

AI Tools for SEO: From LLM Workflows to MCP

As an SEO, I've always
looked up to **developers.**
They're usually the ones
who can turn my vision into
reality.

While engineers and developers are still essential, “**vibe coding**” is a shortcut to build small, functional applications

You'll be able to create your
own functional app by the
end of the day. **TODAY.**

LLM
(API Access)



Connector
(Optional)



**Google Colab or
Sheets**



TL;DR: Ask ChatGPT to write you the code. Like magic ✨

📄 📄 ChatGPT 4o with canvas ▾

I need a Google Colab code to upload URL embeddings and find the five closest related pages. Please exclude the initial URL from the matching list.



Sure! Here is a Google Colab code that allows you to upload URL embeddings, find the five closest related pages, and exclude the initial URL from the results. This example uses cosine similarity to determine closeness:

🔗 Url Embeddings Search

This code allows you to upload a CSV file containing the URL and its embeddings. It then computes the cosine similarity between the provided URL and all others, excluding the initial one, and finally returns the five closest URLs.

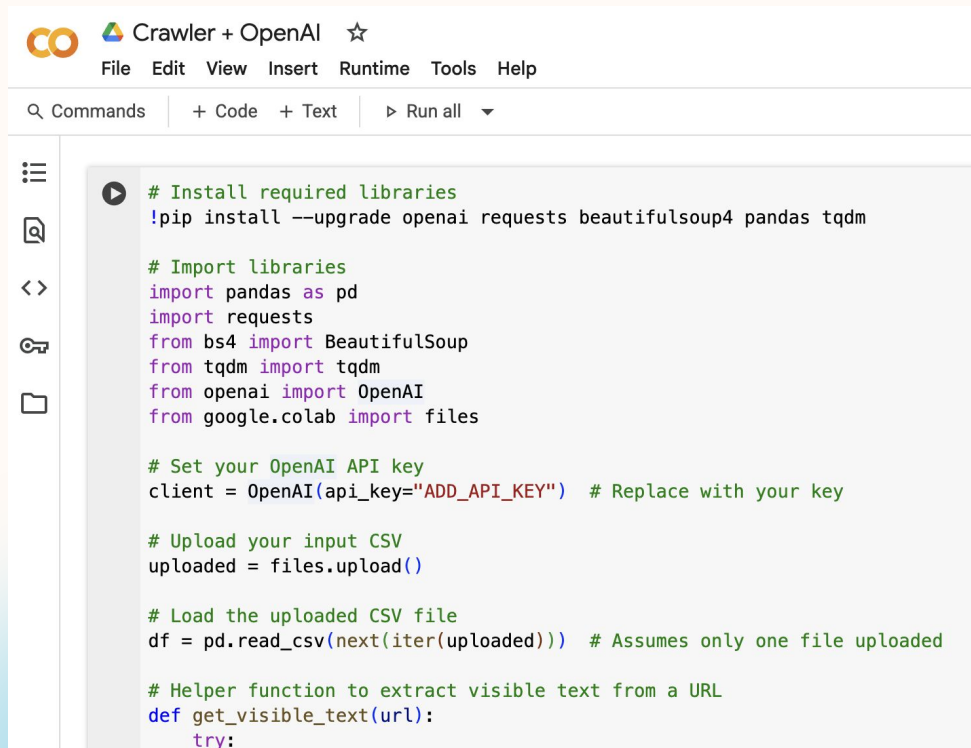
CSV File Format:

- The CSV should have the URLs in the

✕ Url Embeddings Search

```
15 # Extract the embeddings matrix
16 embeddings = df.iloc[:, 1:].values
17
18 # Calculate cosine similarity between all embeddings
19 similarities = cosine_similarity(embeddings)
20
21 # Set similarity of the initial index with itself to -1 (to exclude it)
22 similarities[initial_index, initial_index] = -1
23
24 # Find the five closest embeddings
25 closest_indices = np.argsort(similarities[initial_index])[::-1][5:]
26
27 # Get the corresponding URLs
28 closest_urls = df.iloc[closest_indices, 0]
29 return closest_urls
30
31 # Main code
32 # Step 1: Upload the embeddings file (must be a CSV with URL as the first column)
33 print("Please upload the URL embeddings CSV file (URLs and their vector embeddings)...")
34 df = upload_embeddings()
35
36 # Step 2: Input the initial URL to find related pages
37 initial_url = input("Enter the URL for which you want to find related pages: ")
38 initial_index = df[df.iloc[:, 0] == initial_url].index
39
40 if len(initial_index) == 0:
41     print("The provided URL is not in the uploaded embeddings.")
42 else:
43     # Step 3: Find and display the five closest related pages
44     initial_index = initial_index[0]
45     closest_pages = find_closest_embeddings(df, initial_index)
46     print("\nThe five closest related pages are:")
47     for url in closest_pages:
48         print(url)
49
```

Google Colab (Python)



The screenshot displays the Google Colab interface. At the top, the title bar reads "Crawler + OpenAI" with a star icon. Below it is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A toolbar shows "Commands", "+ Code", "+ Text", and "Run all". The left sidebar contains icons for a menu, search, code editor, runtime, and files. The main area shows a code cell with the following Python code:

```
# Install required libraries
!pip install --upgrade openai requests BeautifulSoup4 pandas tqdm

# Import libraries
import pandas as pd
import requests
from bs4 import BeautifulSoup
from tqdm import tqdm
from openai import OpenAI
from google.colab import files

# Set your OpenAI API key
client = OpenAI(api_key="ADD_API_KEY") # Replace with your key

# Upload your input CSV
uploaded = files.upload()

# Load the uploaded CSV file
df = pd.read_csv(next(iter(uploaded))) # Assumes only one file uploaded

# Helper function to extract visible text from a URL
def get_visible_text(url):
    try:
```

Google Sheets App Scripts (JavaScript)

The screenshot displays the Google Sheets App Script editor interface. On the left, the 'Extensions' menu is open, showing 'Add-ons', 'Macros', 'Apps Script', 'AppSheet', and 'Looker Studio'. The main editor window is titled 'Edge Graph Score Tracker' and indicates 'Unsaved changes'. The script editor shows a function named `fetchDailyQueries()` with the following code:

```
1 function fetchDailyQueries() {  
2   const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
3   const lastRow = sheet.getLastRow(); // Find the last row with data  
4  
5   // Iterate through all rows starting from Row 2 (skipping headers)  
6   for (let row = 2; row <= lastRow; row++) {  
7     const query = sheet.getRange(row, 10).getValue(); // Column J (Query)  
8     if (query) {  
9       const apiKey = 'ADD_API_KEY_HERE'; // Replace with your actual API key.  
10      const serviceUrl = 'https://kgsearch.googleapis.com/v1/entities:search'  
11  
12      const params = {  
13        query: query,  
14        limit: 3, // Fetch up to 3 results  
15        indent: true,  
16        key: apiKey,  
17      };  
18    }
```

Copy, Paste and Run

🔍 Commands

+ Code

+ Text

▶ Run all



```
from openai import OpenAI
from google.colab import files

# Set your OpenAI API key
client = OpenAI(api_key="ADD_API_KEY") # Replace with your key
```



