AI Tools for SEO: From LLM Workflows to MCP

WIX STUDIO

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.

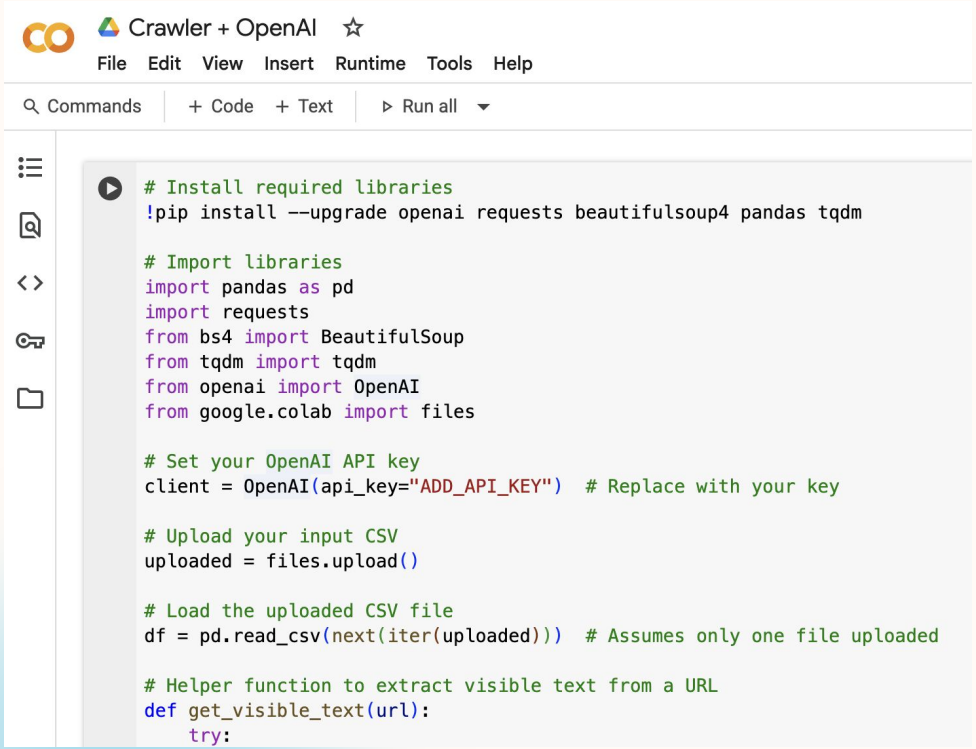
CSV File Format:

- The CSV should have the URLs in the

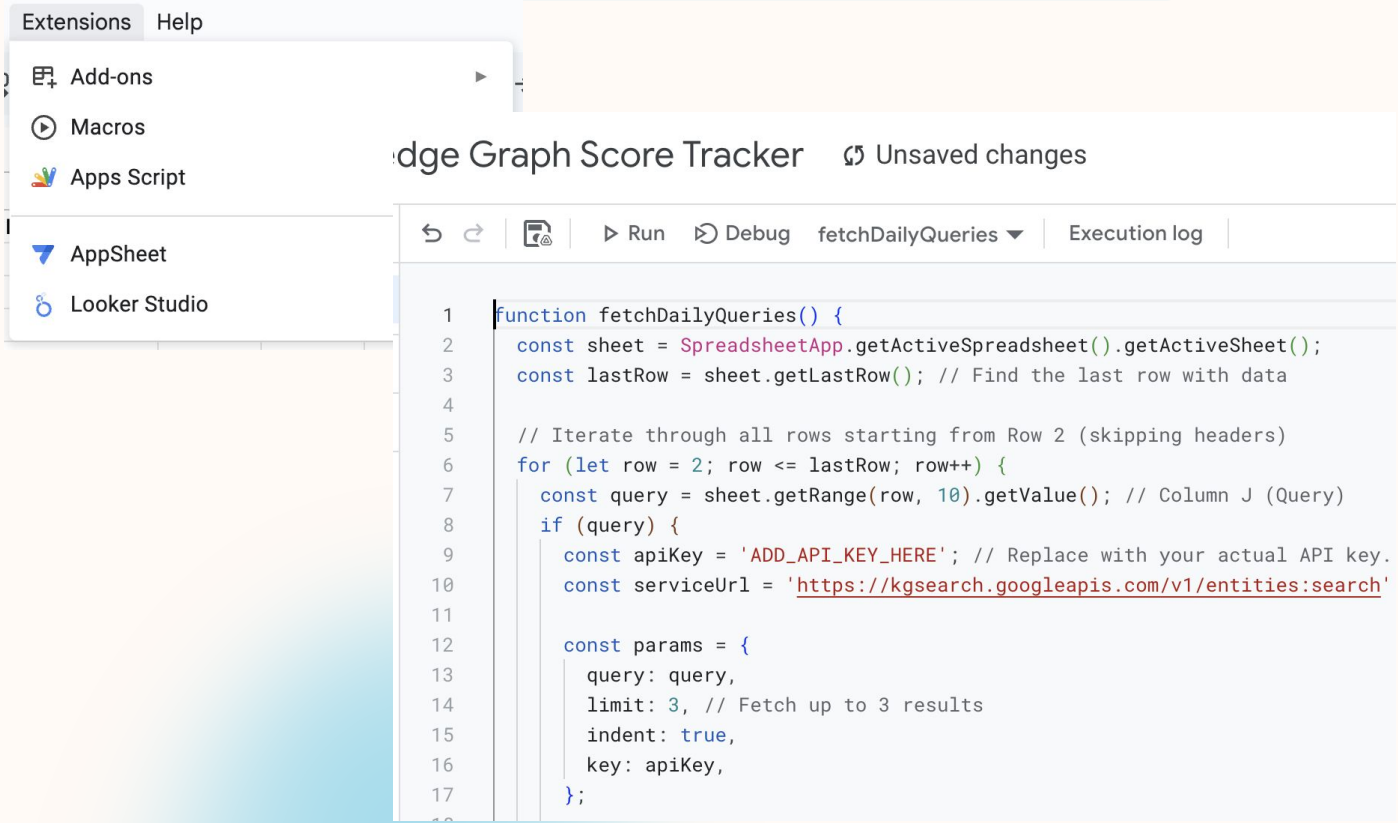
Url Embeddings Search

```
    # Extract the embeddings matrix
    embeddings = df.iloc[:, 1:].values

    # Calculate cosine similarity between all embeddings
    similarities = cosine_similarity(embeddings)

    # Set similarity of the initial index with itself to -1 (to exclude it)
    similarities[initial_index, initial_index] = -1

    # Find the five closest embeddings
    closest_indices = np.argsort(similarities[initial_index])[::-1][:5]

    # Get the corresponding URLs
    closest_urls = df.iloc[closest_indices, 0]
    return closest_urls

# Main code
# Step 1: Upload the embeddings file (must be a CSV with URL as the first column)
print("Please upload the URL embeddings CSV file (URLs and their vector embeddings)...")
df = upload_embeddings()

# Step 2: Input the initial URL to find related pages
initial_url = input("Enter the URL for which you want to find related pages: ")
initial_index = df[df.iloc[:, 0] == initial_url].index

if len(initial_index) == 0:
    print("The provided URL is not in the uploaded embeddings.")
else:
    # Step 3: Find and display the five closest related pages
    initial_index = initial_index[0]
    closest_pages = find_closest_embeddings(df, initial_index)
    print("\nThe five closest related pages are:")
    for url in closest_pages:
        print(url)
```

