and shovel icon digital red for any design isolated on white
unearthed : thanks to science , we may see the rebirth of the chestnut
actor and his son during the award red carpet arrivals
man with a briefcase in an airport .
the previous logo , compared with the latest logo .
get the scoop : read all about transformation in magazine
firefighters stand on lanes of the freeway near smoldering hot spots of a large fire that consumed an under - construction apartment
singer 's tour bus goes up in flames .
dress and jacket modelled at show .
a horse and jockey ride on the track
```

```
documents_split = list()
for i in range(N):
  documents_split.append(documents[i].split())
print(documents_split)
```

```
[['a', 'very', 'typical', 'bus', 'station'], ['sierra', 'looked', 'stunning', 'in', 'this', 'top', 'and', 'this', 'skirt', 'while', 'per
```

BAG OF WORDS

```
all_words = list()
for i in range(N):
  all_words = documents_split[i] + all_words
print(set(all_words))
```

```
{'60fps', 'rat', 'allowed', 'buildings', 'underground', 'been', 'wrap', 'fits', 'ufo', 'multicultural', 'captivity', 'ups', 'beside', 'p
```

```
def word_count(string_inp):
  my_string = string_inp.lower().split()
  my_dict = {}
  for item in set(all_words):
    my_dict[item] = my_string.count(item)
  return my_dict
```

```
documents[0]
```

```
'a very typical bus station'
```

```
print(word_count(documents[0]))
```

```
{'60fps': 0, 'rat': 0, 'allowed': 0, 'buildings': 0, 'underground': 0, 'been': 0, 'wrap': 0, 'fits': 0, 'ufo': 0, 'multicultural': 0, 'c
```

```
freq_dict = {}
```

```
#initialising the frequency deictionaty to store document frequency of terms
def freqq(string_inp):
  my_string = string_inp.lower().split()
  for item in set(all_words):
      freq_dict[item] = 0
```

```
def freq(string_inp):
  my_string = string_inp.lower().split()
  for item in set(all_words):
    if my_string.count(item) >= 1:
      freq_dict[item] = freq_dict[item] + 1


freqq(documents[0])
for i in range(N):
  freq(documents[i])


print(freq_dict)
```

```
{'60fps': 1, 'rat': 1, 'allowed': 1, 'buildings': 6, 'underground': 1, 'been': 12, 'wrap': 1, 'fits': 1, 'ufo': 1, 'multicultural': 1, '
```

calculating idf

```
import math
idf_dic = {}
for a,b in freq_dict.items():
  if b!=0:
    idf_dic[a] = math.log(N/b,2)
```

IDF - Values

```
print(idf_dic)
```

```
{'60fps': 10.965784284662087, 'rat': 10.965784284662087, 'allowed': 10.965784284662087, 'buildings': 8.380821783940931, 'underground': 1
```

```
tf_dict = list()
# wf_dict = list()
for i in range(N):
  tf_dict.append(word_count(documents[i]))
  # wf_dict.append(dict())
```

TERM FREQUENCIES

```
print(tf_dict[0])
```

```
{'60fps': 0, 'rat': 0, 'allowed': 0, 'buildings': 0, 'underground': 0, 'been': 0, 'wrap': 0, 'fits': 0, 'ufo': 0, 'multicultural': 0, 'c
```

```
import copy
weight =  copy.deepcopy(tf_dict)
```

Calculating tf-idf

```
i = 0
for i in range(len(weight)):
  for a,b in tf_dict[i].items():
    weight[i][a] = b * idf_dic[a]


print(weight[2])
```

```
{'60fps': 0.0, 'rat': 0.0, 'allowed': 0.0, 'buildings': 0.0, 'underground': 0.0, 'been': 0.0, 'wrap': 0.0, 'fits': 0.0, 'ufo': 0.0, 'mul
```

Query part