▶ Run

⌘ Debug

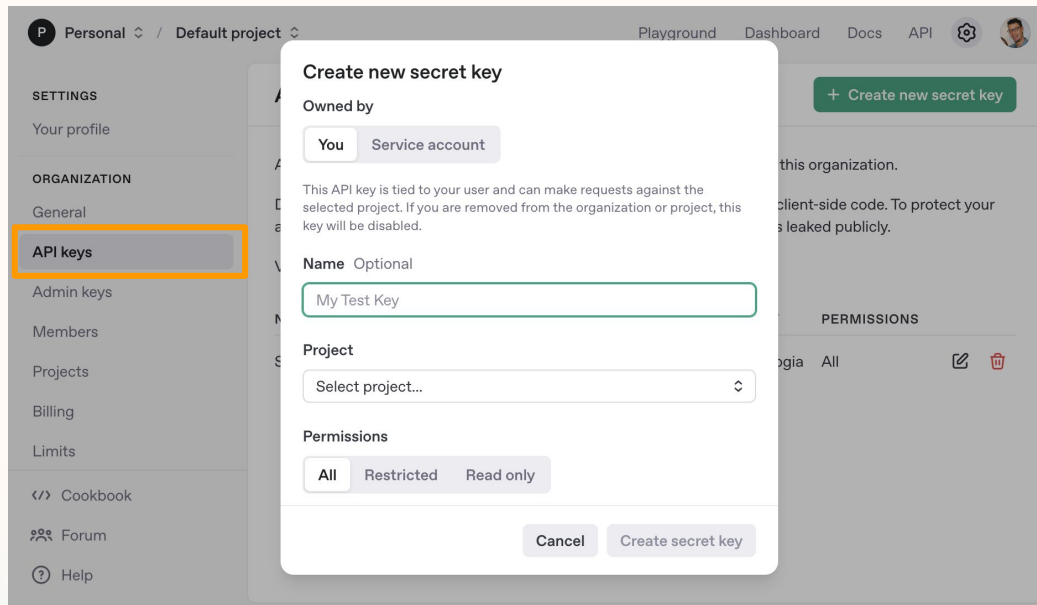
fetchDailyQueries ▼

Execu

```
7  const query = sheet.getRange(row, 10).getValue(); // Column J (Query)
8  if (query) {
9      const apiKey = 'ADD_API_KEY_HERE'; // Replace with your actual API key.
10     const serviceUri = 'https://kgsearch.googleapis.com/v1/entities:search'
11
```

Create your OpenAI API Key

- openai.com/api/
 - Pay as you go billing
 - \$20 will take you far
-
- ai.google.dev/gemini-api/docs/api-key
 - Free tier
 - Data is used for training



Start with a cheap OpenAI model

Model	Input \$/1M toks	Output \$/1M toks	Cost per article (2k tokens)	Total for 1,000
GPT-4.1	\$2.00	\$8.00	$(2\text{ k} / 1\text{ M}) \times (\$2 + \$8) = \0.02	\$20
GPT-4.1 mini	\$0.40	\$1.60	$(2\text{ k} / 1\text{ M}) \times (\$0.4 + \$1.6) = \0.004	\$4
GPT-4o-mini	\$0.15*	\$0.60*	$(2\text{ k} / 1\text{ M}) \times (\$0.15 + \$0.60) = \0.0015	\$1.50

What to ask

How to ask

Refine your ask

01

Traffic & Revenue Overlap

```
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import files
```

```
# Step 1: Upload CSV file
uploaded = files.upload()
```

```
# Step 2: Load the CSV file into a DataFrame
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)
```

```
# Step 3: Ensure the CSV has the necessary columns (landing_page_url, traffic, revenue)
if 'traffic' not in df.columns or 'revenue' not in df.columns:
    print("Error: CSV must contain 'traffic' and 'revenue' columns.")
else:
```

You don't have the resources to keep optimizing in all pages

Only a percentage of pages represent most of your traffic AND lead generation/revenue

```
# Step 4: Sort the DataFrame by traffic and revenue
df_sorted_by_traffic = df.sort_values(by='traffic', ascending=False)
df_sorted_by_revenue = df.sort_values(by='revenue', ascending=False)
```

```
# Step 5: Calculate cumulative traffic and revenue
df_sorted_by_traffic['cumulative_traffic'] = df_sorted_by_traffic['traffic'].cumsum() / df_sorted_by_traffic['traffic'].sum() * 100
df_sorted_by_revenue['cumulative_revenue'] = df_sorted_by_revenue['revenue'].cumsum() / df_sorted_by_revenue['revenue'].sum() * 100
```

```
# Step 6: Create text output for traffic
print("\nCumulative traffic")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_traffic[df_sorted_by_traffic['cumulative_traffic'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of traffic")
```

This allows you to find the "sweet spot" between both groups

```
# Step 7: Create text output for revenue
print("\nCumulative revenue")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_revenue[df_sorted_by_revenue['cumulative_revenue'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of revenue")
```

Ask ChatGPT to write your code

I need a Python script for Google Colab that can:

Calculate and display the cumulative percentages for both traffic and revenue from GA4, showing how many pages represent 20%, 40%, 60%, 80%, and 100% of the total.

Plot two graphs: one for cumulative traffic and one for cumulative revenue distribution across the pages. Find the URLs that represent 50% of the traffic and revenue.

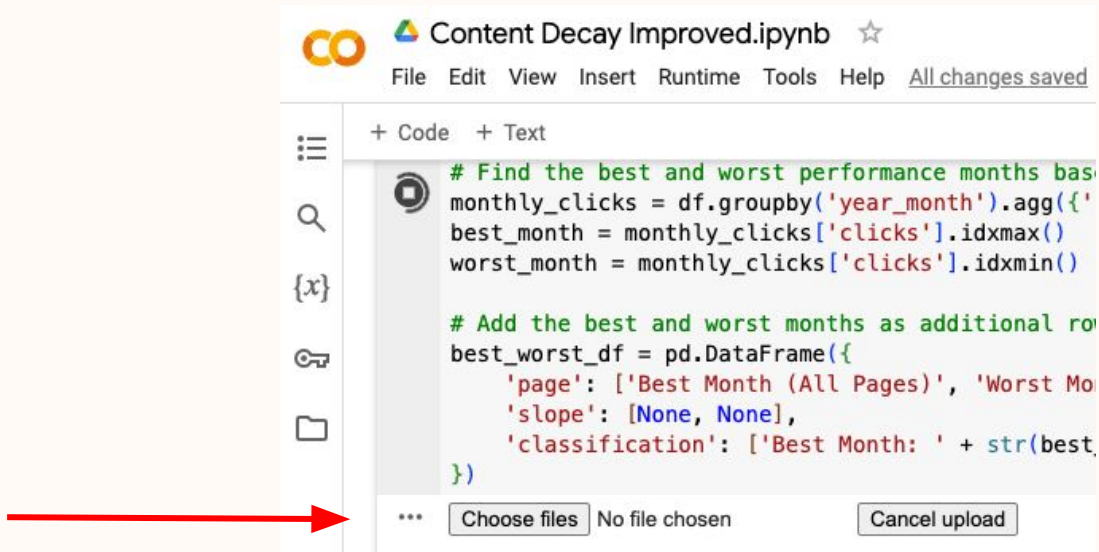
The input will be given by a CSV file I'll upload. Combine the URLs into one file and automatically export this data to a new CSV file.