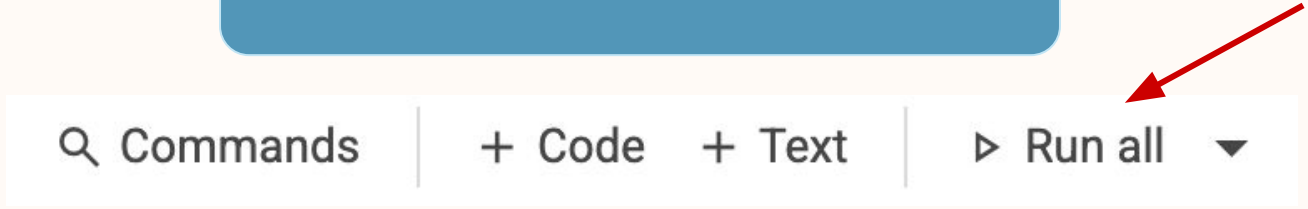
Google Colab (Python)



Google Sheets App Scripts (JavaScript)



Copy, Paste and Run



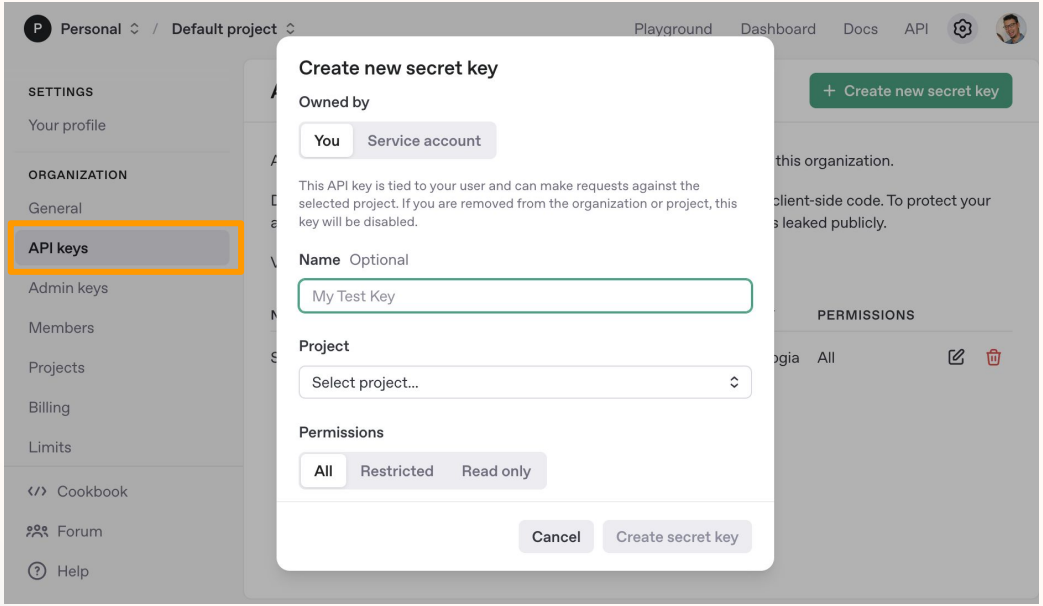

```python
from openai import OpenAI
from google.colab import files

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


```
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
```

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	(2k / 1M) × ($2 + $8) = $0.02	**$20**
GPT-4.1 mini	$0.40	$1.60	(2k / 1M) × ($0.4 + $1.6) = $0.004	**$4**
GPT-4o-mini	$0.15*	$0.60*	(2k / 1M) × ($0.15 + $0.60) = $0.0015	**$1.50**

What to ask

How to ask

Refine your ask

01

Traffic & Revenue Overlap

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

This allows you to find the “sweet spot” between both groups

```
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 i
file_name = list(uploaded.key
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 DataFr
    df_sorted_by_traffic = df
    df_sorted_by_revenue = df

    # Step 5: Calculate cumul
    df_sorted_by_traffic['cumu                                    raffic['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 traff
    for percentage in [20, 40
        page_count = df_sorte                                    0]
        print(f"First {page_c

    # Step 7: Create text out
    print("\nCumulative reven
    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.

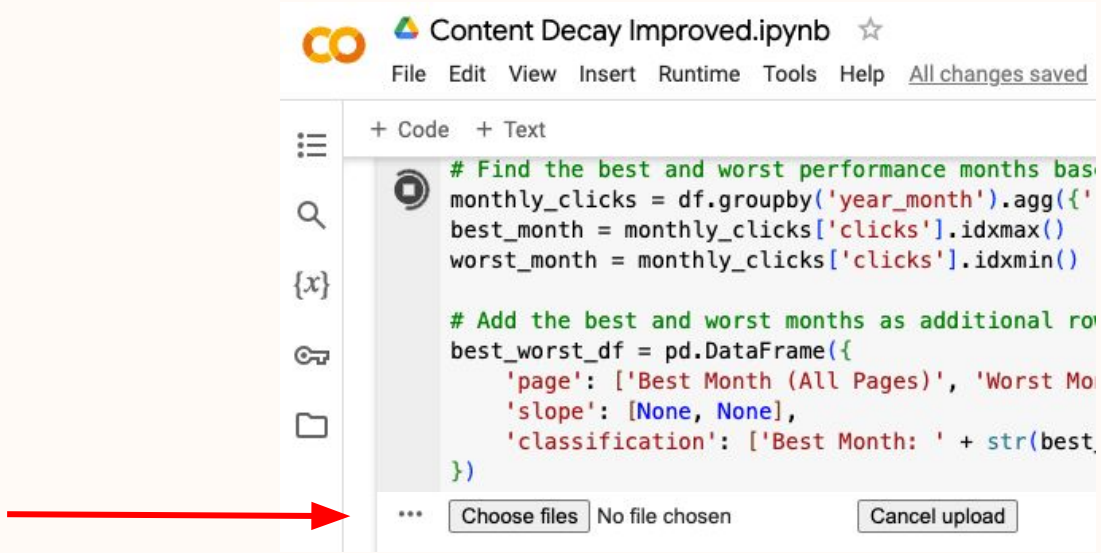
The input will be given by a CSV file I'll upload. Combine the URLs into one file and automatically export this data to a new CSV file.

