**תרגיל בית 2 אחזור מידע:**

**מגישים:**

**שם הקבוצה : Rank Rangers**

שיר כהן, ת.ז. 316216340

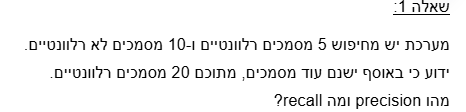
אלמוג קדוש, ת.ז. 315699439

עמית וינוגרד, ת.ז. 316597723

אלמוג אלבז, ת.ז. 213037369

**קישור לgithub:**

<https://github.com/cohensh96/Rank-Rangers--Information-Retrieval>



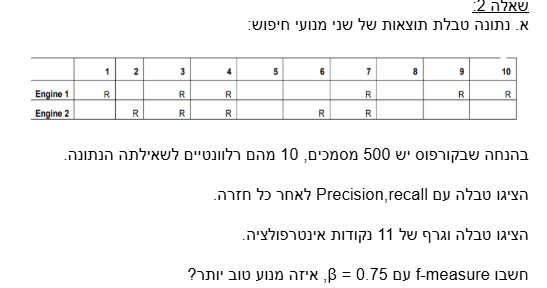
חישוב Percision:

מספר המסמכים הרלוונטיים שהוחזרו על ידי המערכת (5) חלקי סך כל המסמכים שהוחזרו על ידי המערכת (15) ולכן 0.33

חישוב Recall:

מספר המסמכים הרלוונטיים שהוחזרו על ידי המערכת (5) חלקי סך כל המסמכים הרלוונטיים (25) (בהרצאה נאמר שה20 מסמכים הנוספים הם מסמכים רלוונטיים שלא הוחזרו) לכן 5/25=0.2

2.



במנוע חיפוש 1 התוצאות הרלוונטיות הם במיקומים: 1,3,4,7,9,10

במנוע חיפוש 2 התוצאות הרלוונטיות הם במיקומים: 2,3,4,6,7

תוצאות ה-percision, recall עבור כל מנוע חיפוש:

מנוע חיפוש 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| מספר מסמך | סך הכל רלוונטיים | סך הכל הוחזרו | |  | | --- | |  |   percision | |  | | --- | |  |   recall |
| 1 | 1 | 1 | 1/1 = 1 | 1/10 = 0.1 |
| 2 | 1 | 2 | 1/2 = 0.5 | 1/10 = 0.1 |
| 3 | 2 | 3 | 2/3 = 0.67 | 2/10 = 0.2 |
| 4 | 3 | 4 | 3/4 = 0.75 | 3/10 = 0.3 |
| 5 | 3 | 5 | 3/5 = 0.6 | 3/10 = 0.3 |
| 6 | 3 | 6 | 3/6 = 0.5 | 3/10 = 0.3 |
| 7 | 4 | 7 | 4/7 = 0.57 | 4/10 = 0.4 |
| 8 | 4 | 8 | 4/8 = 0.5 | 4/10 = 0.4 |
| 9 | 5 | 9 | 5/9 = 0.56 | 5/10 = 0.5 |
| 10 | 6 | 10 | 6/10 = 0.6 | 6/10 = 0.6 |

מנוע חיפוש 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| מספר מסמך | סך הכל רלוונטיים | סך הכל הוחזרו | |  | | --- | |  |   percision | |  | | --- | |  |   recall |
| 1 | 0 | 1 | 0/1 = 0 | 0/10 = 0 |
| 2 | 1 | 2 | 1/2 = 0.5 | 1/10 = 0.1 |
| 3 | 2 | 3 | 2/3 = 0.67 | 2/10 = 0.2 |
| 4 | 3 | 4 | 3/4 = 0.75 | 3/10 = 0.3 |
| 5 | 3 | 5 | 3/5 = 0.6 | 3/10 = 0.3 |
| 6 | 4 | 6 | 4/6 = 0.67 | 4/10 = 0.4 |
| 7 | 5 | 7 | 5/7 = 0.71 | 5/10 = 0.5 |
| 8 | 5 | 8 | 5/8 = 0.63 | 5/10 = 0.5 |
| 9 | 5 | 9 | 5/9 = 0.56 | 5/10 = 0.5 |
| 10 | 5 | 10 | 5/10 = 0.5 | 5/10 = 0.5 |

כדי לחשב את הinterpolated precision:

נגדיר 11 רמות Recall שהן: 0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0

נחשב את ה- הinterpolated precision עבור כל רמת Recall.