✕ Traffic Revenue Analysis



```
62     'URL'].tolist()
63     revenue_50_percent_urls = revenue_sorted.loc[:revenue_50_percent_index,
64     'URL'].tolist()
65
66     # Find overlapping URLs between traffic and revenue 50%
67     overlapping_urls = list(set(traffic_50_percent_urls) & set(revenue_50_percent_urls))
68     print(f"Number of URLs representing ~50% of both Traffic and Revenue:
69     {len(overlapping_urls)}")
70
71     # Step 7: Export combined data to a new CSV file
72     data_combined = pd.merge(traffic_sorted[['URL', 'Traffic', 'Cumulative Traffic %']],
73     revenue_sorted[['URL', 'Revenue', 'Cumulative Revenue %']],
74     on='URL',
75     how='outer').fillna(0)
76
77     output_filename = 'combined_traffic_revenue_data.csv'
78     data_combined.to_csv(output_filename, index=False)
79     print(f"Combined data export ●")
```

Upload your export from analytics: date, page and traffic columns



**Fewer than 3k
pages are
responsible for
80% of traffic
and revenue**

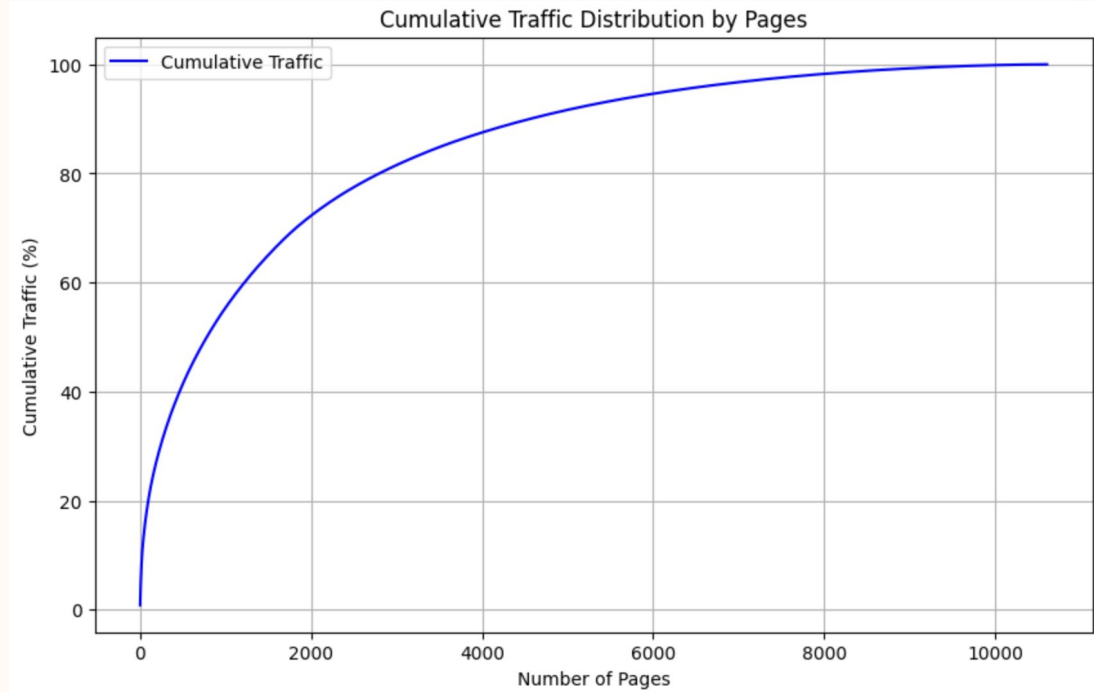


Cumulative traffic distribution:
First 95 pages = 20% of traffic
First 462 pages = 40% of traffic
First 1221 pages = 60% of traffic
First 2797 pages = 80% of traffic
First 10607 pages = 100% of traffic



Cumulative revenue distribution:
First 65 pages = 20% of revenue
First 327 pages = 40% of revenue
First 980 pages = 60% of revenue
First 2611 pages = 80% of revenue
First 10607 pages = 100% of revenue

Auto-generated visuals to help you tell a story



02

Crawler with AI Prompts

(Google Sheets)

Disclaimer: You can already do this in better ways with existing SEO tools. This is just an example to illustrate AI's coding capabilities.

```
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import files
```

```
# Step 1: Upload CSV file
uploaded = files.upload()
```

```
# Step 2: Load the CSV file into a pandas DataFrame
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)
```

```
# Step 3: Ensure the CSV has
if 'traffic' not in df.columns:
    print("Error: CSV must contain 'traffic' column")
else:
```

Crawl URLs just by adding them on a spreadsheet

```
# Step 4: Sort the DataFrame by traffic and revenue
df_sorted_by_traffic = df.sort_values(by='traffic', ascending=False).reset_index(drop=True)
df_sorted_by_revenue = df.sort_values(by='revenue', ascending=False).reset_index(drop=True)
```

```
# Step 5: Calculate cumulative traffic and revenue
df_sorted_by_traffic['cumulative_traffic'] = df_sorted_by_traffic['traffic'].cumsum()
df_sorted_by_revenue['cumulative_revenue'] = df_sorted_by_revenue['revenue'].cumsum()
```

```
# Step 6: Create text output for traffic distribution
print("\nCumulative traffic distribution:")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_traffic[df_sorted_by_traffic['cumulative_traffic'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of traffic")
```

```
# Step 7: Create text output for revenue distribution
print("\nCumulative revenue distribution:")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_revenue[df_sorted_by_revenue['cumulative_revenue'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of revenue")
```

```
df_sorted_by_traffic['traffic'].sum() * 100
df_sorted_by_revenue['revenue'].sum() * 100
```

Run prompts automatically

EXAMPLE: How to ask ChatGPT to write your code

Please write me a Python code to use with Google Colab.

I'll provide you with a URL, which you need to crawl / scrape the content. The URLs are in column A.

Then, run this by the prompt in Column B. This will make a call OpenAI (I'll provide an API key) and return the answer in a CSV.

If your code fails, ask a few more questions

What should I name my columns?

Name your columns exactly like this in the CSV you upload to Colab:

Copy Edit

url, prompt

Each row should follow this structure:

url	prompt
https://example.com/page1	Summarize the key points of this page.
https://example.com/page2	What is the main argument here?

Keep all URLs in the `url` column and your custom instructions in the `prompt` column. The script will match them row by row and return the AI's response in a new CSV.

I got this error

"OpenAI error:

You tried to access `openai.ChatCompletion`, but this is no longer supported in `openai>=1.0.0` - see the README at <https://github.com/openai/openai-python> for the API.

You can run `openai migrate` to automatically upgrade your codebase to use the 1.0.0 interface.

Alternatively, you can pin your installation to the old version, e.g. `pip install openai==0.28`

A detailed migration guide is available here:

<https://github.com/openai/openai-python/discussions/742>"

Thanks for the heads up. You're using `openai` version 1.0 or newer, which uses a **new syntax**. Let's update the code for the current version.

