Reddit Pioneer

Reddit Pioneer Searcher

אתר מסכם פרוייקט

מגישים:

דניאל פריימוביץ, ניצן עזרא, ליאור בוזגלו, עומר זומרשטיין

**1. מהו תחום העיסוק המרכזי של האתר? מהו המידע הזמין למשתמשי האתר? ענו בפסקה אחת. צרפו את הקישור לאתר.**

אנו מתמקדים בבניית זחלן לאתר האינטרנטי Reddit, אתר פורומים מוביל עבור מגוון רחב של נושאים.

כל אדם שנכנס יוצר לעצמו משתמש, יכול לבצע חיפושים ולהגיב על פוסטים, בהתבסס על מידע זה אנו ניצור שאילתות

בזמן אמת.

האתר מתמקד באיסוף מידע מתתי-פורומים שונים של Reddit, במטרה להציג למשתמשים תוצאות ותכנים לפי מילות מפתח או או שאילתה.

באתר ניתן למצוא מגוון פוסטים ודיונים, אשר עוברים עיבוד וסינון כדי להבליט את התכנים הרלוונטיים ביותר למשתמש—למשל,

פוסטים בעלי מעורבות גבוהה (לייקים, תגובות), וכן פוסטים המקושרים לתת-פורומים מובילים (Subreddits).

Reddit Pioneer Searcher

**2. רשמו שלוש שאילתות מעניינות שהייתם רוצים לקבל עליהן תשובה באתר, והאתר אינו עונה עליהן כעת. עבור כל שאילתא כזו, ציינו מהם פרטי המידע הנדרשים לצורך מענה על השאילתא.**

שאילתה 1:

“איזה פוסטים על חתולים מצחיקים קיימים מהחודש האחרון תחת קטגוריה HOT."

(בreddit אין אופציה לבצע את החיפוש הזה באופן ישיר)

פרטי מידע נדרשים:

פילוח נושאי פוסטים לפי מילת מפתח (pets/cats).

ספירת פוסטים לכל Subreddit בתחום החתולים.

נתונים לגבי תדירות הפרסום ומועדי הפרסום בכל Subreddit.

כמות לייקים ותגובות לכל פוסט

שאילתה 2:

“אילו פוסטים זכו ליחס הגבוה ביותר בין לייקים (Upvotes) לתגובות, בנושא מסויימים?”

פרטי מידע נדרשים:

איתור פוסטים רלוונטיים למילות מפתח כגון “חתולים מצחיקים”, “חיות מצחיקות” או “דברים מצחיקים”.

נתוני לייקים (Upvotes) ונתוני תגובות (Comments) עבור הפוסטים שנמצאו.

חישוב היחס (לייקים/תגובות) לכל פוסט.

דירוג הפוסטים מהיחס הגבוה ביותר לנמוך ביותר.

שאילתה 3:

“מה הם הפוסטים על חתולים מצחיקים שעוררו הכי הרבה מחלוקות היום. ”

פרטי מידע נדרשים:

פילוח פוסטים לפי מילת מפתח (cats/funny).

נתוני לייקים (Upvotes) ותגובות (Comments) כדי למדוד פופולריות/מעורבות.

שימוש בתאריכים ושעות על מנת לקבל תוצאות רלוונטיות ליום הנוכחי.

**3.**

בנו זחלן המחזיר את התוצאות לשאילתות שהגדרתם. מומלץ (אך לא חובה) לבנות את הזחלן בפייתון. הציגו את קוד הזחלן.

.env השומר את הTOKEN הספציפי שקיבלנו בשביל גישה לAPI שמסופק על ידי REDDIT עצמם

REDDIT\_CLIENT\_ID="nmcA32g8dnE\_Dc3cBfr38Q"

REDDIT\_CLIENT\_SECRET="ECvq3EUxRiJ7P3F587RLoCmdkCwncw"

REDDIT\_USER\_AGENT="my\_reddit\_search/1.0 by u/Precise\_Pioneer"

:קוד מנוע החיפוש עצמו

Reddit Pioneer Searcher- github

import praw

import os

import re

import pandas as pd

from collections import Counter

from sklearn.feature\_extraction.text import TfidfVectorizer

from dotenv import load\_dotenv

# Load environment variables

load\_dotenv()

# Initialize Reddit API

reddit = praw.Reddit(

client\_id=os.getenv("REDDIT\_CLIENT\_ID"),

client\_secret=os.getenv("REDDIT\_CLIENT\_SECRET"),

user\_agent=os.getenv("REDDIT\_USER\_AGENT"),

)

print("Reddit API initialized. Read-only mode:", reddit.read\_only)