בכל אחת מהנקודות הללו נקח את הערך המקסימלי של ה-precision שנמצא עבור ערך recall השווה או גדול לנקודה זו.

קוד פייתוני מצורף:

import matplotlib.pyplot as plt

recall = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

precision\_engine1 = [1, 1, 0.67, 0.75, 0.57, 0.56, 0.6, 0, 0, 0, 0]

precision\_engine2 = [0.75, 0.5, 0.67, 0.75, 0.67, 0.71, 0, 0, 0, 0, 0]

plt.figure(figsize=(8, 6))

plt.plot(recall, precision\_engine1, marker='o', label='Engine 1')

plt.plot(recall, precision\_engine2, marker='\*', label='Engine 2')

plt.title('11-point Interpolated Average Precision-Recall Curve')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.legend()

plt.grid()

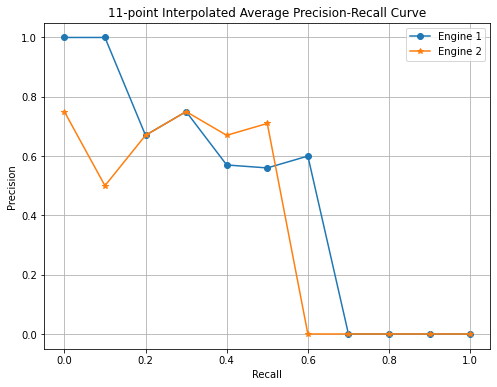
plt.show()

מנוע חיפוש 1: מנוע חיפוש 2:

|  |  |  |
| --- | --- | --- |
| Interpolated Percision | |  | | --- | |  |   Recall |
| 1 | 0 |
| 1 | 0.1 |
| 0.67 | 0.2 |
| 0.75 | 0.3 |
| 0.57 | 0.4 |
| 0.56 | 0.5 |
| 0.6 | 0.6 |
| 0 | 0.7 |
| 0 | 0.8 |
| 0 | 0.9 |
| 0 | 1.0 |

|  |  |  |
| --- | --- | --- |
| Interpolated Percision | |  | | --- | |  |   Recall |
| 0.75 | 0 |
| 0.5 | 0.1 |
| 0.67 | 0.2 |
| 0.75 | 0.3 |
| 0.67 | 0.4 |
| 0.71 | 0.5 |
| 0 | 0.6 |
| 0 | 0.7 |
| 0 | 0.8 |
| 0 | 0.9 |
| 0 | 1.0 |

וזה הגרף שקיבלנו:



נחשב את הf-measure :

מנוע חיפוש 1:

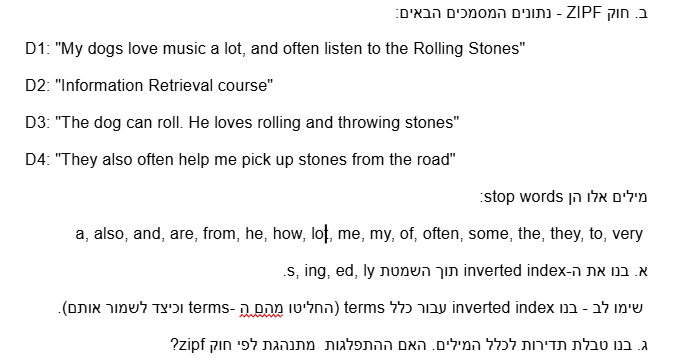
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | סך הכל רלוונטיים | סך הכל הוחזרו | percision | Recall | F measure |
| 1 | 1 | 1 | 1/1 = 1 | 1/10 = 0.1 | 0.2358 |
| 2 | 1 | 2 | 1/2 = 0.5 | 1/10 = 0.1 | 0.2049 |
| 3 | 2 | 3 | 2/3 = 0.67 | 2/10 = 0.2 | 0.3629 |
| 4 | 3 | 4 | 3/4 = 0.75 | 3/10 = 0.3 | 0.4870 |
| 5 | 3 | 5 | 3/5 = 0.6 | 3/10 = 0.3 | 0.4412 |
| 6 | 3 | 6 | 3/6 = 0.5 | 3/10 = 0.3 | 0.4032 |
| 7 | 4 | 7 | 4/7 = 0.57 | 4/10 = 0.4 | 0.4944 |
| 8 | 4 | 8 | 4/8 = 0.5 | 4/10 = 0.4 | 0.4587 |
| 9 | 5 | 9 | 5/9 = 0.56 | 5/10 = 0.5 | 0.5368 |
| 10 | 6 | 10 | 6/10 = 0.6 | 6/10 = 0.6 | 0.6 |
| AVG |  |  |  |  | 0.422 |

מנוע חיפוש 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | סך הכל רלוונטיים | סך הכל הוחזרו | percision | Recall | F measure |
| 1 | 0 | 1 | 0/1 = 0 | 0/10 = 0 | 0/0=X |
| 2 | 1 | 2 | 1/2 = 0.5 | 1/10 = 0.1 | 0.2049 |
| 3 | 2 | 3 | 2/3 = 0.67 | 2/10 = 0.2 | 0.3629 |
| 4 | 3 | 4 | 3/4 = 0.75 | 3/10 = 0.3 | 0.4870 |
| 5 | 3 | 5 | 3/5 = 0.6 | 3/10 = 0.3 | 0.4412 |
| 6 | 4 | 6 | 4/6 = 0.67 | 4/10 = 0.4 | 0.5390 |
| 7 | 5 | 7 | 5/7 = 0.71 | 5/10 = 0.5 | 0.6167 |
| 8 | 5 | 8 | 5/8 = 0.63 | 5/10 = 0.5 | 0.5761 |
| 9 | 5 | 9 | 5/9 = 0.56 | 5/10 = 0.5 | 0.5368 |
| 10 | 5 | 10 | 5/10 = 0.5 | 5/10 = 0.5 | 0.5 |
| AVG |  |  |  |  | 0.473 |

f-measure\_engine2 f-measure\_engine1 <

לפי זה ניתן לראות כי מנוע 2 טוב יותר ממנוע 1.

**שאלה 2:**  
  


נסיר את הstop words וגם את סיומות s, ing, ed, ly:

“D1: “dog love music, listen Roll Stone"

“D2: “Information Retrieval course:

“D3: “dog can roll. He love roll throw stone"

“D4: “ help pick up stone from road"

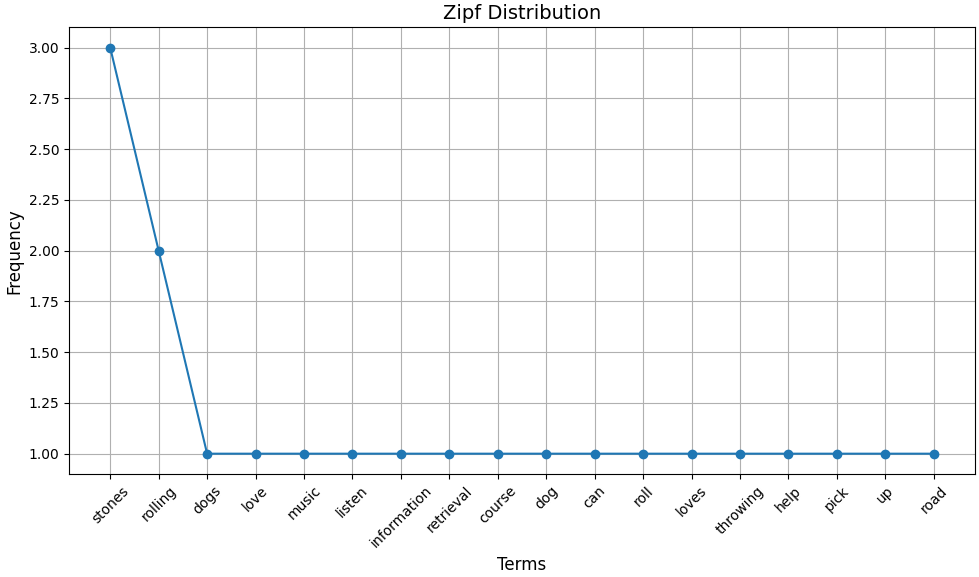
סעיף א:

|  |  |  |
| --- | --- | --- |
| Frequency: | Shows in: | Term |
| 2 | D1,D3 | Dog |
| 2 | D1,D3 | Love |
| 1 | D1 | Music |
| 1 | D1 | Listen |
| 3 | D1,D3 | Roll |
| 3 | D1,D3,D4 | Stone |
| 1 | D2 | Information |
| 1 | D2 | Retrieval |
| 1 | D2 | Course |
| 1 | D3 | Can |
| 1 | D3 | Throw |
| 1 | D4 | Help |
| 1 | D4 | Pick |
| 1 | D4 | Up |
| 1 | D4 | Road |

עבור כל term:

|  |  |  |  |
| --- | --- | --- | --- |
| Frequency | Save as: | Shows in: | Term |
| 1 | Dog | D1 | Dogs |
| 1 |  | D1 | Love |
| 1 |  | D1 | Music |
| 1 |  | D1 | Listen |
| 2 | Roll | D1,D3 | Rolling |
| 3 | Stone | D1,D3,D4 | Stones |
| 1 |  | D2 | Information |
| 1 |  | D2 | Retrieval |
| 1 |  | D2 | Course |
| 1 |  | D3 | Dog |
| 1 |  | D3 | Can |
| 1 |  | D3 | Roll |
| 1 | Love | D3 | Loves |
| 1 | Throw | D3 | Throwing |
| 1 |  | D4 | Help |
| 1 |  | D4 | Pick |
| 1 |  | D4 | up |
| 1 |  | D4 | Road |

כדי לבדוק האם ההתפלגות מתאימה לחוק זיפ נבנה את הגרף בפייתון:



נראה שלא תואם לחוק זיפ כי היינו מצפים לראות קו ישר יורד לאורך כל הדרך ואילו כאן אנו רואים ירידה מאוד חדה ואז קו שטוח לגמרי.

קוד מצורף לבניית הגרף:

import re

from collections import Counter

import matplotlib.pyplot as plt

documents = [

    "My dogs love music a lot, and often listen to the Rolling Stones",

    "Information Retrieval course",

    "The dog can roll. He loves rolling and throwing stones",

    "They also often help me pick up stones from the road"]

stop\_words = [

    "a", "also", "and", "are", "from", "he", "how", "lot", "me", "my", "of", "often", "some", "the", "they", "to", "very"]

def clean\_and\_tokenize(text):

    text = text.lower()

    text = re.sub(r"[^a-z\s]", "", text)

    words = text.split()

    return [word for word in words if word not in stop\_words]

all\_words = []

for doc in documents:

    all\_words.extend(clean\_and\_tokenize(doc))

word\_counts = Counter(all\_words)

sorted\_word\_counts = word\_counts.most\_common()

words, frequencies = zip(\*sorted\_word\_counts)

plt.figure(figsize=(10, 6))

plt.plot(range(1, len(frequencies) + 1), frequencies, marker="o")

plt.title("Zipf Distribution", fontsize=14)

plt.xlabel("Terms", fontsize=12)

plt.ylabel("Frequency", fontsize=12)

plt.xticks(range(1, len(frequencies) + 1), words, rotation=45)

plt.grid(True)

plt.tight\_layout()

plt.show()

שאלה 3: קדם פרויקט –בניית זחלן

הריצו את החיפוש והחזירו את התוצאות בקובץ אקסל, עם כל השדות הרלוונטיים.