✓ Updated Script (for `openai>=1.0.0`)

```
answer = call_openai(text, prompt)
results.append({
    "url": url,
    "prompt": prompt,
    "response": answer
})

# Save to a new CSV file
output_df = pd.DataFrame(results)
output_df.to_csv("output_with_responses.csv", index=False)

# Download the result
files.download("output_with_responses.csv")
```

```
Requirement already satisfied: openai in /usr/local/lib/python3.11/dist-packages (1.86.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (2.32.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (4.13.4)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.3.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (4.67.1)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from openai) (4.9.0)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from openai) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from openai) (0.28.1)
Requirement already satisfied: jiter<1,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from openai) (0.10.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from openai) (2.11.5)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-packages (from openai) (1.3.1)
Requirement already satisfied: typing-extensions<5,>=4.11 in /usr/local/lib/python3.11/dist-packages (from openai) (4.14.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests) (2025.4.26)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4) (2.7)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->openai) (1.0.9)
```

url	prompt	response
https://www.wix.com/seo/learn/resource/chatgpt-google-colab-seo-tools	Summarize this article in one phrase	The article explains how to create three SEO tools using ChatGPT and Google Colab, and discusses the benefits of using these platforms for coding and application creation.
https://www.wix.com/seo/learn/resource/what-is-serp-analysis	Summarize this article in two phrases	The article discusses how to run a search engine result page (SERP) analysis to improve chances of ranking for a particular keyword, detailing different SERP features and why it's essential to understand them. It also explains strategies to incorporate SERP analysis into a content plan, how to conduct large-scale SERP analyses and the importance of understanding user intent behind keywords.
https://www.wix.com/seo/learn/resource/create-an-seo-proposal	Summarize this article in three phrases	The article discusses the importance of trust in creating an effective SEO proposal. It stresses that explaining to clients or stakeholders about their own involvement in a project is crucial, and that without their participation, desired business outcomes cannot be achieved. The article also urges SEOs to prioritize tasks that can demonstrably make an impact, rather than trying to cover all bases.

03

Related Pages Finder

(Google Colab)

```
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import files
```

```
# Step 1: Upload CSV file
uploaded = files.upload()
```

```
# Step 2: Load the CSV file into a DataFrame
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)
```

```
# Step 3: Ensure the CSV has the necessary columns (landing_page_url, traffic, revenue)
if 'traffic' not in df.columns or 'revenue' not in df.columns:
    print("Error: CSV must contain 'traffic' and 'revenue' columns.")
else:
```

```
# Step 4: Sort the DataFrame by traffic and revenue
df_sorted_by_traffic = df.sort_values('traffic')
df_sorted_by_revenue = df.sort_values('revenue')
```

```
# Step 5: Calculate cumulative traffic and revenue
df_sorted_by_traffic['cumulative_traffic'] = df_sorted_by_traffic['traffic'].cumsum()
df_sorted_by_revenue['cumulative_revenue'] = df_sorted_by_revenue['revenue'].cumsum() / df_sorted_by_revenue['revenue'].sum() * 100
```

```
# Step 6: Create text output for traffic
print("\nCumulative traffic")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_traffic[df_sorted_by_traffic['cumulative_traffic'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of traffic")
```

```
# Step 7: Create text output for revenue
print("\nCumulative revenue")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_revenue[df_sorted_by_revenue['cumulative_revenue'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of revenue")
```

Internal link opportunities at scale

Tag pages

Content clustering

Vector Embeddings + Cosine Similarity

A vector embedding is a numerical representation of text that captures their semantic meaning and relationships

Embeddings for <https://www.guspelogia.com/write-seo-dev-tickets>

[illegible]

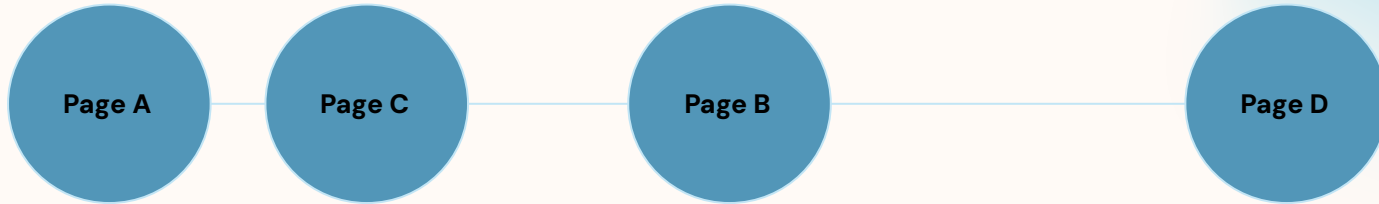
We're looking at every word and relationship between them, not just keywords, titles, etc

A hand-drawn diagram illustrating word embeddings. On the left, a vertical column of boxes represents dimensions: 'Royalty', 'Masculinity', 'Femininity', 'Age', and an ellipsis. To the right, four vertical columns represent words: 'King', 'Queen', 'Woman', and 'Princess', each with corresponding numerical values in its boxes.

	King	Queen	Woman	Princess
Royalty	0.99	0.99	0.02	0.98
Masculinity	0.99	0.05	0.01	0.02
Femininity	0.05	0.93	0.999	0.94
Age	0.7	0.6	0.5	0.1
...	...	...	...	...

Image source: medium.com/artificial-coder

“Cosine similarity discovers how close vector embeddings from one page is from another. Crunching the vectors!”



Use Screaming Frog with API access to an LLM to extract page embeddings for you



System User	
Search Presets	
Name	Comments
Detect inappropriate content	Detect harmful, inappropriate, or unsafe content across digita...
Extract embeddings from page content	Extract embeddings from page content
Generate alt text for images	Use OpenAI to generate alt text for all image URLs crawled...
Intent of Page	Intent of Page
Language of Page	Language of the body text of the page
Sentiment of page	One word page content sentiment using OpenAI

EXAMPLE: How to ask ChatGPT to write your code

Write Python (Google Colab) code that reads a CSV file containing URLs and their embeddings, then calculates cosine similarity to find the top 5 related URLs for each page.

The CSV file should have headers "URL" and "Embeddings". Save the results as a new CSV with columns "URL" and "Related URLs". Download CSV automatically.

Write Python (Google Colab) code that reads a CSV file containing URLs and their embeddings, then calculates cosine similarity to find the top 5 related URLs for each page. The CSV file should have the headers "URL" and "Embeddings". Ensure that the embeddings are converted from strings to numpy arrays. Save the result as a new CSV with columns "URL" and "Related URLs".

< 2/2 >



Certainly! Below is the Python code that reads a CSV file containing URLs and their embeddings, computes cosine similarity for each URL, and finds the top 5 related URLs for each URL. It saves the result as a new CSV file with columns "URL" and "Related URLs".