```
print("Enter the query:")
query = input()
```

```
Enter the query:
hello
```

```
tf_query = word_count(query)
```

Tf for the query

```
print(tf_query)
```

```
{'60fps': 0, 'rat': 0, 'allowed': 0, 'buildings': 0, 'underground': 0, 'been': 0, 'wrap': 0, 'fits': 0, 'ufo': 0, 'multicultural': 0, 'c
```

```
wf_query = copy.deepcopy(tf_query)
```

```
for a,b in tf_query.items():
    wf_query[a] = b * idf_dic[a]
```

```
print(wf_query)
```

```
{'60fps': 0.0, 'rat': 0.0, 'allowed': 0.0, 'buildings': 0.0, 'underground': 0.0, 'been': 0.0, 'wrap': 0.0, 'fits': 0.0, 'ufo': 0.0, 'mul
```

```
def distance_comp(rn_dict):
  dist = 0
  for a,b in wf_query.items():
    dist = dist + pow(b - rn_dict[a] , 2)
  return math.sqrt(dist)
```

```
dist = list()
for i in range(N):
  #print(weight[i])
  dist.append(distance_comp(weight[i]))
```

Euclidean Distance between other docs to the query

```
print(dist)
```

```
[17.505884307093687, 30.499421515510733, 19.942606018800824, 21.794360623348993, 16.587448620243578, 26.862598505572326, 27.325719237888
```

```
similarity_list = list()
for i in range(N):
  similarity_list.append(1 / (1 + dist[i]))
```

Similarity between docs and the query

```
print(similarity_list)
```

```
[0.0540368665125978, 0.03174661475949921, 0.04774954936851074, 0.04387050009973379, 0.05685873042716246, 0.03589040698411553, 0.0353036
```

Top 10 Ranked documents for the given query using euclidean similarity measure

```
print("GIVEN QUERY - ",query)
print()
print("--------Top 10 ranked documents-------")
print()
import numpy
array_similarity_list = numpy.array(similarity_list)
sort_index = numpy.argsort(array_similarity_list)
# print(sort_index)
for i in range(len(sort_index)-1,len(sort_index)-11,-1):
  # print(sort_index[i])
  print(df.iloc[sort_index[i],0],"- index - ",sort_index[i])
  print(df.iloc[sort_index[i],1])
  print()
```

```
GIVEN QUERY -  hello

--------Top 10 ranked documents-------

person arrives to the premiere - index -  264
व्यक्ति प्रीमियर के लिए आता है

person arrives at the premiere . - index -  1066
व्यक्ति प्रीमियर पर आता है।

the sign for a city . - index -  139
एक शहर के लिए संकेत।

a city in the mountains - index -  1055
पहाड़ों में एक शहर

actor arrives to the premiere - index -  30
अभिनेता प्रीमियर के लिए आता है

actor arrives at the premiere - index -  1755
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  621
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  946
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  220
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  1819
अभिनेता प्रीमियर पर आता है
```

COSINE SIMILARITY

```
from scipy import spatial

similarity_list_cosine = list()
for i in range(N):
  similarity_list_cosine.append(1 - spatial.distance.cosine(list(wf_query.values()), list(weight[i].values())))
```

```
/usr/local/lib/python3.10/dist-packages/scipy/spatial/distance.py:636: RuntimeWarning: invalid value encountered in scalar divide
  dist = 1.0 - uv / np.sqrt(uu * vv)
```