כעת, עליכם לבנות אינדקס לתוצאות שחזרו.

1. בנו inverted index ל-15 מילים הנפוצות ביותר שחזרו. במידה והמילים הנפוצות מופיעות ביותר מ-20 דפים, התייחסו רק ל- 20 הדפים הראשונים שחזרו.
2. בחרו את אחת השאילתות בה עשיתם שימוש, וחשבו tf-idf של המושגים בשאילתא, על פני כל הדפים שחזרו. פרטו את כל שלבי החישוב.
3. הציגו את 10 התוצאות הראשונות שחזרו עבור השאילתה ל-2 משתמשים שונים. בקשו מכל משתמש להחליט עבור כל תוצאה האם היא רלוונטית. חשבו precision ו-recall.

**שאלה 3:**

**סעיף 1:**

**הפתרון נמצא תחת לשונית Top 15 Inverted Index בקובץ אקסל.**

**קוד התוכנית :**

# -\*- coding: utf-8 -\*-

import requests

from bs4 import BeautifulSoup

import re

from nltk.stem import PorterStemmer

from urllib.parse import urljoin

import time

from collections import defaultdict

import math # For log in IDF

import pandas as pd

class SearchEngine:

def \_\_init\_\_(self):

self.stemmer = PorterStemmer()

self.stop\_words = {

'a', 'about', 'above', 'after', 'again', 'against', 'all', 'am', 'an', 'and', 'any',

'are', 'aren', 'as', 'at', 'be', 'because', 'been', 'before', 'being', 'below', 'between',

'both', 'but', 'by', 'can', 'could', 'did', 'do', 'does', 'doing', 'don', 'down', 'during',

'each', 'few', 'for', 'from', 'further', 'had', 'has', 'have', 'having', 'he', 'her',

'here', 'hers', 'herself', 'him', 'himself', 'his', 'how', 'i', 'if', 'in', 'into', 'is',

'it', 'its', 'itself', 'just', 'me', 'more', 'most', 'my', 'myself', 'no', 'nor', 'not',

'now', 'of', 'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours', 'ourselves', 'out',

'over', 'own', 's', 'same', 'she', 'should', 'so', 'some', 'such', 'than', 'that', 'the',

'their', 'theirs', 'them', 'themselves', 'then', 'there', 'these', 'they', 'this', 'those',

'through', 'to', 'too', 'under', 'until', 'up', 'very', 'was', 'we', 'were', 'what', 'when',

'where', 'which', 'while', 'who', 'whom', 'why', 'will', 'with', 'you', 'your', 'yours',

'yourself', 'yourselves'

}

self.url\_to\_id = {}

self.id\_to\_url = {}

self.next\_id = 1

# inverted\_index[word][doc\_id] = count of word in doc

self.inverted\_index = defaultdict(lambda: defaultdict(int))

# global\_index[word][url] = count of word in url

self.global\_index = defaultdict(lambda: defaultdict(int))

def get\_url\_id(self, url):

"""Map URLs to unique IDs"""

if url not in self.url\_to\_id:

self.url\_to\_id[url] = self.next\_id

self.id\_to\_url[self.next\_id] = url

self.next\_id += 1

return self.url\_to\_id[url]

def fetch\_page(self, url):

"""Fetch the HTML page content"""

try:

time.sleep(1)

response = requests.get(url, timeout=10)

if response.status\_code == 200:

return BeautifulSoup(response.text, 'html.parser')

else:

print(f"Failed to fetch {url}: Status code {response.status\_code}")

return None

except Exception as e:

print(f"Error fetching {url}: {str(e)}")

return None

def get\_links(self, soup, base\_url):

"""Extract links from a page"""

links = set()

if soup:

for link in soup.find\_all('a', href=True):

url = link['href']

absolute\_url = urljoin(base\_url, url)

if absolute\_url.startswith(base\_url):

links.add(absolute\_url)

return links

def index\_words(self, soup, url):

"""

Index words from a page, storing the stemmed form.

