Share of Search

Charles Shaw

Abstract

The Share of Search tool leverages Google Trends data via SerpAPI to analyze and visualize search interest over specified time frames for a variety of queries. This sophisticated tool is designed not only to assess and compare the relative popularity of different search terms within specified geographical regions but also to uncover related queries and topics that are emerging as areas of interest. By providing insights into market trends, competitive landscapes, and consumer behaviour, the tool aids marketers, researchers, and analysts in making informed strategic decisions. Additionally, the tool's ability to generate and export detailed visualizations and data reports enhances its utility for a comprehensive analysis of search trends. With its robust data fetching capabilities and flexible configuration options, the Share of Search tool serves as a practical resource for tracking digital trends and understanding the dynamics of online searches.

This version: May 31, 2024

Contents

1. Introduction

Share of Search is an analytical tool that leverages Google Trends data through SerpAPI to provide insights into search trends over specified periods. It allows users to compare the popularity of up to five different search terms simultaneously within a given geographical area. This tool is particularly useful for marketers, researchers, and analysts looking to understand market dynamics, consumer interest, and competition.

2. Setup and Initialization

To utilize the Share of Search tool, several components need to be installed and set up correctly. The system requirements and installation steps are outlined below:

2.1. System Requirements

- Python 3.6 or higher
- Pip (Python package installer)
- Access to an internet connection for API requests

2.2. Installation

Install the required Python packages using pip:

```
pip install -q serpapi pandas matplotlib seaborn
```

3. Configuration

Configuration involves setting API keys, defining queries, geographic location, and date range for analysis.

3.1. API Key Configuration

Set the SerpAPI key in the environment or directly within the script. This API key is used to authenticate requests to the SerpAPI service.

```
API_KEY = "your_serpapi_key_here"
```

3.2. Query Configuration

Define up to five queries to analyze trends over time. Set queries in the tool's configuration to track and compare search interests across different terms. Example queries include "Virgin Money", "Lloyds Bank", "Barclays", "Santander Bank", and "NatWest".

```
QUERIES = ["Virgin Money", "Lloyds Bank", "Barclays", "Santander Bank", "NatWest"]
```

3.3. Geographic Location

Configure the geographic scope of the analysis by setting the geographical code. This code can represent a country, such as 'GB' for Great Britain, or a more specific region. e.g., 'US-AL' for Alabama, 'GB-ENG' for England. To set it to Great Britain and USA do this geo = ["GB", "US"]. Defaults to "World".

```
GEO = "GB"
```

3.4. Date Configuration

The date range for the analysis can be configured to suit different analytical needs. The default setting is the past five years ('today 5-y'). For last 3 years use 'today 3-y'. For more specific time frames, configure exact dates.

```
DATE = "today 5-y"
```

3.5. Granularity and Smoothing

Set the granularity of the data and the smoothing period to reflect shorter-term trends. Can be 'weekly' ("W"), or 'monthly'. Default is 'M' (monthly). 'daily' ("D") is currently experimental.

```
GRANULARITY = "M"
SMOOTHING_PERIOD = 60
```

4. Advanced Configuration

Leverage advanced features to enhance the accuracy and relevance of your search data analysis.

4.1. Keyword Lists

Define keyword lists to refine the scope of your search data retrieval. This list can include broad categories or specific items.

```
QUERIES = ["Virgin Money", "Lloyds Bank", "Barclays", "Santander Bank", "NatWest"]
```

4.2. Advanced Keywords

Google Trends offers suggested narrowed search terms that can provide more targeted data. Use the `get_suggestions()` function to find encoded topics and choose the most relevant one for your analysis.

```
# Example of using an encoded topic for a specific search term
ADVANCED_KEYWORDS = ["/m/025rw19"]  # Iron Chemical Element
```

5. Usage

Execute the tool from the command line or an IDE supporting Python.