Url Similarity Csv

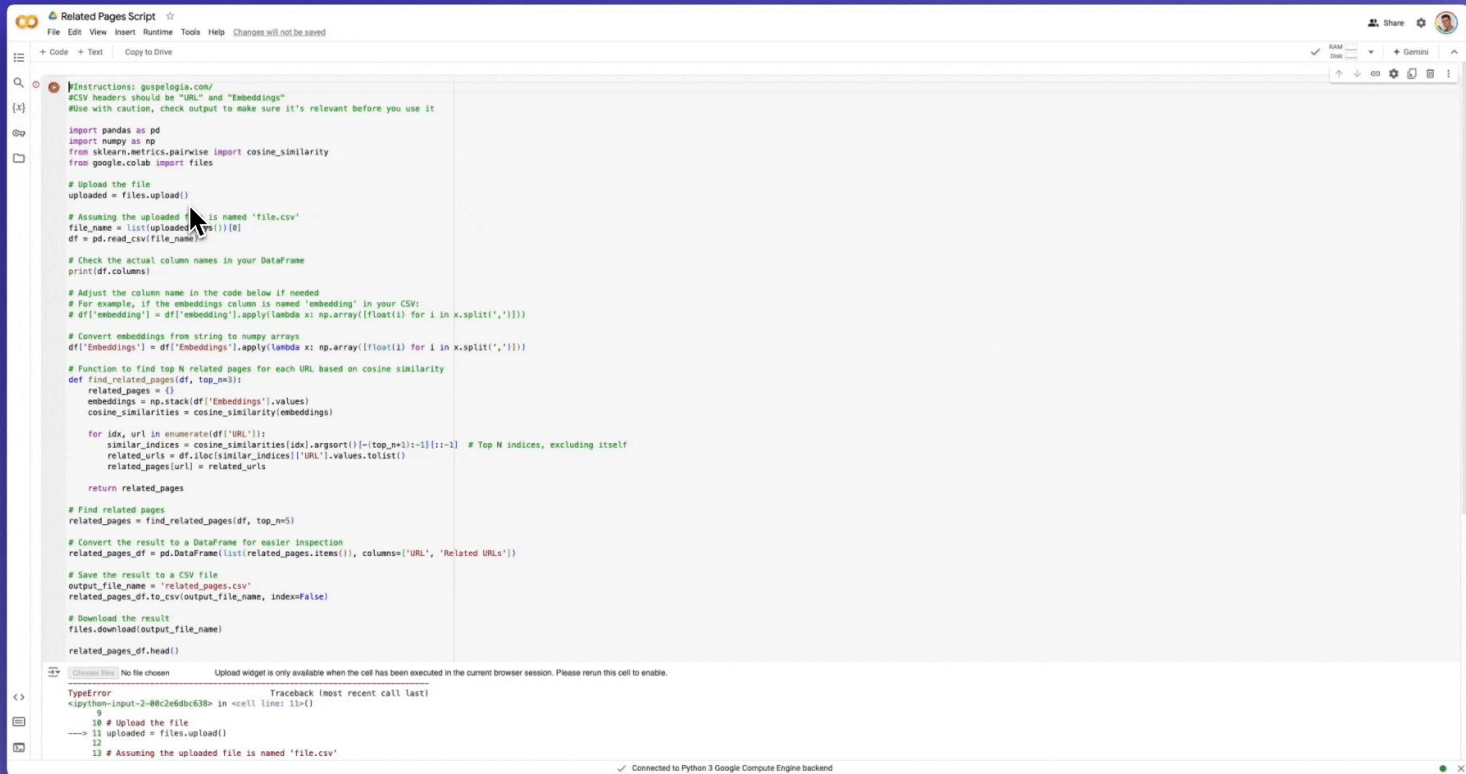
This code reads the input CSV, converts embeddings from strings to numpy arrays, computes cosine similarity, and identifies the top 5 related URLs for each URL. The resulting CSV will have the "URL" and "Related URLs" columns.

Make sure to adjust the `input_file` variable to the name of your CSV file, and ensure your embeddings are stored in a format that can be evaluated correctly (as lists of numbers in string form).

Url Similarity Csv

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.metrics.pairwise import cosine_similarity
4 import ast
5
6 # Load the CSV file
7 input_file = "urls_with_embeddings.csv" # Replace with your file name
8 output_file = "related_urls.csv"
9
10 # Read the CSV into a DataFrame
11 df = pd.read_csv(input_file)
12
13 # Convert embeddings from string to numpy array
14 df["Embeddings"] = df["Embeddings"].apply(lambda x: np.array(ast.literal_eval(x)))
15
16 # Calculate cosine similarity between embeddings
17 embeddings_matrix = np.stack(df["Embeddings"].values)
18 cosine_sim_matrix = cosine_similarity(embeddings_matrix)
19
20 # Find the top 5 related URLs for each URL
21 related_urls = []
22 for idx, row in df.iterrows():
23     # Get similarity scores for the current URL, excluding itself
24     similarity_scores = cosine_sim_matrix[idx]
25     related_indices = np.argsort(similarity_scores)[-6:-1][::-1] # Get top 5 excluding itself
26     top_related_urls = df.iloc[related_indices]["URL"].tolist()
27     related_urls.append(top_related_urls)
28
29 # Add the related URLs to the DataFrame
30 df["Related URLs"] = ["", ".".join(urls) for urls in related_urls]
31
32 # Save the result as a new CSV file
33 df[["URL", "Related URLs"]].to_csv(output_file, index=False)
34
35 print(f"Related URLs saved to {output_file}")
36
```

Run Colab Notebook + Results



```
Instructions: guspelogia.com/
#CSV headers should be "URL" and "Embeddings"
#Use with caution, check output to make sure it's relevant before you use it

import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from google.colab import files

# Upload the file
uploaded = files.upload()

# Assuming the uploaded file is named 'file.csv'
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)

# Check the actual column names in your DataFrame
print(df.columns)

# Adjust the column name in the code below if needed
# For example, if the embeddings column is named 'embedding' in your CSV:
df['embedding'] = df['embedding'].apply(lambda x: np.array([float(i) for i in x.split(',')]))

# Convert embeddings from string to numpy arrays
df['Embeddings'] = df['Embeddings'].apply(lambda x: np.array([float(i) for i in x.split(',')]))

# Function to find top N related pages for each URL based on cosine similarity
def find_related_pages(df, top_n=3):
    related_pages = {}
    embeddings = np.stack(df['Embeddings'].values)
    cosine_similarities = cosine_similarity(embeddings)

    for idx, url in enumerate(df['URL']):
        similar_indices = cosine_similarities[idx].argsort()[-(top_n+1):][::-1] # Top N indices, excluding itself
        related_urls = df.iloc[similar_indices]['URL'].values.tolist()
        related_pages[url] = related_urls

    return related_pages

# Find related pages
related_pages = find_related_pages(df, top_n=5)

# Convert the result to a DataFrame for easier inspection
related_pages_df = pd.DataFrame(list(related_pages.items()), columns=['URL', 'Related URLs'])

# Save the result to a CSV file
output_file_name = 'related_pages.csv'
related_pages_df.to_csv(output_file_name, index=False)

# Download the result
files.download(output_file_name)

related_pages_df.head()
```

Choose file No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
TypeError                                Traceback (most recent call last)
<ipython-input-2-00c2ef6db638> in <cell line: 13>()
      9
----> 10 # Upload the file
      11 uploaded = files.upload()
      12
      13 # Assuming the uploaded file is named 'file.csv'
```

Connected to Python 3 Google Compute Engine backend

04

Knowledge Panel Tracker (Google Sheets)

```
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import files
```

```
# Step 1: Upload CSV file
uploaded = files.upload()
```

```
# Step 2: Load the CSV file into a pandas DataFrame
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)
```

```
# Step 3: Ensure the CSV has 'traffic' column
if 'traffic' not in df.columns:
    print("Error: CSV must contain 'traffic' column")
else:
```

Find entities on Google

```
# Step 4: Sort the DataFrame by traffic and revenue
df_sorted_by_traffic = df.sort_values(by='traffic', ascending=False).reset_index(drop=True)
df_sorted_by_revenue = df.sort_values(by='revenue', ascending=False).reset_index(drop=True)
```

```
# Step 5: Calculate cumulative traffic and revenue
df_sorted_by_traffic['cumulative_traffic'] = df_sorted_by_traffic['traffic'].cumsum()
df_sorted_by_revenue['cumulative_revenue'] = df_sorted_by_revenue['revenue'].cumsum()
```