# Define stopwords

STOPWORDS = set([

"and", "or", "the", "is", "in", "to", "a", "of", "on", "for", "with", "it", "as", "at", "this", "that",

"an", "be", "are", "by", "was", "were", "from", "has", "have", "had", "but", "not", "you", "we", "they", "he", "she", "i", "me", "my"

])

# Function to clean text (remove emojis, stopwords, special characters)

def clean\_text(text):

text = re.sub(r"[^\w\s]", "", text)

text = text.lower()

words = text.split()

filtered\_words = [word for word in words if word not in STOPWORDS]

return " ".join(filtered\_words)

# Function to search Reddit and save results to Excel

def search\_reddit\_to\_excel(subreddit, query, limit, output\_file="reddit\_results.xlsx"):

try:

print(f"Searching Reddit for '{query}' in r/{subreddit}...")

results = reddit.subreddit(subreddit).search(

query=query,

limit=limit,

sort="hot",

time\_filter="month"

)

data = []

for post in results:

data.append({

"Title": post.title,

"Body": post.selftext,

"Reddit Post URL": f"https://reddit.com{post.permalink}",

"Score": post.score,

"Comments": post.num\_comments,

"Subreddit": post.subreddit.display\_name,

})

if data:

df = pd.DataFrame(data)

df.to\_excel(output\_file, index=False)

print(f"Results saved to {output\_file}")

else:

print(f"No results found for '{query}' in r/{subreddit}")

except Exception as e:

print(f"An error occurred: {e}")

raise

# Function to calculate TF-IDF

def calculate\_tfidf(input\_file="reddit\_results.xlsx", query="Funny cats"):

try:

df = pd.read\_excel(input\_file)

df["Content"] = df["Title"].astype(str) + " " + df["Body"].astype(str)

cleaned\_content = df["Content"].apply(clean\_text)

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform(cleaned\_content)

terms = vectorizer.get\_feature\_names\_out()

query\_terms = clean\_text(query).split()

print("\nTF-IDF Scores for Query Terms:")

for term in query\_terms:

if term in terms:

term\_index = vectorizer.vocabulary\_.get(term)

tfidf\_score = tfidf\_matrix[:, term\_index].toarray().sum()

print(f"{term}: {tfidf\_score}")

else:

print(f"{term}: Not found in the results.")

return vectorizer, tfidf\_matrix

except Exception as e:

print(f"An error occurred while calculating TF-IDF: {e}")

raise

# Function to calculate PageRank

def calculate\_pagerank(input\_file="reddit\_results.xlsx", output\_file="ranked\_results.xlsx"):

try:

df = pd.read\_excel(input\_file)

df["Relevance"] = df["Score"] # Assuming relevance is based on the score

# Normalize engagement scores

total\_upvotes = df["Score"].sum() or 1

total\_comments = df["Comments"].sum() or 1

df["Engagement Score"] = (

0.7 \* df["Score"] / total\_upvotes + 0.3 \* df["Comments"] / total\_comments

)

# Combine relevance and engagement

df["PageRank"] = 0.7 \* df["Relevance"] + 0.3 \* df["Engagement Score"]

# Rank posts

ranked\_df = df.sort\_values(by="PageRank", ascending=False)

# Save ranked results

ranked\_df.to\_excel(output\_file, index=False)

print(f"Ranked results saved to '{output\_file}'")

except Exception as e:

print(f"An error occurred while calculating PageRank: {e}")

raise

# Main script

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

try:

search\_reddit\_to\_excel("all", "Funny cats", limit=1000)

except Exception as e:

print(f"Error during Reddit search: {e}")

try:

calculate\_tfidf(query="Funny cats")

except Exception as e:

print(f"Error during TF-IDF calculation: {e}")

try:

calculate\_pagerank()

except Exception as e:

print(f"Error during PageRank calculation: {e}")

**4. רשמו טכנולוגיות שונות מעניינות שהשתמשתם בהם בפרויקט**

PRAW (Python Reddit API Wrapper)

ספרייה זו מאפשרת גישה ל-Reddit API, השתמשנו בה כדי לשלוף פוסטים מ-Reddit על פי מילות מפתח ותנאים מוגדרים מראש.

dotenv

כלי זה עוזר לטעון משתני סביבה מקובץ .env. הוא שימושי כדי להגן על המידע הרגיש כמו מזהים וסיסמאות של ה-API ולאפשר שינוי קל של ההגדרות הסביבתיות.

pandas

ספרייה זו משמשת לניתוח וניהול נתונים ביעילות. בקוד שלנו, השתמשנו ב-pandas כדי ליצור DataFrame של התוצאות ולבצע חישובים ודירוגים שונים על פי נתוני המעורבות של הפוסטים.