5.1. Running the Tool

Navigate to the script directory and execute:

```
python share_of_search.py
```

5.2. Outputs

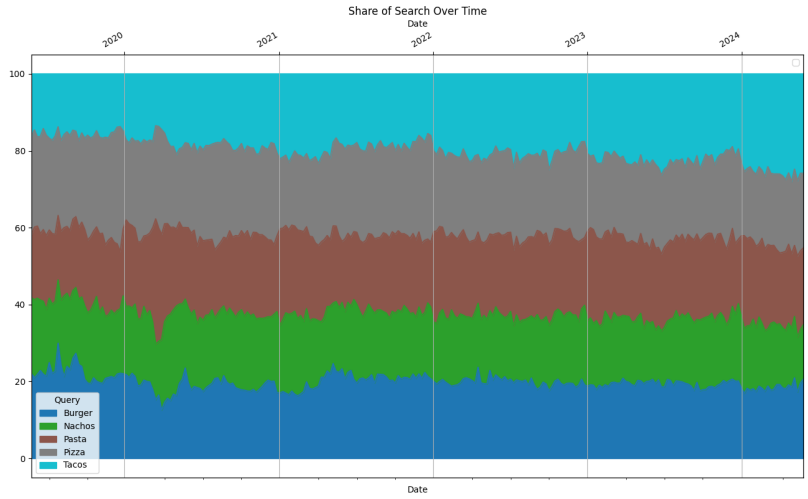
Outputs include:

- Time series plots of search interest.
- CSV files containing raw and processed data.
- Smoothed interest plots to identify general trends.

The "Share of Search" tool provides several types of visual outputs that help in analysing search trends over time for various queries. Below are examples of the charts generated by the tool:

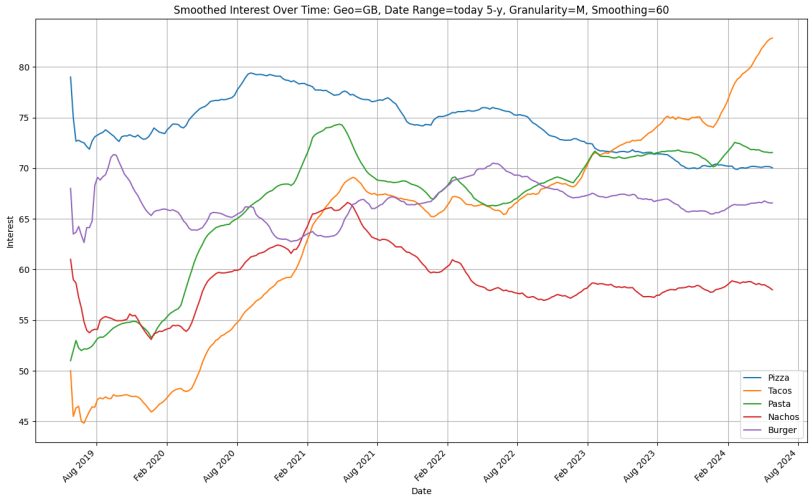
5.2.1. Share of Search Over Time

Figure 1: Share of Search Over Time: An area chart showing the proportion of search interest among the queries over time. This visualization helps in comparing the relative popularity or market share of search terms within a geographic locale.



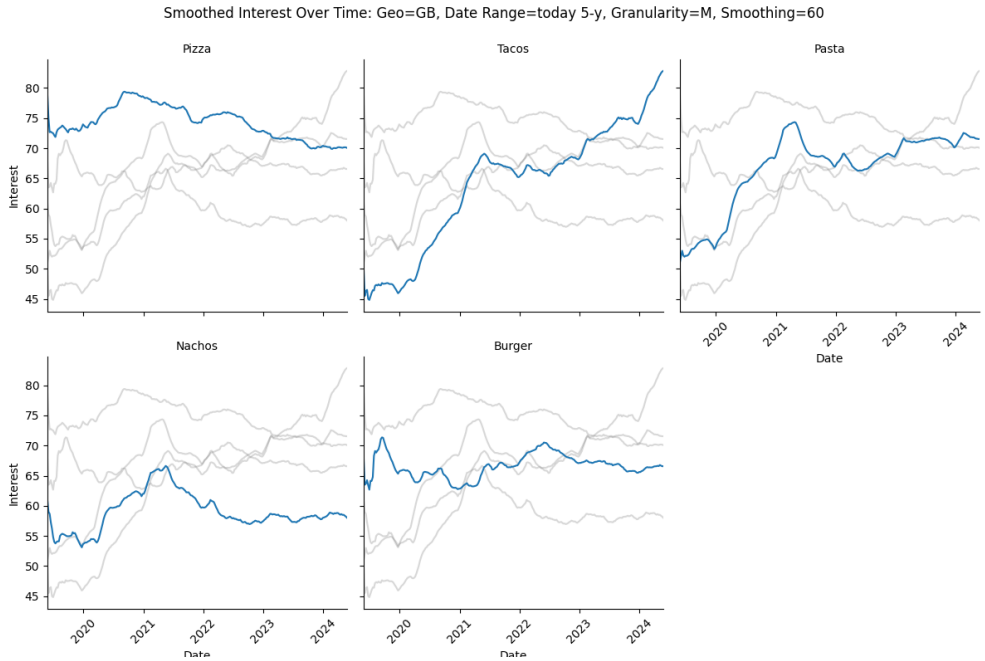
5.2.2. Smoothed Interest over Time

Figure 2: Smoothed Interest over Time: This chart smooths out the fluctuations seen in the 'Interest over Time' to highlight underlying trends in search interest, providing a clearer picture of long-term movements without short-term volatility.