Returns a dictionary of {word: count\_in\_this\_document}.

"""

index = defaultdict(int)

url\_id = self.get\_url\_id(url)

if soup:

words = re.findall(r'\w+', soup.get\_text())

for word in words:

word = word.lower()

if word not in self.stop\_words:

stemmed\_word = self.stemmer.stem(word)

# Increase counters in the local index (for this page)

index[stemmed\_word] += 1

# Update the global/inverted index

self.inverted\_index[stemmed\_word][url\_id] += 1

self.global\_index[stemmed\_word][url] += 1

return dict(index)

def crawl\_and\_index(self, start\_url, max\_pages=5):

"""Crawl and index pages. Returns a dict: {doc\_id: {word: count}}"""

visited = set()

to\_visit = {start\_url}

page\_indexes = {}

while to\_visit and len(visited) < max\_pages:

url = to\_visit.pop()

if url in visited:

continue

print(f"Crawling: {url} (ID: {self.get\_url\_id(url)})")

soup = self.fetch\_page(url)

if soup:

page\_indexes[self.get\_url\_id(url)] = self.index\_words(soup, url)

visited.add(url)

links = self.get\_links(soup, start\_url)

to\_visit.update(links - visited)

return page\_indexes

def save\_to\_excel(self, page\_indexes, filename="results.xlsx"):

"""

Save:

1) URL Mappings

2) Page Word Counts

3) Top 15 Inverted Index

"""

# 1) URL Mappings sheet

url\_mappings = [{"URL ID": url\_id, "URL": url} for url, url\_id in self.url\_to\_id.items()]

url\_df = pd.DataFrame(url\_mappings)

# 2) Page Word Counts sheet

rows = []

for url\_id, index in page\_indexes.items():

for word, count in index.items():

rows.append({"URL ID": url\_id, "Word": word, "Count": count})

counts\_df = pd.DataFrame(rows)

# 3) Calculate global word frequencies to find top 15

word\_frequencies = defaultdict(int)

for word, docs in self.inverted\_index.items():

word\_frequencies[word] = sum(docs.values())

top\_words = sorted(word\_frequencies.items(), key=lambda x: x[1], reverse=True)[:15]

top\_words\_set = {word for word, \_ in top\_words}

# Filter the inverted index for the top 15 words

inverted\_rows = []

for word, docs in self.inverted\_index.items():

if word in top\_words\_set:

document\_ids = ", ".join(map(str, docs.keys()))

inverted\_rows.append({"Word": word, "Document IDs": document\_ids})

inverted\_df = pd.DataFrame(inverted\_rows)

# Save to Excel (initial sheets)

with pd.ExcelWriter(filename, engine="openpyxl") as writer:

url\_df.to\_excel(writer, index=False, sheet\_name="URL Mappings")

counts\_df.to\_excel(writer, index=False, sheet\_name="Page Word Counts")

inverted\_df.to\_excel(writer, index=False, sheet\_name="Top 15 Inverted Index")

print(f"Results saved to '{filename}'.")

def calculate\_tf\_idf\_for\_query(self, query, page\_indexes, filename="results.xlsx"):

"""

1) Preprocess query (tokenize, remove stopwords, stemming).

2) For each query word (concept), for each document:

- Compute TF

- Compute IDF

- Compute TF-IDF

3) Save detailed results (doc\_id, url, concept, TF, IDF, TF-IDF) to a new sheet in the Excel file

with 10 decimal places.