```
print(similarity_list_cosine)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Top 10 Ranked documents using cosine similarity using cosine similarity

```
print("GIVEN QUERY - ",query)
print()
print("--------Top 10 ranked documents using cosine similarity-------")
print()
import numpy
array_similarity_list = numpy.array(similarity_list_cosine)
sort_index = numpy.argsort(similarity_list_cosine)
```

```
# print(sort_index)
for i in range(len(sort_index)-1,len(sort_index)-11,-1):
  # print(sort_index[i])
  print(df.iloc[sort_index[i],0],"-index - ",sort_index[i])
  print(df.iloc[sort_index[i],1])
  print()
```

```
GIVEN QUERY -  hello

--------Top 10 ranked documents using cosine similarity-------

a very typical bus station -index -  0
एक बहुत ही विशिष्ट बस स्टेशन

a horse and jockey ride on the track -index -  1999
ट्रैक पर एक घोड़ा और जॉकी की सवारी

dress and jacket modelled at show . -index -  1998
ड्रेस और जैकेट शो में मॉडलिंग।

singer 's tour bus goes up in flames . -index -  1997
गायक की टूर बस आग में जाती है।

firefighters stand on lanes of the freeway near smoldering hot spots of a large fire that consumed an under - construction apartment bui
अग्निशामक एक बड़ी आग के गर्म धब्बे को सुलझाने के पास फ्रीवे के लेन पर खड़े हैं जो एक कम-निर्माण अपार्टमेंट बिल्डिंग का उपभोग करते हैं।

get the scoop : read all about transformation in magazine -index -  1995
स्कूप प्राप्त करें: पत्रिका में परिवर्तन के बारे में सब कुछ पढ़ें

the previous logo , compared with the latest logo . -index -  1994
नवीनतम लोगो की तुलना में पिछले लोगो।

man with a briefcase in an airport . -index -  1993
एक हवाई अड्डे में एक ब्रीफकेस वाला आदमी।

actor and his son during the award red carpet arrivals -index -  1992
पुरस्कार रेड कार्पेट आगमन के दौरान अभिनेता और उसका बेटा

unearthed : thanks to science , we may see the rebirth of the chestnut -index -  1991
पता चला: विज्ञान के लिए धन्यवाद, हम चेस्टनट की पुनर्जन्म देख सकते हैं
```

Ranked retreival using log term weighting

```
log_tf_dict = []
```

```
def word_count_log(string_inp):
  my_string = string_inp.lower().split()
  my_dict = {}
  for item in set(all_words):
    if my_string.count(item) >= 1:
      my_dict[item] = 1 + math.log(my_string.count(item),2)
    else:
      my_dict[item] = 0
  return my_dict
```

```
log_tf_dict = list()
# wf_dict = list()
for i in range(N):
  log_tf_dict.append(word_count_log(documents[i]))
```

```
print(log_tf_dict[0])
```

```
{'60fps': 0, 'rat': 0, 'allowed': 0, 'buildings': 0, 'underground': 0, 'been': 0, 'wrap': 0, 'fits': 0, 'ufo': 0, 'multicultural': 0, 'c
```

```
import copy
log_weight =  copy.deepcopy(log_tf_dict)
```

```
print(idf_dic)
```

```
{'60fps': 10.965784284662087, 'rat': 10.965784284662087, 'allowed': 10.965784284662087, 'buildings': 8.380821783940931, 'underground': 1
```

```
print(log_tf_dict[0])
```

```
), 'rat': 0, 'allowed': 0, 'buildings': 0, 'underground': 0, 'been': 0, 'wrap': 0, 'fits': 0, 'ufo': 0, 'multicultural': 0, 'captivity':
```

```
i = 0
for i in range(len(log_weight)):
  for a,b in log_tf_dict[i].items():
    log_weight[i][a] = b * idf_dic[a]
```

```
print(log_tf_dict[0])
```

```
{'60fps': 0, 'rat': 0, 'allowed': 0, 'buildings': 0, 'underground': 0, 'been': 0, 'wrap': 0, 'fits': 0, 'ufo': 0, 'multicultural': 0, 'c
```

```
print(log_weight[2])
```

```
{'60fps': 0.0, 'rat': 0.0, 'allowed': 0.0, 'buildings': 0.0, 'underground': 0.0, 'been': 0.0, 'wrap': 0.0, 'fits': 0.0, 'ufo': 0.0, 'mul
```