NLTK (Natural Language Toolkit)

השתמשנו ב-PorterStemmer מתוך ה-NLTK כדי לבצע סטמינג למילים בטקסטים.

re (Regular Expressions)

ספרייה זו מאפשרת עבודה עם ביטויים רגולריים כדי לבצע ניקוי של הטקסטים על ידי הסרת תווים מיוחדים והמרת הטקסט לאותיות קטנות.

**5.**

כמה זמן רצו השאילתות שלכם?

כל עוד מבקשים כמות מוגבלת של קבצים (לדוגמא: 20) זמן ריצת השאילתה נע בין 0.50 ל-0.8 שניות, תלוי בעומס על ה-API באותו רגע ומורכבות השאילתה ששלחנו.

נוסף על כך מתבצע חישוב TFIDF שתלוי בזמן עיבוד המעבד של המחשב ולוקח בין 0.1 ל-0.2. שימוש במטמון יכול לשפר זמן זה במידה ומשתמשים בשאילתות אשר חוזרות על עצמן.

כאשר מבקשים כמויות גדולות ובנוסף עליהם מבצעים חישוב ודירוג פוסטים על פי פופולריות ומעורבות, זמן הריצה עלול להיות גבוה יותר ולהגיע לכ-3 שניות.

**6.**

בנו inverted index ל- 15 המילים הנפוצות שחזרו (מספיק להתיחס ל -20 הדפים הראשונים שחזרו).

Query:The hottest posts about cats from the past month

INVERTED INDEX:

word posts

like 2

video 5

just 18

Amazing 1

cat 4,6,7,8,9,10,12,13,14,15,17,18,19,20

pic 19

clips 19

weird 3

all 19

about 19

petting 5

funny 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

your 3,19

up 18

out 20

**7. בחרו את אחת השאילתות, וחשבו tfIdf של המושגים בשאילתא**

חישוב TF-IDF

TF-TABLE:

Terms D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20

like 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

so 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

just 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

her 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

cat 0 0 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1

im 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

if 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

its 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

all 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

about 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

when 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

funny 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

your 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

up 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

out 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

TF-IDF TABLE

**8. האם בדפים שהוחזרו קיימים hubs? Authorities?**

במידע שהתקבל מהזחלן מופיעים 'Authorities' – פוסטים המקושרים אל 'Hubs' – אך בפועל אין לנו דפי 'Hubs' בתוצאות. למעשה, ה-Hubs ברדיט הם תתי-הפורומים (Subreddits), הכוללים מגוון נושאים שבהם משתמשים משתפים פוסטים. מכיוון שהשאילתות שלנו מכוונות ישירות לפוסטים הרלוונטיים, איננו נתקלים ישירות בדפי ה-Hub עצמם.

**9. בחרו 10 דפים שהזחלן החזיר, ואשר יש ביניהם קישורים .חשבו pageRank לכל דף**

בחרו 10 דפים שהזחלן החזיר ואשר יש ביניהם קישורים. חשבו PageRank לכל דף, מומלץ להעזר בסקריפט מוכן. הציגו את החישובים ואת הדירוג הסופי.

חישוב TF-IDF

**1. נחשב את כמות הפעמים שכל מילה מופיעה בפוסט מסוים (TF).**

**2. נחשב עד כמה מילה נפוצה בין כל הפוסטים (IDF). מילים נפוצות פחות יקבלו משקל גבוה יותר.**

TF-IDF

= TF × IDF

חישוב דמיון קוסינוס:

מחשבים את מידת הדמיון בין השאילתה לבין כל פוסט באמצעות נוסחת דמיון קוסינוס.

שימוש ב-לייקים ותגובות כמדד פופולריות של הפוסט: ניתן משקלים לפעולות שמשתמשים עושים על הפוסט (0.7 ללייק ו-0.3 לתגובה) ואז משלבים זאת בדירוג הסופי.

חישוב ה-PAGERANK הסופי:

משקל של 0.7 רלוונטיות ו-0.3 חשיפה.

**10. Feedback Relevance**

הראו לשני משתמשים שונים את הדירוג

מהסעיף הקודם, בקשו מהם לסמן relevance ובצעו feedback relevance. האם ניתן להציע שאילתא מותאמת על מנת לשפר תוצאות?