"""

# 1) Preprocess the query

query\_words = re.findall(r'\w+', query.lower())

processed\_query = []

for qw in query\_words:

if qw not in self.stop\_words:

stemmed\_qw = self.stemmer.stem(qw)

processed\_query.append(stemmed\_qw)

# 2) Collect stats for TF-IDF

N = len(page\_indexes) # total number of documents

doc\_total\_words = {}

for doc\_id, index\_dict in page\_indexes.items():

total\_count = sum(index\_dict.values())

doc\_total\_words[doc\_id] = total\_count

# 3) Compute TF, IDF, TF-IDF

tf\_idf\_rows = []

for doc\_id, index\_dict in page\_indexes.items():

url = self.id\_to\_url[doc\_id]

total\_words\_in\_doc = doc\_total\_words[doc\_id]

for qw in processed\_query:

freq\_in\_doc = self.inverted\_index[qw].get(doc\_id, 0)

tf = freq\_in\_doc / total\_words\_in\_doc if total\_words\_in\_doc > 0 else 0.0

df = len(self.inverted\_index[qw].keys()) # number of docs containing qw

idf = math.log(N / df, 10) if df > 0 else 0.0

tf\_idf = tf \* idf

tf\_idf\_rows.append({

"Document ID": doc\_id,

"URL": url,

"Query Concept": qw,

"TF": tf,

"IDF": idf,

"TF-IDF": tf\_idf

})

# 4) Create a DataFrame and append to the existing Excel file,

# ensuring 10 decimal places in numeric columns.

tf\_idf\_df = pd.DataFrame(tf\_idf\_rows)

with pd.ExcelWriter(filename, engine="openpyxl", mode="a", if\_sheet\_exists="replace") as writer:

# The 'float\_format' parameter ensures up to 10 digits after the decimal point

tf\_idf\_df.to\_excel(

writer,

index=False,

sheet\_name="Query TF-IDF",

float\_format="%.10f" # ensures 10 places

)

print("TF-IDF results for the query have been appended to the Excel file.")

def main():

search\_engine = SearchEngine()

# 1) Crawl up to 20 pages

start\_url = "https://www.bundesgesundheitsministerium.de/en/"

page\_indexes = search\_engine.crawl\_and\_index(start\_url, max\_pages=20)

# 2) Save the initial results

search\_engine.save\_to\_excel(page\_indexes, "results.xlsx")

# 3) Define the query

user\_query = "What are the average waiting times in hospitals in Germany by region?"

# 4) Calculate TF-IDF for that query, saving results with 10 decimal precision

search\_engine.calculate\_tf\_idf\_for\_query(user\_query, page\_indexes, "results.xlsx")

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

main()

**סעיף 2:**

השאילתה היא:

" What are the average waiting times in hospitals in Germany by region?”

**שלב ראשון:**

פירוק לtoken :

[What, are, the, average, waiting, times, in, hospitals, in, Germany, by, region]

**שלב שני:**

Stop words:

[average, waiting, times, hospitals, Germany, region]

**שלב שלישי:**

Stemming:

[average, wait, time, hospital, Germany, region]

**שלב רביעי:**

ביצענו את חישובי הTF-IDF בקוד כולל פירוט השלבים כתיעוד והתוצאה נשמרה בקובץ אקסל בלשונית QUERY TF-IDF.

תחילה חישבנו את ערכי הtf עבור כל מילה בכל מסמך ע״י החישוב הבא :

*Tf=מספר המילים שמילה מופיעה במסמך חלקי סך כל המילים במסמך*

*לאחר מכן חישבנו את הidf ע״י החישוב הבא :*

*Log10(מספר המסמכים חלקי מספר המסמכים שבהם המילה מופיעה)*

לבסוף חישבנו את הtf-idf עי הכפלת הtf בidf.

ניתן לראות את המימוש בקוד המועתק בסעיף 1.