5.2.3. Smoothed Interest over Time (Split View)

Figure 3: Smoothed Interest over Time (Split View): A panel of charts showing smoothed interest over time for different queries. Each panel highlights the individual query while graying out others.



6. Function Definitions

Below are detailed descriptions of the functions used in the script:

6.1. fetch_data

Fetches Google Trends data for a given query and geographic location.

```
def fetch_data(query, geo, date, api_key):
    """
    Fetches Google Trends data for a given query and geographic location.
    Args:
        query (str): The search query term.
        geo (str): The geographic location for the search query.
        date (str): The date range for the search query.
        api_key (str): The API key for authenticating the request.
    Returns:
        list: A list of dictionaries containing the timeline data.
    Raises:
        ChunkedEncodingError: If a chunked encoding error occurs after retrying.
    """
```

6.2. parse_date_range

Parses a date range string and returns the start and end dates.

```
def parse_date_range(date_str):
    """
    Parses a date range string and returns the start and end dates.
    Args:
        date_str (str): A string representing the date range in the format "<years>-y".
    Returns:
        tuple: A tuple containing the start date (datetime) and end date (datetime).
    Raises:
        ValueError: If the date string format is not recognized.
    """
```

6.3. plot_share_of_search

Plots the share of search over time for different queries as a stacked area chart.

```
def plot_share_of_search(df):
    """
    Plots the share of search over time for different queries as a stacked area chart.
    Args:
        df (pd.DataFrame): DataFrame containing the data to plot.
    """
```

6.4. plot_smoothed_interest_over_time

Plots the smoothed interest over time for different queries.

```
def plot_smoothed_interest_over_time(df):
    """
    Plots the smoothed interest over time for different queries.
    Args:
        df (pd.DataFrame): DataFrame containing the data to plot.
    """
```

6.5. plot_smoothed_interest_over_time_split

Creates a panel of charts showing smoothed interest over time for different queries.

```
def plot_smoothed_interest_over_time_split(df):
    """
    Creates a panel of charts showing smoothed interest over time for different queries.
    Args:
        df (pd.DataFrame): DataFrame containing the data to plot.
    """
```

7. Data Handling and Export

The tool also provides functionalities to save the processed data into CSV files for further analysis.

```
# Save all DataFrames to CSV
combined_df.to_csv("Share_of_Search.csv", index=False)
smoothed_df.to_csv("Smoothed.csv", index=False)
```

8. Related Queries

This section of the tool fetches and visualizes related queries for specified search terms using Google Trends data via SerpAPI. It helps identify emerging trends and related topics that are gaining popularity, providing additional insights into user interests and market shifts.

8.1. Fetching Data

The function `fetch_related_queries` queries the SerpAPI for related search terms based on the user's input. It returns a list of rising and top queries associated with the primary search term.

```
def fetch_related_queries(search_term):
    # API parameters
    api_key = "your_api_key_here"
    params = {
        'engine': 'google_trends',
        'q': search_term,
        'data_type': 'RELATED_QUERIES',
        'api_key': api_key
    }
    url = 'https://serpapi.com/search.json'

    # Send GET request
    response = requests.get(url, params=params)

    if response.status_code == 200:
        return response.json()['related_queries']
    else:
        raise Exception("Failed to fetch related queries.")
```

8.2. Visualization

Visualizations of related queries provide a graphical representation of the data, highlighting trends and the relative search volume of related terms.