Person, Organization, Thing

```
df_sorted_by_traffic['traffic'].sum() * 100
df_sorted_by_revenue['revenue'].sum() * 100
```

```
# Step 6: Create text output for traffic distribution
print("\nCumulative traffic distribution:")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_traffic[df_sorted_by_traffic['cumulative_traffic'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of traffic")
```

```
# Step 7: Create text output for revenue distribution
print("\nCumulative revenue distribution:")
for percentage in [20, 40, 60, 80, 100]:
    page_count = df_sorted_by_revenue[df_sorted_by_revenue['cumulative_revenue'] <= percentage].shape[0]
    print(f"First {page_count} pages = {percentage}% of revenue")
```

This is where you find if Google recognizes as entity

```
@id: https://www.google.com/search?kgmid=/g/11rscwh51j
name: Gus Pelogia
@type: ['Person', 'Thing']
description: N/A
url: N/A
resultScore: 348.1849975585938
```

Search results for "gus pelogia" showing a profile card for Gus Pelogia, an SEO professional.

Gus Pelogia
SEO professional

A man with glasses and a blue and red striped shirt is speaking at a conference. A "LinkedIn" logo is visible in the bottom left corner of the image.

Gus Pelogia is a **journalist turned SEO**, working in digital marketing since 2012. He's currently an SEO Product Manager at Indeed, the #1 job site in the world with over 350 million unique visitors every month.

<https://ie.linkedin.com › gpelogia>

Gus Pelogia - Indeed - LinkedIn Ireland

LinkedIn

Gus Pelogia on LinkedIn: Are you interested in learning how to run ...

Gus Pelogia's Post. View profile for Gus Pelogia, graphic. Gus Pelogia. Senior SEO Product...

A portrait of Gus Pelogia, a man with glasses and a blue shirt.

Gus Pelogia
Gus Pelogia | SEO

First, get your Google Cloud API

The API key is free to use, but you must create your own



Knowledge Graph Search API

[Google](#)

Searches the Google Knowledge Graph for entities.

ENABLE TRY THIS API [↗](#)

Click to enable this API

1 Credential Type

Which API are you using?

Different APIs use different auth platforms and some credentials can be restricted to only call certain APIs.

Select an API *
Knowledge Graph Search API

What data will you be accessing? *

Different credentials are required to authorise access depending on the type of data that you request. [Learn more](#)

☒ **Public data**

Google data that is publicly available, like public Maps data showing restaurant information. This will create an API key.

NEXT

2 Your credentials

DONE CANCEL

EXAMPLE: How to ask ChatGPT to write your code

Write a Python script that calls the Google Knowledge Graph API, searches for a specific query (e.g., 'Taylor Swift'), and prints the fields @id, name, @type, description, url, and resultScore.

If the query doesn't return a valid result, please print the message 'Query is not present on Knowledge Graph panel.'

This is meant to work on Google Colab.

Can you help me write a Python script that calls the Google Knowledge Graph API, searches for a specific query (e.g., 'Taylor Swift'), and prints the fields @id, name, @type, description, url, and resultScore? If the query doesn't return a valid result, please print the message 'Query is not present on Knowledge Graph panel.'

This is meant to work on Google Colab.

< 2/2 >



Sure! Here's a Python script for Google Colab that calls the Google Knowledge Graph API to search for a specific query and print the desired fields. Please make sure you have a valid API key for the Knowledge Graph API before running it.

Google Kg Api Search

Make sure to replace 'YOUR_GOOGLE_API_KEY' with your actual Google API key. You can also modify the query variable with any term you wish to search for. This script makes an HTTP request to the Google Knowledge Graph API, processes the response, and prints the fields you specified.

If the query does not return a valid result, it will print "Query is not present on Knowledge Graph panel."



Google Kg Api Search

```
1 import requests
2 import json
3
4 # Set your Google Knowledge Graph API key here
5 API_KEY = 'YOUR_GOOGLE_API_KEY'
6
7 def search_knowledge_graph(query):
8     url = "https://kgsearch.googleapis.com/v1/entities:search"
9     params = {
10         "query": query,
11         "key": API_KEY,
12         "limit": 1,
13         "indent": True,
14     }
15
16     response = requests.get(url, params=params)
17     if response.status_code == 200:
18         results = response.json()
19         if "itemListElement" in results and len(results["itemListElement"]) > 0:
20             element = results["itemListElement"][0]["result"]
21             print("@id:", element.get("@id", "N/A"))
22             print("name:", element.get("name", "N/A"))
23             print("@type:", element.get("@type", "N/A"))
24             print("description:", element.get("description", "N/A"))
25             print("url:", element.get("url", "N/A"))
26             print("resultScore:", results["itemListElement"][0].get("resultScore", "N/A"))
27         else:
28             print("Query is not present on Knowledge Graph panel.")
29     else:
30         print("Error in API request. Status code:", response.status_code)
31
32 # Set the query you want to search for
33 query = "Taylor Swift"
34
35 # Call the function to search
36 search_knowledge_graph(query)
37
```