Google Analytics 4

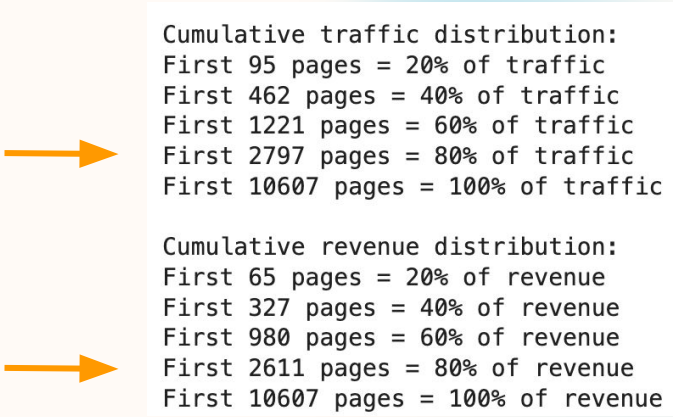
Traffic Revenue Analysis

```
'URL'].tolist()
revenue_50_percent_urls = revenue_sorted.loc[:revenue_50_percent_index,
'URL'].tolist()

# Find overlapping URLs between traffic and revenue 50%
overlapping_urls = list(set(traffic_50_percent_urls) & set(revenue_50_percent_urls))
print(f"Number of URLs representing ~50% of both Traffic and Revenue:
{len(overlapping_urls)}")

# Step 7: Export combined data to a new CSV file
data_combined = pd.merge(traffic_sorted[['URL', 'Traffic', 'Cumulative Traffic %']]
                         revenue_sorted[['URL', 'Revenue', 'Cumulative Revenue %']]
                         on='URL',
                         how='outer').fillna(0)

output_filename = 'combined_traffic_revenue_data.csv'
data_combined.to_csv(output_filename, index=False)
print(f"Combined data export ●
```

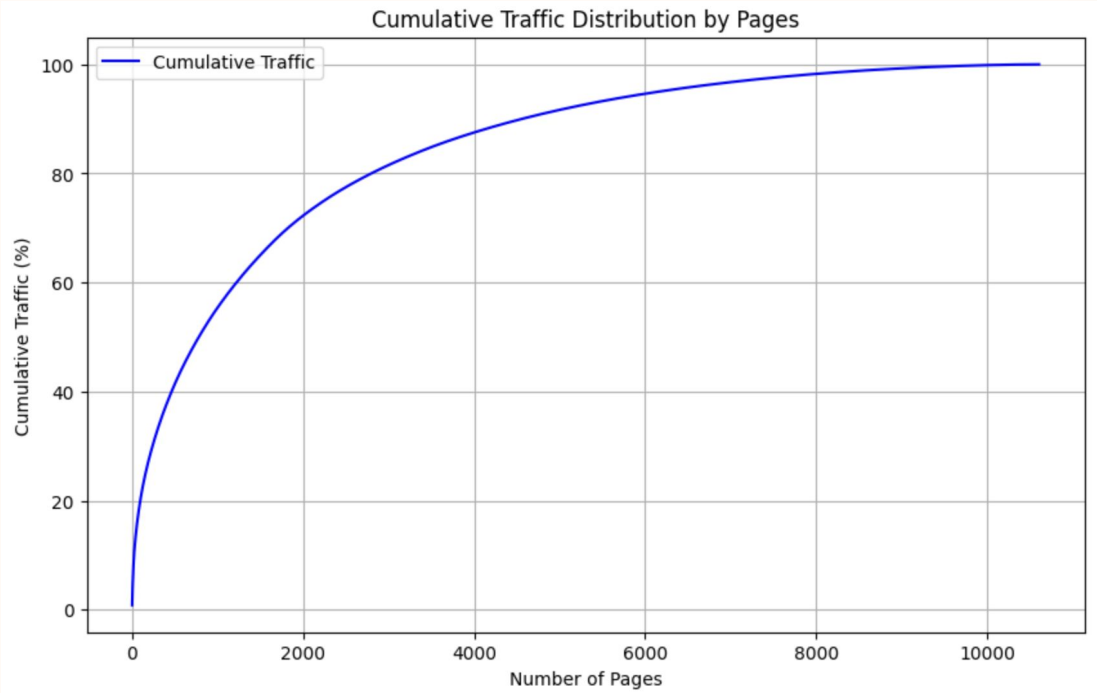
Upload your export from analytics: date, page and traffic columns



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



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
df = pd.read_csv(file_name)

# Step 3: Ensure the CSV has
if 'traffic' not in df.column
    print("Error: CSV must co
else:
    # Step 4: Sort the DataFra
    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 cumul
    df_sorted_by_traffic['cum                                                raffic['traffic'].sum() * 100
    df_sorted_by_revenue['cum                                                evenue['revenue'].sum() * 100