**סעיף 3:**

**10 המסמכים הראשונים שחזרו הינם:**

|  |  |
| --- | --- |
| **URL ID** | **URL** |
| 1 | https://www.bundesgesundheitsministerium.de/en/ |
| 2 | https://www.bundesgesundheitsministerium.de/en/international/european-health-policy.html |
| 3 | https://www.bundesgesundheitsministerium.de/en/ministry/leadership/st-dr-tomas-steffen.html |
| 4 | https://www.bundesgesundheitsministerium.de/en/topics.html |
| 5 | https://www.bundesgesundheitsministerium.de/en/#c29679 |
| 6 | https://www.bundesgesundheitsministerium.de/en/topics/digitalisation/digitalisation-in-healthcare.html |
| 7 | https://www.bundesgesundheitsministerium.de/en/#c27721 |
| 8 | https://www.bundesgesundheitsministerium.de/en/#c31215 |
| 9 | https://www.bundesgesundheitsministerium.de/en/data-protection.html |
| 10 | https://www.bundesgesundheitsministerium.de/en/#c29680 |

משתמש 1:

משתמש1 סימן את ה-5 דפים הממורקרים הבאים כרלוונטים מתוך כל ה10 דפים הראשונים שחזרו:

|  |  |
| --- | --- |
| **URL ID** | **URL** |
| 1 | https://www.bundesgesundheitsministerium.de/en/ |
| 2 | https://www.bundesgesundheitsministerium.de/en/international/european-health-policy.html |
| 3 | https://www.bundesgesundheitsministerium.de/en/ministry/leadership/st-dr-tomas-steffen.html |
| 4 | https://www.bundesgesundheitsministerium.de/en/topics.html |
| 5 | https://www.bundesgesundheitsministerium.de/en/#c29679 |
| 6 | https://www.bundesgesundheitsministerium.de/en/topics/digitalisation/digitalisation-in-healthcare.html |
| 7 | https://www.bundesgesundheitsministerium.de/en/#c27721 |
| 8 | https://www.bundesgesundheitsministerium.de/en/#c31215 |
| 9 | https://www.bundesgesundheitsministerium.de/en/data-protection.html |
| 10 | https://www.bundesgesundheitsministerium.de/en/#c29680 |

משתמש 2:

משתמש2 סימן את ה-7 דפים הממורקרים הבאים כרלוונטים מתוך כל ה10 דפים הראשונים שחזרו:

|  |  |
| --- | --- |
| **URL ID** | **URL** |
| 1 | https://www.bundesgesundheitsministerium.de/en/ |
| 2 | https://www.bundesgesundheitsministerium.de/en/international/european-health-policy.html |
| 3 | https://www.bundesgesundheitsministerium.de/en/ministry/leadership/st-dr-tomas-steffen.html |
| 4 | https://www.bundesgesundheitsministerium.de/en/topics.html |
| 5 | https://www.bundesgesundheitsministerium.de/en/#c29679 |
| 6 | https://www.bundesgesundheitsministerium.de/en/topics/digitalisation/digitalisation-in-healthcare.html |
| 7 | https://www.bundesgesundheitsministerium.de/en/#c27721 |
| 8 | https://www.bundesgesundheitsministerium.de/en/#c31215 |
| 9 | https://www.bundesgesundheitsministerium.de/en/data-protection.html |
| 10 | https://www.bundesgesundheitsministerium.de/en/#c29680 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | מסמך |
| R | R |  |  | R |  |  |  | R | R | משתמש1 |
| R | R | R |  | R |  | R |  | R |  | משתמש2 |

עבור משתמש 1:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | מסמך |
| 5/10 | 4/9 | 3/8 | 3/7 | 3/6 | 2/5 | 2/4 | 2/3 | 2/2 | 1/1 | Precision |
| 5/10 | 4/10 | 3/10 | 3/10 | 3/10 | 2/10 | 2/10 | 2/10 | 2/10 | 1/10 | Recall |

עבור משתמש 2:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | מסמך |
| /106 | /95 | /84 | 3/7 | 3/6 | 2/5 | /42 | 1/3 | 1/2 | 0/1 | Precision |
| 6/10 | 5/10 | 4/10 | 3/10 | 3/10 | 2/10 | 2/10 | 1/10 | 1/10 | 0/10 | Recall |