Step 3: Run Notebook and collect results

Did you like the resources in this template? Get them for free at our other websites

```
Knowledge Graph API
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

# Create the URL and make the request
url = service_url + '?' + urllib.parse.urlencode(params)
response = urllib.request.urlopen(url)
data = json.loads(response.read())

# Check if there are valid results
if not data.get('itemListElement'):
    print("Entity is not present on Knowledge Graph panel")
else:
    # Print the requested fields from the response
    for element in data['itemListElement']:
        result = element.get('result')
        if result:
            kgmid = result.get('@id', 'N/A')
            if kgmid != 'N/A':
                # Remove "kg:" prefix if present
                if kgmid.startswith("kg:"):
                    kgmid = kgmid[3:]
                encoded_kgmid = urllib.parse.quote(kgmid)
                kgmid_url = f"https://www.google.com/search?kgmid={encoded_kgmid}"
            else:
                kgmid_url = 'N/A'

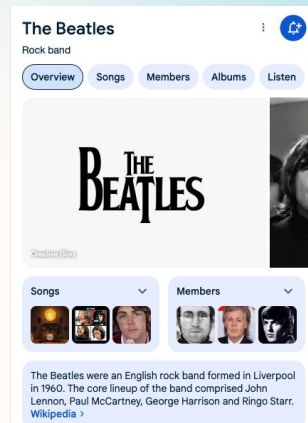
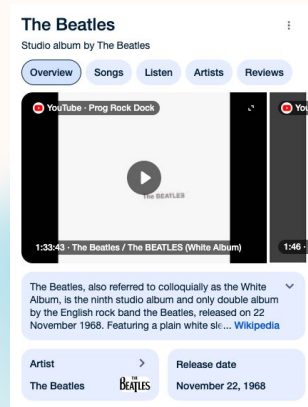
            print(f"@id: {kgmid_url}")
            print(f"name: {result.get('name', 'N/A')}")
            print(f"@type: {result.get('@type', 'N/A')}")
            print(f"description: {result.get('description', 'N/A')}")
            print(f"url: {result.get('url', 'N/A')}")
            print(f"resultScore: {element.get('resultScore', 'N/A')}")
            print(f'-' * 40) # Separator for readability
        else:
            print("Query is not present on Knowledge Graph panel")
            break
```

```
@id: https://www.google.com/search?kgmid=m/07c0j
name: The Beatles
@type: ['MusicGroup', 'Thing', 'Organization']
description: Rock band
url: http://thebeatles.com/
resultScore: 5541.28759765625

-----
@id: https://www.google.com/search?kgmid=m/0lhg4bh
name: The Beatles
@type: ['Thing', 'MusicAlbum']
description: Studio album by The Beatles
url: http://www.thebeatles.com/#/albums/The_Beatles
resultScore: 193.815208056641

-----
@id: https://www.google.com/search?kgmid=m/01kyvd
name: A Hard Day's Night
@type: ['Thing', 'Movie']
description: 1964 film
url: N/A
resultScore: 119.2252807617188

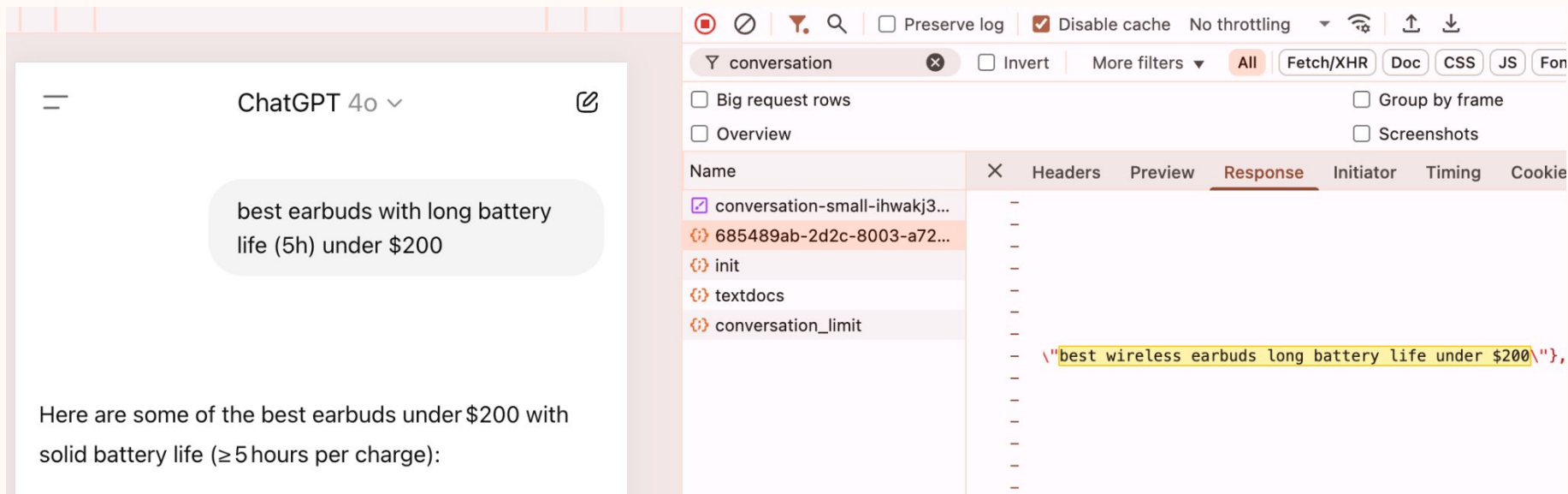
-----
@id: https://www.google.com/search?kgmid=m/0b45_s
name: Help!
@type: ['Thing', 'Movie']
description: 1965 film
url: N/A
resultScore: 115.6331329345703
```



05

ChatGPT Query Extractor (Bookmarklet)

When you search something on ChatGPT, they transform your prompt into multiple queries sent to Bing



The image shows a side-by-side comparison of a ChatGPT chat window and a browser's developer tools network tab. On the left, the ChatGPT interface displays a user prompt: "best earbuds with long battery life (5h) under \$200". Below the prompt, the model's response begins: "Here are some of the best earbuds under \$200 with solid battery life (≥5 hours per charge):". On the right, the browser's developer tools network tab is open, showing a list of network requests. The selected request is "conversation-small-ihwakj3...", which is a POST request to the OpenAI API. The response tab is active, showing the JSON response from the API. The response is a JSON object with a "choices" array containing one object with a "text" field. The text value is "best wireless earbuds long battery life under \$200", which is highlighted in yellow in the image. This demonstrates how the user's prompt is transformed into a specific query sent to the OpenAI API.

ChatGPT 4o ▾

best earbuds with long battery life (5h) under \$200

Here are some of the best earbuds under \$200 with solid battery life (≥5 hours per charge):

conversation

Big request rows

Overview

Name

conversation-small-ihwakj3...

685489ab-2d2c-8003-a72...

init

textdocs

conversation_limit

Headers

Preview

Response

Initiator

Timing

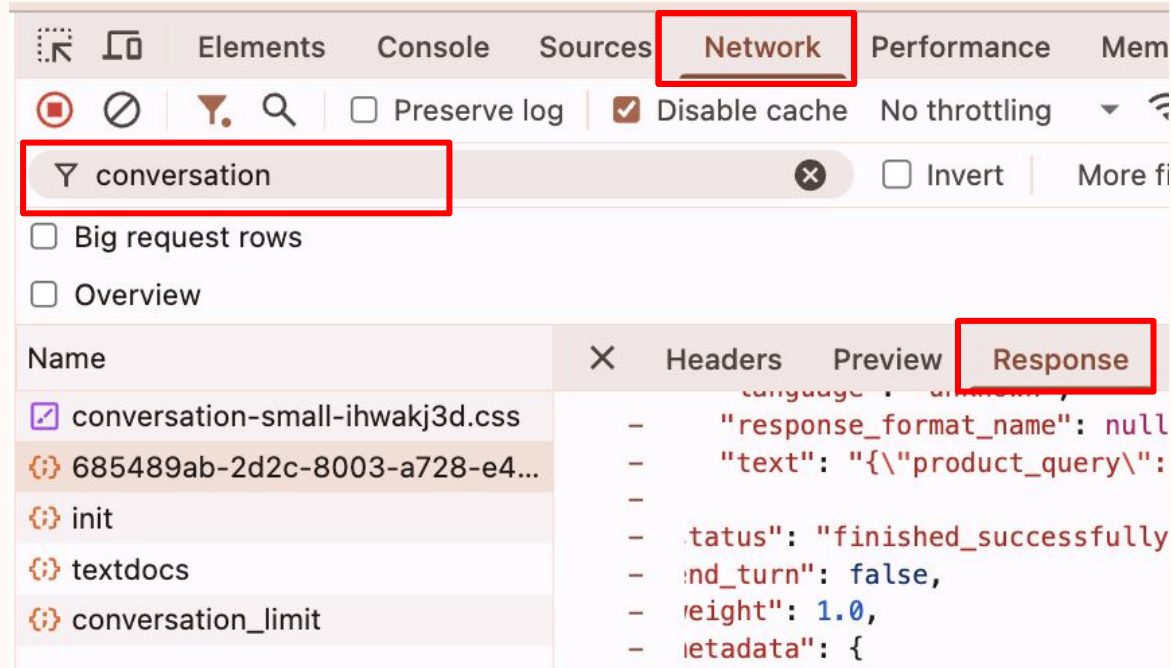
Cookie

Group by frame

Screenshots

best wireless earbuds long battery life under \$200

These queries are registered in your browser



Inspect > Network > “Conversation” > Response

Instead of digging through code, how about asking an LLM to create a bookmarklet?