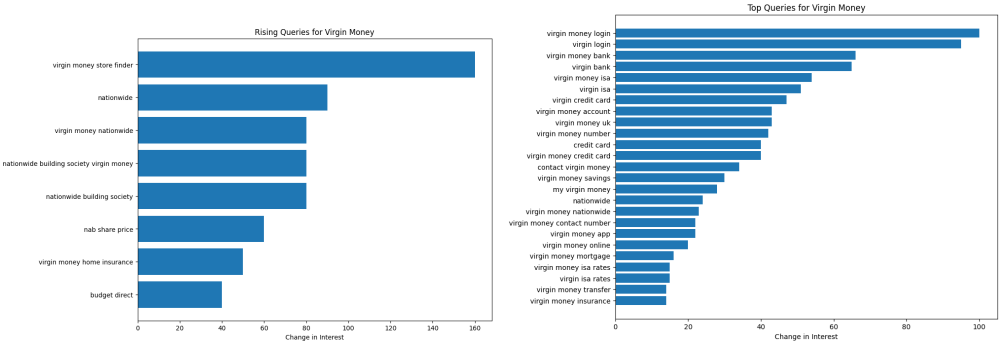
```
def plot_queries(queries, title):
    labels = [q['query'] for q in queries]
    values = [int(q['value'].strip('%+')) for q in queries]

    plt.figure(figsize=(10, 8))
    plt.barh(labels, values)
    plt.xlabel('Change in Interest')
    plt.title(title)
    plt.gca().invert_y_axis()
    plt.savefig(f"{title}.png")
    plt.show()
```

9. Related Topics

Similar to related queries, the tool also fetches related topics that are relevant to the search terms. This function utilizes the SerpAPI to pull data about topics related to the main query, providing deeper insights into broader subjects or categories linked with the search term.

Figure 4: Rising & Top Queries.



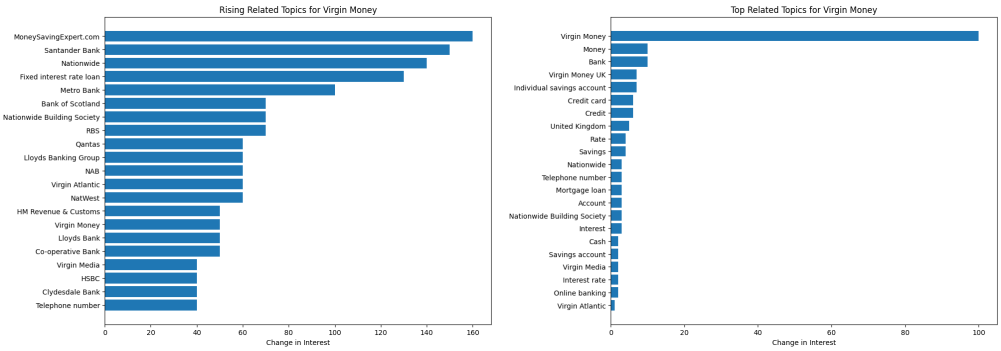
9.1. Fetching Data

Data for related topics is retrieved through a dedicated API endpoint, which categorizes topics by their rising and top status.

```
def fetch_related_topics(search_term):
    api_key = "your_api_key_here"
    params = {
        'engine': 'google_trends',
        'q': search_term,
        'data_type': 'RELATED_TOP toPICS',
        'api_key': api_key
    }
    url = 'https://serpapi.com/search.json'

    response = requests.get(url, params=params)
    if response.status_code == 200:
        return response.json()['related_topics']
    else:
        raise Exception("Failed to fetch related topics.")
```

Figure 5: Rising & Top Topics.



9.2. Visualization

Visualization for related topics is designed to showcase the popularity and relevance of each topic in a comparable graphical format.

```
def plot_topics(topics, title):
    labels = [t['topic']['title'] for t in topics]
    values = [int(t['value'].strip('%+')) for t in topics]

    plt.figure(figsize=(10, 8))
    plt.barh(labels, values)
    plt.xlabel('Change in Interest')
    plt.title(title)
    plt.gca().invert_y_axis()
    plt.savefig(f"{title}.png")
    plt.show()
```

These new sections provide detailed code and explanations for the functionalities related to "Related Queries" and "Related Topics". By incorporating these sections, you will be able to offer a more comprehensive guide to using all features of your tool.

10. Downloadable Outputs

The script can package the output files into a zip file for easy downloading.

```
from google.colab import files

def create_downloadable_zip(excluded_files, zip_name='SoS_output_files.zip'):
    # Start command
    command = "zip -r " + zip_name + " /content"

    # Exclude files
    for file in excluded_files:
        command += " -x " + "\"*/" + file + "*\""

    # Run the command
    os.system(command)
    # Download the file
    files.download(zip_name)

# List of files to exclude
excluded_files = ["drive", "sample_data", "raw_data.csv"]

# Download files
create_downloadable_zip(excluded_files)
```

11. Conclusion

The Share of Search tool provides a powerful way to visualize search trends and compare the popularity of different terms. It is particularly useful for marketers, researchers, and analysts looking to understand market dynamics, consumer behaviour, and competition.