    # Step 6: Create text out
    print("\nCumulative traff
    for percentage in [20, 40,
        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 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")
```

Crawl URLs just by adding them on a spreadsheet

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.

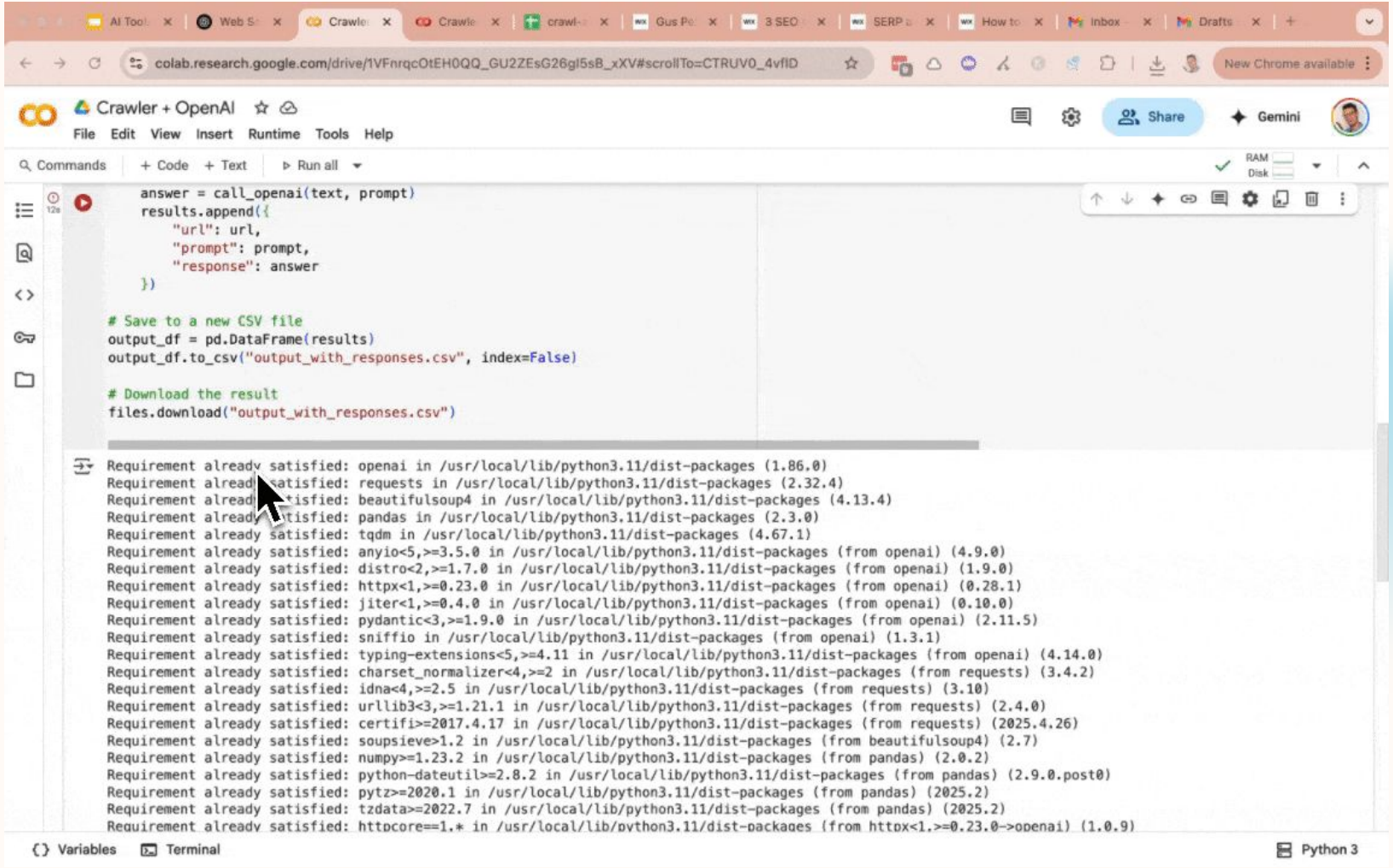
You can run `openai migrate` to automatically upgrade your codebase to use the 1.0.0 interface.

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`)**


colab.research.google.com/drive/1VFnrqcOtEH0QQ_GU2ZEsG26gl5sB_xXV#scrollTo=CTRUV0_4vfID
Crawler + OpenAI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
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)
Variables
Terminal
Python 3

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)


Internal link opportunities at scale
Tag pages
Content clustering

Vector Embeddings + Cosine Similarity

What are vector embeddings?

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

Address	(ChatGPT) Extract embeddings from page conten...
https://www.guspelogia.com/	-0.013361351,0.00984234,0.00978108,0.013817393,-...
https://www.guspelogia.com/blog	-0.023292037,0.014379196,-0.0013030263,0.013249...
https://www.guspelogia.com/map-related-pages-embeddings	-0.03336033,0.028152429,-0.003392576,0.00800528...
https://www.guspelogia.com/book-online	-0.034532774,-0.010436008,0.048117362,-0.006128...
https://www.guspelogia.com/write-seo-dev-tickets	-0.014598452,0.00513234,0.014722849,-0.00158698...
https://www.guspelogia.com/google-dma-serp-changes	-0.011425617,0.084135699,0.013785663,0.01873052,-...
https://www.guspelogia.com/public-speaking	-0.021230659,-0.010016579,0.030386535,0.0015397...
https://www.guspelogia.com/screaming-frog-chatgpt-paginas-rel...	-0.025497664,0.031419784,-0.0064167166,0.012677...
https://www.guspelogia.com/how-to-build-knowledge-panel	-0.008920509,0.024403073,-0.009870828,0.0444499...
https://www.guspelogia.com/combining-internal-and-external-lin...	-0.019747287,-0.005057232,0.025112234,0.0196402...
https://www.guspelogia.com/seo-product-manager-job	-0.0077240844,0.049245507,0.021521686,0.008645...
https://www.guspelogia.com/skip-the-slides-seo-business-case	-0.036185283,0.053586677,0.01555306,0.00160884...
https://www.guspelogia.com/blog/search	0.01651117,-0.0191009,0.0140529,0.009767356,-0.0...
https://www.guspelogia.com/business-acumen-seo	-0.033323254,0.023986373,0.015445661,0.0192383...
https://www.guspelogia.com/product-management-principles-seo	-0.018450664,0.020344952,0.014251907,0.0241186...
https://www.guspelogia.com/prd	0.03177193,-0.021666612,-0.0039030174,-0.000030...

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

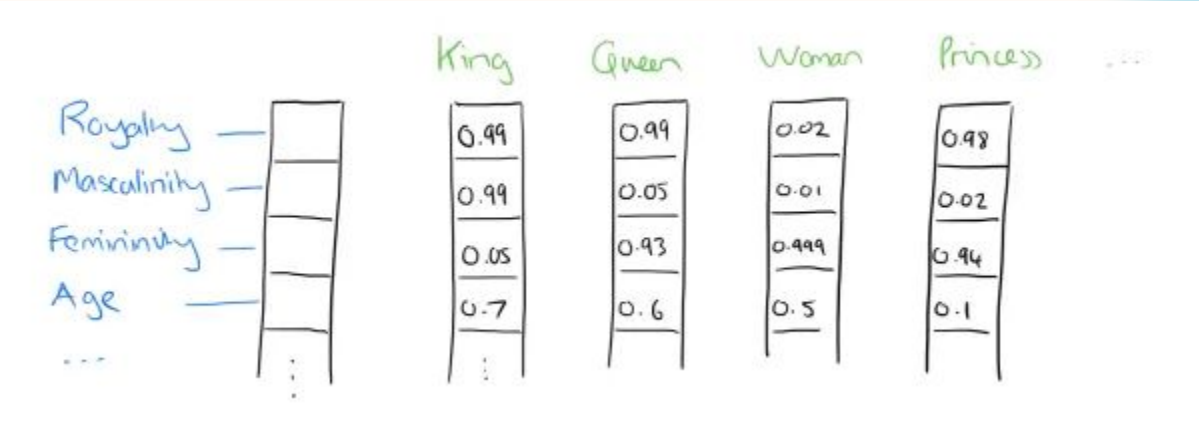
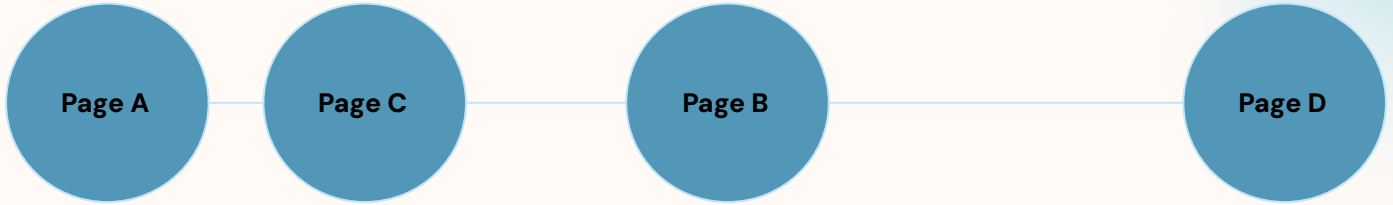
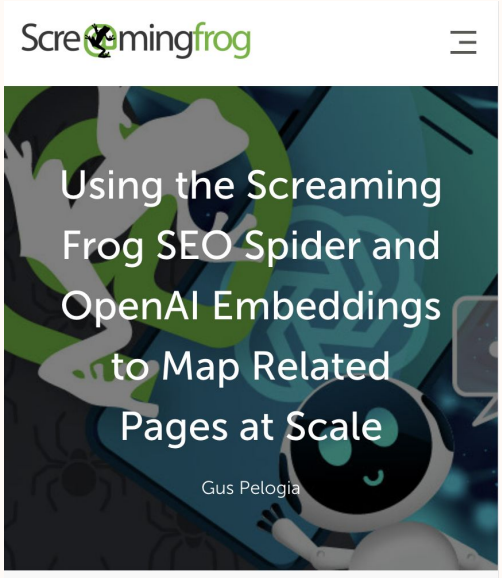


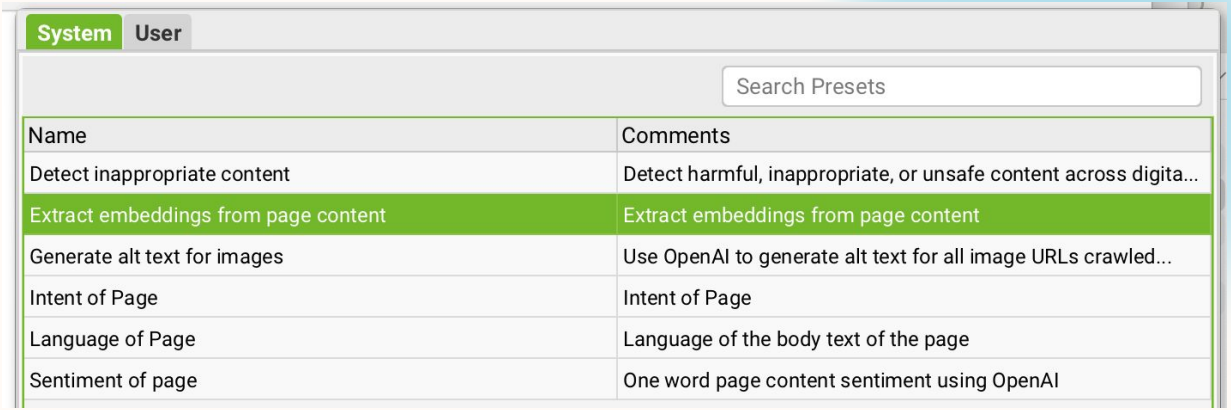
Image source: medium.com/artifical-coder

“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





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.

ChatGPT 4o with canvas

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

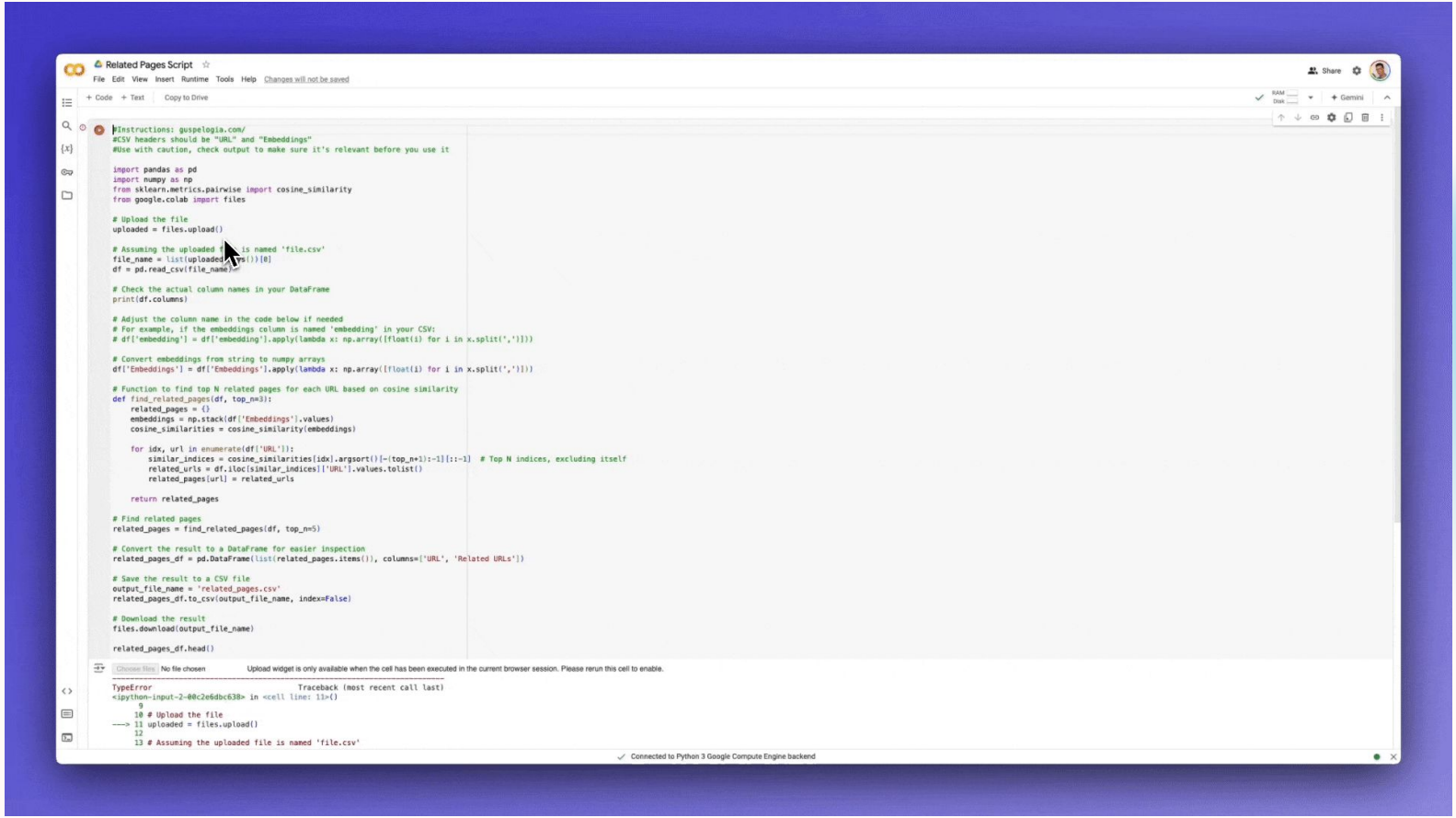
This code reads the input CSV, converts embeddings from strings to numpy arrays, computes cosine similarity, and identifies the top 5 related URLs for each URL. The resulting CSV will have the "URL" and "Related URLs" columns.

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