הצגנו את תוצאות החיפוש לשני משתמשים פעילים ברדיט. מתוך התוצאות הצגנו 10 פוסטים שעלו מהחיפוש, ושאלנו אותם עד כמה הם רלוונטיים עבורם. להלן רשימת הפוסטים אשר הצגנו להם:

מספר כותרת הפוסט

1 Show me your cat’s silliest facial expression to add to this doodle😋

2 Funny cat vs dog

3 A very funny cat is being spoiled

4 Something for the Colonel to consider 😸

5 My cat is very funny and photogenic, and I would be happy if someone would draw him

6 She looks so smol, I love her ❤️

7 Kind of funny, but why do cats watch you when you sleep? When I wake I often open my eyes to the shadow version of this

8 That one Sheba cat food ad with the neglectful mom.

9 Getting a New Kitten! Help Me Pick a Name for the Little one!

10 Thought this screenshot from the Dagi-Rahtborn playthrough would make for a funny meme. Poor Jo'sijar (aka Captain Cat Sparrow)... (סיפור חתולי מצחיק)

משתמש ראשון

פרופיל העדפות: אוהב תוכן צחוקים, בעל חיבה לחתולים ובעל פן הומוריסטי בולט.

פוסטים רלוונטיים עבורו:

(1) "Show me your cat’s silliest facial expression…" – מתבקש להציג הבעות פנים מצחיקות

(2) "Funny cat vs dog" – מעלה תכנים משעשעים של חתולים וכלבים

(3) "A very funny cat is being spoiled" – חתול מצחיק שמפנקים אותו

(5) "My cat is very funny and photogenic…" – שוב, דגש על הומור וחתול פוטוגני

(6) "She looks so smol, I love her ❤️" – פוסט חמוד במיוחד

(7) "Kind of funny, but why do cats watch you when you sleep?" – הומור על התנהגות משונה של חתולים

(10) "Thought this screenshot… funny meme…" – פוטנציאל למם מצחיק

פוסטים פחות רלוונטיים עבורו:

(4) "Something for the Colonel to consider…" – פחות ברור ומצחיק בעיניו

(8) "That one Sheba cat food ad…" – פחות עוסק בחומרים מצחיקים ויותר בפרסומת/סיפור ספציפי

(9) "Getting a New Kitten! Help Me Pick a Name…" – לא מחפש כרגע שמות לגורים

בסה״כ, המשתמש הראשון סימן 7 פוסטים כרלוונטיים (1,2,3,5,6,7,10).

חישוב Precision ו-Recall עבור משתמש ראשון

משתמש שני

פרופיל העדפות: אדם רציני מתעניין בגידול חתולים, אוהב חתולים משתעשעים. מחפש תוכן שעוזר ללמוד על חתולים.

פוסטים רלוונטיים עבורו:

(1) "Show me your cat’s silliest facial expression…" – מתבקש להציג הבעות פנים מצחיקות

(3) "A very funny cat is being spoiled" – חתול מצחיק שמפנקים אותו

(5) "My cat is very funny and photogenic…" – מעוניין בתמונות ובמידע על החתול, אולי גם השראה לציור

(7) "Kind of funny, but why do cats watch you when you sleep?" – דיון על התנהגות חתולים

(8) "That one Sheba cat food ad…" – מתעניין במזונות חתולים או פרסומות שקשורות לטיפול בהם

(9) "Getting a New Kitten! Help Me Pick a Name for the Little one!" – רלוונטי מאוד לאימוץ וגידול גורים

(6) "She looks so smol, I love her ❤️" – עוסק בגורים קטנים וחמודים (מתאים לאינטרס המשתמש לגבי גורים)

(10) "Thought this screenshot… funny meme…" – למרות שהפוסט מצחיק, יש בו גם סיפור על החתול שמושך את תשומת ליבו

בסה"כ, המשתמש השני סימן 8 פוסטים כרלוונטיים (1,3,5,6,7,8,9,10).

חישוב Precision ו-Recall עבור משתמש שני

הצעות לשאילתות מותאמות

עבור המשתמש הראשון

"Funny cat memes OR silly cat expressions OR cute cat videos"

מיקוד בתכנים משעשעים, מצחיקים וחמודים.

עבור המשתמש השני

"Cat care tips OR adopting a kitten OR how to name a new cat"

מיקוד במידע מעשי לאימוץ וגידול חתולים.

בדוגמה זו ניתן לראות כיצד שני המשתמשים בוחרים פוסטים שונים על סמך העדפות שונות (אחד מחפש בעיקר תוכן מצחיק וקליל, והשני מחפש ידע וטיפים הקשורים לטיפול וגדילה של חתולים).

החישוב של Precision ו-Recall מציג עד כמה התוצאות הרלוונטיות מתוך כלל התוצאות שהוצגו (Precision) ועד כמה הן משקפות את כלל התוכן הזמין בסאברדיט (Recall).

© Precision Pioneers, Data Retrieval Project 2025