Query part

```
log_tf_query = word_count_log(query)
```

```
print(log_tf_query)
```

```
{'60fps': 0, 'rat': 0, 'allowed': 0, 'buildings': 0, 'underground': 0, 'been': 0, 'wrap': 0, 'fits': 0, 'ufo': 0, 'multicultural': 0, 'c
```

```
log_wf_query = copy.deepcopy(log_tf_query)
```

```
for a,b in log_tf_query.items():
    log_wf_query[a] = b * idf_dic[a]
```

```
print(log_wf_query)
```

```
{'60fps': 0.0, 'rat': 0.0, 'allowed': 0.0, 'buildings': 0.0, 'underground': 0.0, 'been': 0.0, 'wrap': 0.0, 'fits': 0.0, 'ufo': 0.0, 'mul
```

```
def log_distance_comp(rn_dict):
  dist = 0
  for a,b in log_wf_query.items():
    dist = dist + pow(b - rn_dict[a] , 2)
  return math.sqrt(dist)
```

```
dist_log = list()
for i in range(N):
  dist_log.append(log_distance_comp(log_weight[i]))
```

Distance between other docs to the query

```
log_similarity_list = list()
for i in range(N):
  log_similarity_list.append(1 / (1 + dist_log[i]))
```

```
print(log_similarity_list)
```

```
[0.0540368665125978, 0.03174661475949921, 0.04774954936851074, 0.04387050009973379, 0.05685873042716246, 0.03589040698411553, 0.0353036
```

Top 10 Ranked documents for the given query using euclidean similarity measure

```
print("GIVEN QUERY - ",query)
print()
print("--------Top 10 ranked documents-------")
print()
import numpy
log_array_similarity_list = numpy.array(log_similarity_list)
log_sort_index = numpy.argsort(log_array_similarity_list)
# print(sort_index)
for i in range(len(log_sort_index)-1,len(log_sort_index)-11,-1):
  # print(sort_index[i])
  print(df.iloc[log_sort_index[i],0],"- index - ",log_sort_index[i])
  print(df.iloc[log_sort_index[i],1])
  print()
```

```
GIVEN QUERY -  hello

--------Top 10 ranked documents-------

person arrives to the premiere - index -  264
व्यक्ति प्रीमियर के लिए आता है

person arrives at the premiere . - index -  1066
व्यक्ति प्रीमियर पर आता है।

the sign for a city . - index -  139
एक शहर के लिए संकेत।

a city in the mountains - index -  1055
पहाड़ों में एक शहर

actor arrives to the premiere - index -  30
अभिनेता प्रीमियर के लिए आता है

actor arrives at the premiere - index -  1755
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  621
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  946
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  220
अभिनेता प्रीमियर पर आता है

actor arrives at the premiere - index -  1819
अभिनेता प्रीमियर पर आता है
```