```python
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
import ast

# Load the CSV file
input_file = "urls_with_embeddings.csv"  # Replace with your file name
output_file = "related_urls.csv"

# Read the CSV into a DataFrame
df = pd.read_csv(input_file)

# Convert embeddings from string to numpy array
df["Embeddings"] = df["Embeddings"].apply(lambda x: np.array(ast.literal_eval(x)))

# Calculate cosine similarity between embeddings
embeddings_matrix = np.stack(df["Embeddings"].values)
cosine_sim_matrix = cosine_similarity(embeddings_matrix)

# Find the top 5 related URLs for each URL
related_urls = []
for idx, row in df.iterrows():
    # Get similarity scores for the current URL, excluding itself
    similarity_scores = cosine_sim_matrix[idx]
    related_indices = np.argsort(similarity_scores)[-6:-1][::-1]  # Get top 5 excluding itself
    top_related_urls = df.iloc[related_indices]["URL"].tolist()
    related_urls.append(top_related_urls)

# Add the related URLs to the DataFrame
df["Related URLs"] = [", ".join(urls) for urls in related_urls]

# Save the result as a new CSV file
df[["URL", "Related URLs"]].to_csv(output_file, index=False)

print(f"Related URLs saved to {output_file}")
```

Run Colab Notebook + Results



04

Knowledge Panel Tracker (Google Sheets)

```python
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.column
    print("Error: CSV must co
else:
    # Step 4: Sort the DataFra
    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 cumul
    df_sorted_by_traffic['cum                                                affic['traffic'].sum() * 100
    df_sorted_by_revenue['cum                                                venue['revenue'].sum() * 100