1

2

3



ChatGPT 4o ▾



best earbuds with long battery
life (5h) under \$200

Here are some of the best earbuds under \$200 with
solid battery life (≥ 5 hours per charge):



ChatGPT Search Q...

ChatGPT Grounded Search Queries

Copy All

Found 2 search queries in this conversation:

best wireless earbuds long battery life under \$200

wireless earbuds 5h battery life review



Ziggy Shtrosberg

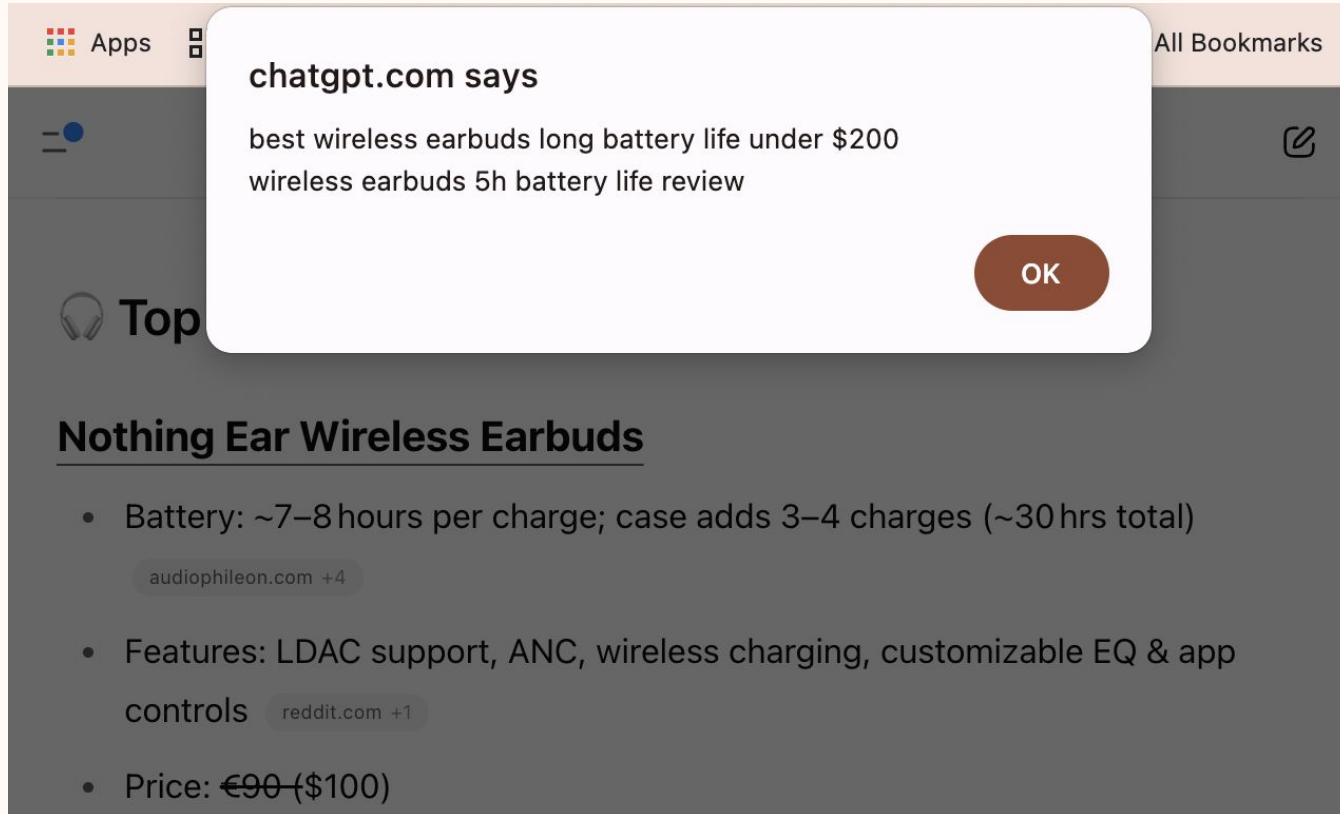


EXAMPLE: How to ask ChatGPT to write your code

I want to create a javascript bookmarklet to add on Chrome.

It should extract the contents inside search_query.

Not as pretty, but amazing for a two-line prompt



**Some more vibe
coding tips**

Always ask for code specific to the platform you intend to use

ModuleNotFoundError Traceback (most recent call last)

<ipython-input-4-10d778873865> in <cell line: 4>()

2 from itertools import combinations

3 import pandas as pd

----> 4 import win32com.client # Not available on Google Colab

5

6 # Define the function to split URLs

ModuleNotFoundError: No module named 'win32com'

You might get away with some errors

Successfully installed openai-1.86.0

```
ERROR: pip's dependency resolver does not currently take into account  
google-colab 1.0.0 requires pandas==2.2.2, but you have pandas 2.3.0  
google-colab 1.0.0 requires requests==2.32.3, but you have requests 2  
dask-cudf-cu12 25.2.2 requires pandas<2.2.4dev0,>=2.0, but you have p  
cudf-cu12 25.2.1 requires pandas<2.2.4dev0,>=2.0, but you have pandas  
Successfully installed openai-1.86.0 pandas-2.3.0 requests-2.32.4
```

Choose files No file chosen

Cancel upload

Tweak your ask until it does what you need

Generated random numbers

I need a Python script for Google Colab that can:

Calculate and display the cumulative percentages for both traffic and revenue, showing how many pages represent 20%, 40%, 60%, 80%, and 100% of the total.

Plot two graphs: one for cumulative traffic and one for cumulative revenue distribution across the pages.

Find the URLs that represent 50% of the traffic and revenue.

Combine the URLs into one file and automatically export this data to a new CSV file.

< 1/2 >

Allowed CSV upload

I need a Python script for Google Colab that can:

Calculate and display the cumulative percentages for both traffic and revenue, showing how many pages represent 20%, 40%, 60%, 80%, and 100% of the total.

Plot two graphs: one for cumulative traffic and one for cumulative revenue distribution across the pages.

Find the URLs that represent 50% of the traffic and revenue. Try to overlap URLs as much as possible, but it's ok if there's no perfect match. Traffic or revenue might be a little higher than 50%.

The input will be given by a CSV file I'll upload.

Combine the URLs into one file and automatically export this data to a new CSV file.

< 2/2 >

Give explicit and direct instructions

I will upload a CSV with headers
[A, B]

Combine both results in one
CSV file

I want a file that downloads
automatically after the numbers
are calculated

Change the input source to use
Excel files

**ALWAYS sense check the results,
don't blindly trust the output.**



ChatGPT & Colab to build SEO tools

By Gus Pelogia



<https://www.wix.com/seo/learn/resource/chatgpt-google-colab-seo-tools>

A man in a white t-shirt and blue jeans stands on the left side of a stage, gesturing with his hands while presenting to a large audience seated in rows of chairs. The audience is diverse in age and appearance, many wearing lanyards. The room has a high ceiling with recessed lighting and a large speaker on the right. A white text box is overlaid in the center-right of the image.

Thank You!
guspelogia.com
Add me on LinkedIn!

CREDITS: This presentation template was created by Slidesgo, and includes icons by Flaticon and infographics & images by Freepik