Cosine Similarity

```
from scipy import spatial

similarity_list_log_cosine = list()
for i in range(N):
  similarity_list_log_cosine.append(1 - spatial.distance.cosine(list(log_wf_query.values()), list(log_weight[i].values())))
```

```
print(similarity_list_log_cosine)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
print("GIVEN QUERY - ",query)
print()
print("--------Top 10 ranked documents using cosine similarity-------")
print()
import numpy
array_similarity_list = numpy.array(similarity_list_log_cosine)
sort_index = numpy.argsort(similarity_list_log_cosine)
# print(sort_index)
for i in range(len(sort_index)-1,len(sort_index)-11,-1):
  # print(sort_index[i])
  print(df.iloc[sort_index[i],0],"-index - ",sort_index[i])
  print(df.iloc[sort_index[i],1])
  print()
```

```
GIVEN QUERY -  hello

--------Top 10 ranked documents using cosine similarity-------

a very typical bus station -index -  0
एक बहुत ही विशिष्ट बस स्टेशन

a horse and jockey ride on the track -index -  1999
ट्रैक पर एक घोड़ा और जॉकी की सवारी

dress and jacket modelled at show . -index -  1998
ड्रेस और जैकेट शो में मॉडलिंग।

singer 's tour bus goes up in flames . -index -  1997
गायक की टूर बस आग में जाती है।

firefighters stand on lanes of the freeway near smoldering hot spots of a large fire that consumed an under - construction apartment bui
अग्निशामक एक बड़ी आग के गर्म धब्बे को सुलझाने के पास फ्रीवे के लेन पर खड़े हैं जो एक कम-निर्माण अपार्टमेंट बिल्डिंग का उपभोग करते हैं।

get the scoop : read all about transformation in magazine -index -  1995
स्कूप प्राप्त करें: पत्रिका में परिवर्तन के बारे में सब कुछ पढ़ें

the previous logo , compared with the latest logo . -index -  1994
नवीनतम लोगो की तुलना में पिछले लोगो।

man with a briefcase in an airport . -index -  1993
एक हवाई अड्डे में एक ब्रीफकेस वाला आदमी।

actor and his son during the award red carpet arrivals -index -  1992
पुरस्कार रेड कार्पेट आगमन के दौरान अभिनेता और उसका बेटा

unearthed : thanks to science , we may see the rebirth of the chestnut -index -  1991
पता चला: विज्ञान के लिए धन्यवाद, हम चेस्टनट की पुनर्जन्म देख सकते हैं
```

Evaluation-Precision and Recall

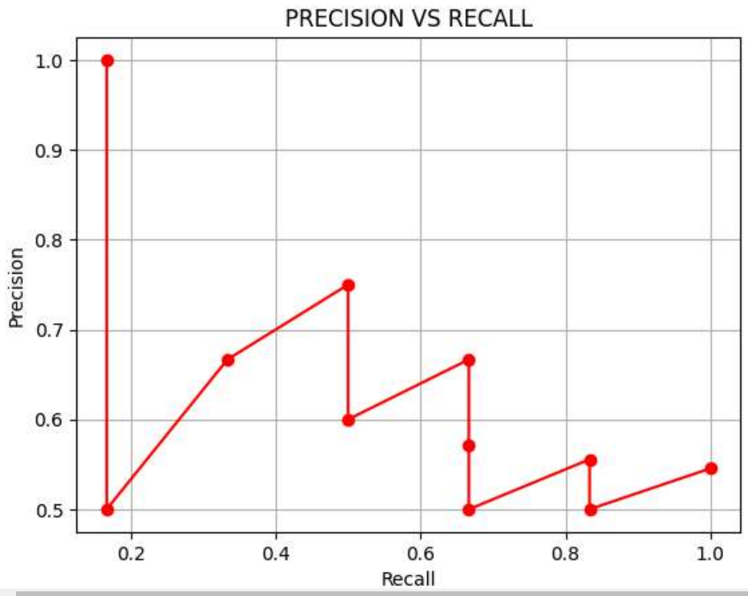
```
r=[]
p=[]
rcount=0
counter=1
pcount=0

def precisionrecall():
  l=list(input().split())

  count=0
  for i in l:
    if i=="R":
      count=count+1

  def recall(s,count):
    global rcount
    if s=="R":
      rcount=rcount+1
      r.append(rcount/count)
    else:
      r.append(rcount/count)
print("Enter the  Relevance for the query")
precisionrecall()
drawgraph()
```

```
Enter the  Relevance for the query
R N R R N R N N R N R
[0.16666666666666666, 0.16666666666666666, 0.3333333333333333, 0.5, 0.5, 0.6666666666666666, 0.6666666666666666, 0.6666666666666666, 0.8
[1.0, 0.5, 0.6666666666666666, 0.75, 0.6, 0.6666666666666666, 0.5714285714285714, 0.5, 0.5555555555555556, 0.5, 0.5454545454545454]
```


PRECISION VS RECALL
Precision
Recall