    # Step 6: Create text out
    print("\nCumulative traff
    for percentage in [20, 40,
        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 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")
```

Find entities on Google

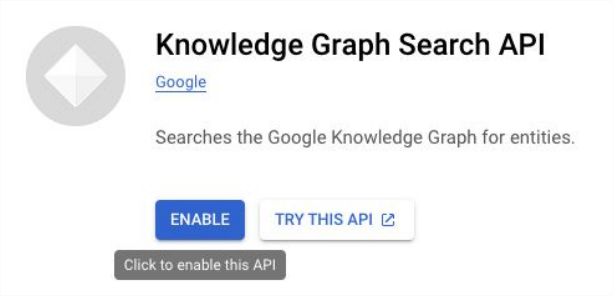
Person, Organization, Thing

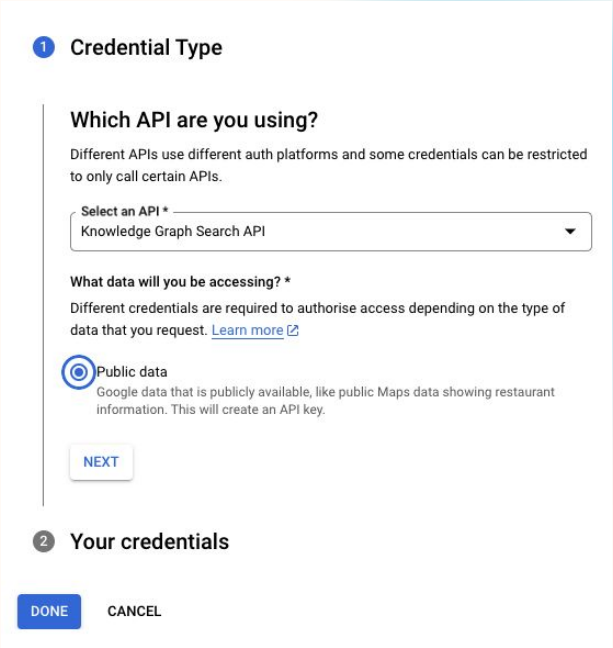
This is where you find if Google recognizes as entity

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

First, get your Google Cloud API

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





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.

ChatGPT 4o with canvas

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

```python
import requests
import json

# Set your Google Knowledge Graph API key here
API_KEY = 'YOUR_GOOGLE_API_KEY'

def search_knowledge_graph(query):
    url = "https://kgsearch.googleapis.com/v1/entities:search"
    params = {
        "query": query,
        "key": API_KEY,
        "limit": 1,
        "indent": True,
    }

    response = requests.get(url, params=params)
    if response.status_code == 200:
        results = response.json()
        if "itemListElement" in results and len(results["itemListElement"]) > 0:
            element = results["itemListElement"][0]["result"]
            print("@id:", element.get("@id", "N/A"))
            print("name:", element.get("name", "N/A"))
            print("@type:", element.get("@type", "N/A"))
            print("description:", element.get("description", "N/A"))
            print("url:", element.get("url", "N/A"))
            print("resultScore:", results["itemListElement"][0].get("resultScore", "N/A"))
        else:
            print("Query is not present on Knowledge Graph panel.")
    else:
        print("Error in API request. Status code:", response.status_code)

# Set the query you want to search for
query = "Taylor Swift"

# Call the function to search
search_knowledge_graph(query)

```

Step 3: Run Notebook and collect results

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

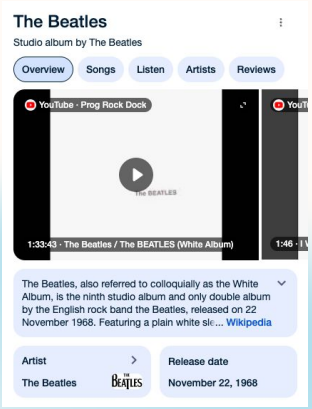
Knowledge Graph API

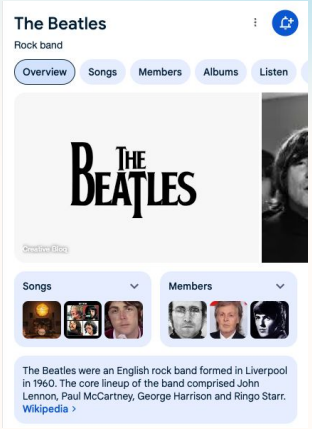
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
# 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('-' * 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/01hg4bh
name: The Beatles
@type: ['Thing', 'MusicAlbum']
description: Studio album by The Beatles
url: http://www.thebeatles.com/#/albums/The_Beatles
resultScore: 193.8152008056641
----------------------------------------
@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
```

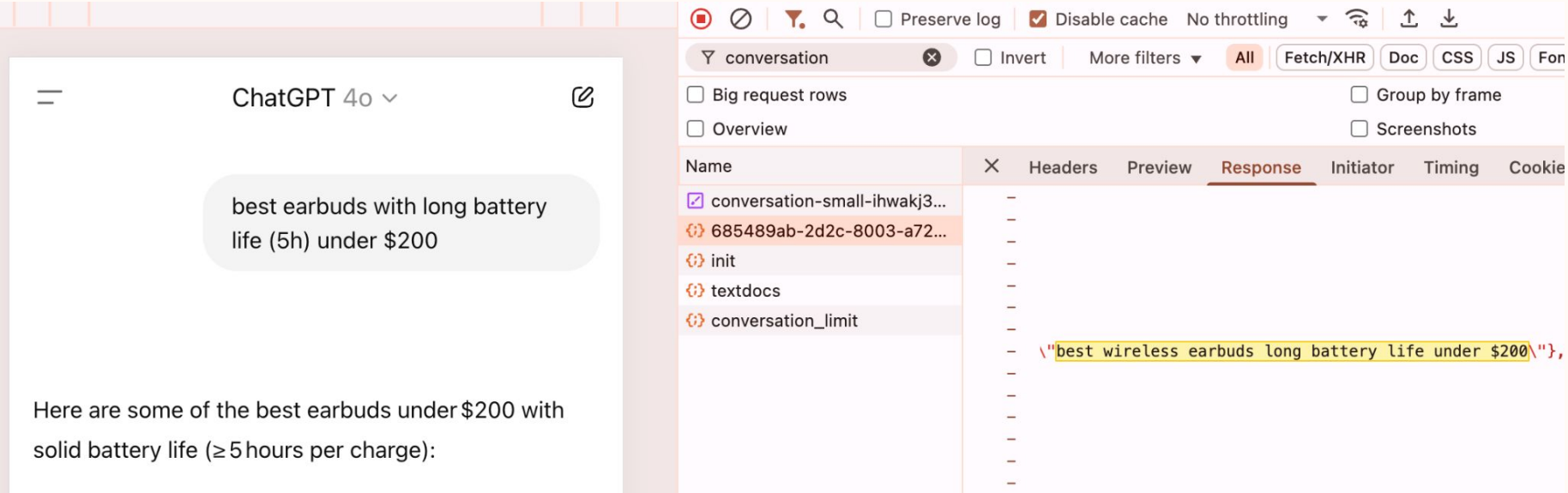




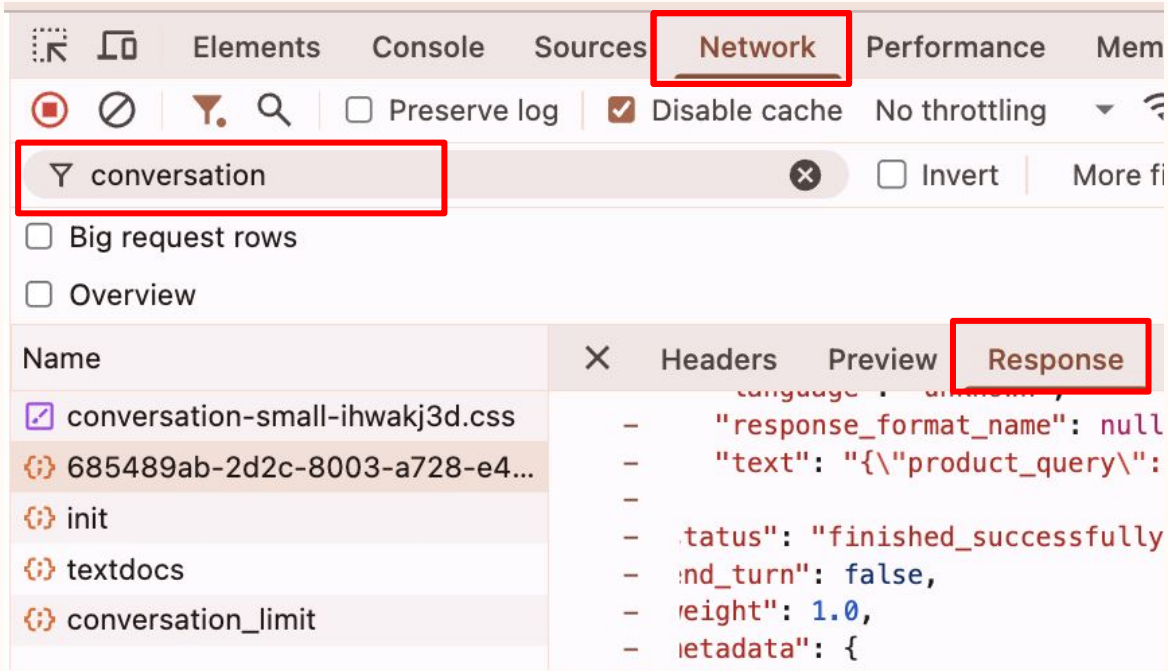
05

ChatGPT Query Extractor (Bookmarklet)

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



These queries are registered in your browser



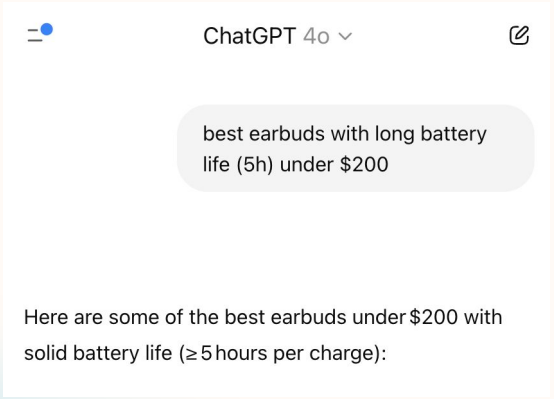
Inspect > Network > “Conversation” > Response

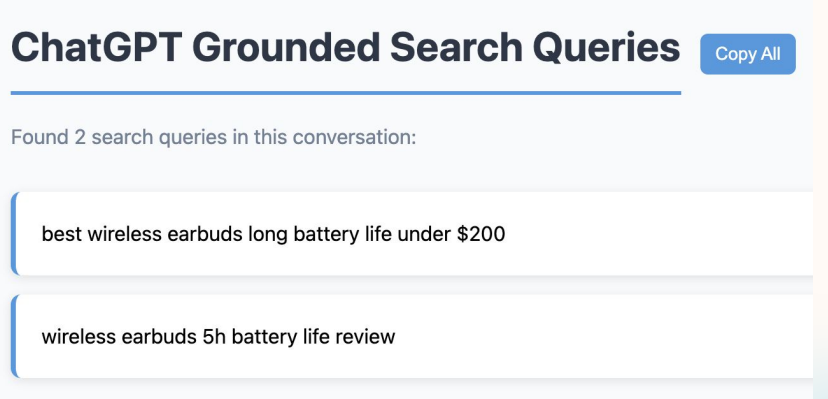
Instead of digging through code, how about asking an LLM to create a bookmarklet?

1

2

3





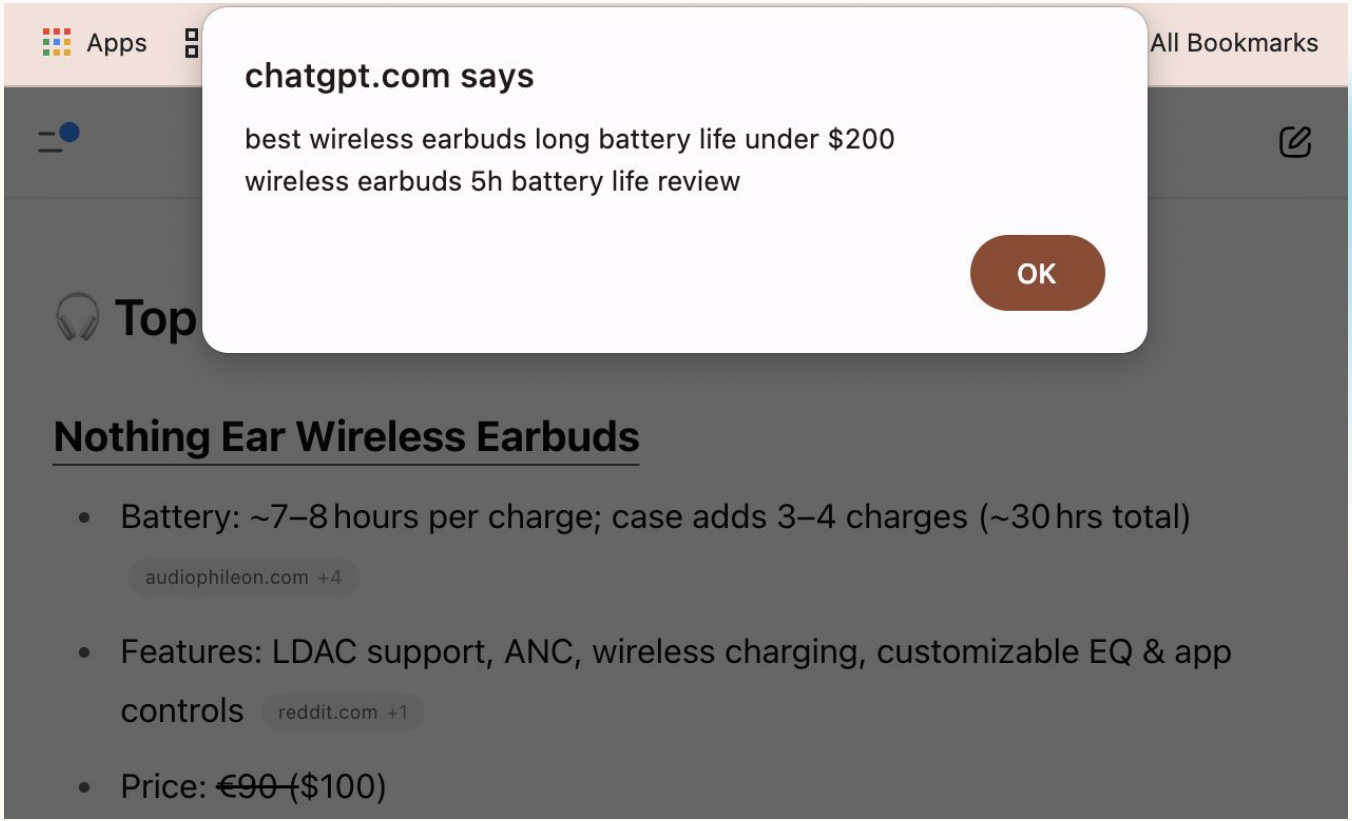
Ziggy Shtrosberg

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

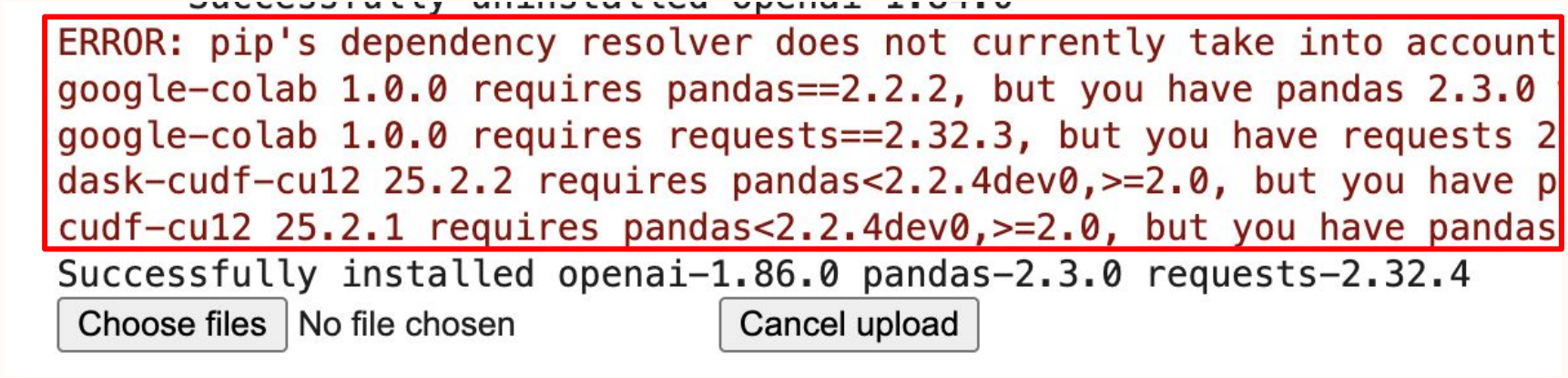


Some more vibe coding tips

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



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

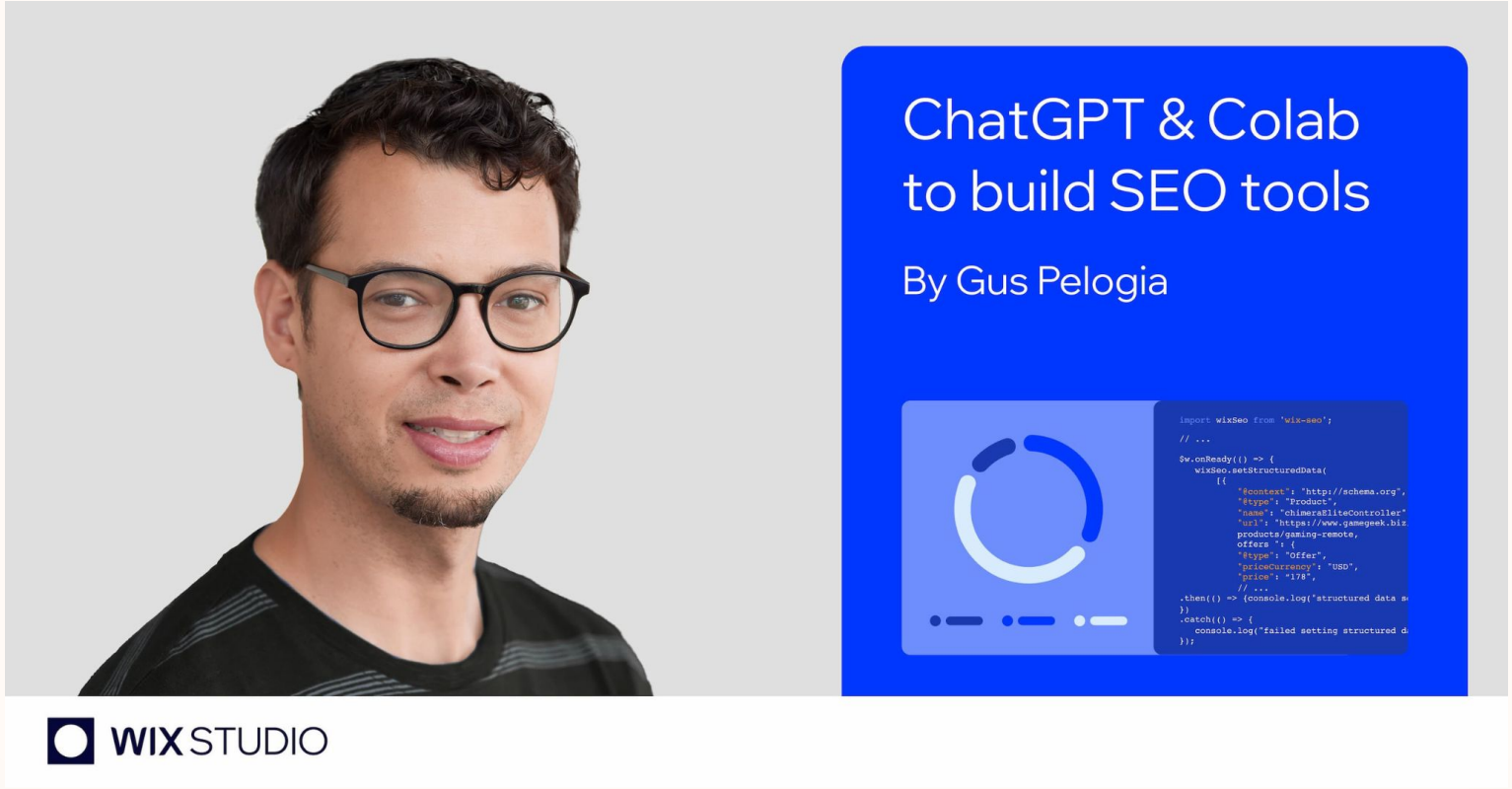
I will upload a CSV with headers [A, B]

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.



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

Thank You!
guspelogia.com
Add me on LinkedIn!