AUTHORITYHACKER SERP STUDY 2019

"What Scrapping & Analyzing 1.1 Million Search Results Taught Us About The Way Google Ranks Your Content In 2019"

Introduction

As pointed out in the article, the SEO industry lacks enough studies. To our knowledge, this might be the largest independent study of search results that has been done in recent years.

The other studies notable studies are Backlinko's Search Engine Ranking Factors Study, and several others done by Ahrefs, such as their recent study of 2 Million Snippets.

While we consider both studies highly reliable and trustworthy, it's important to note that the former relied on donated data from vendors with business interests in SEO, and likewise Ahrefs is one of the dominant SEO vendors in the market.

We found it important to obtain all data independently, to benchmark those findings.

The study by Backlinko served as the original inspiration for organizing this analysis. We have loosely followed their structure.

ABOUT THE STUDY

We have taken 120,000 random keywords and obtained the top ten Google search results for each. Then, we took the relevant metrics from Ahrefs, Google APIs, and the data from our own, custom built crawler.

We worked with the following data points:

Keyword	Volume	URL	Country	Difficulty	Clicks
CPS	Return Rate	Parent Topic	Parent Topic Volume	Referring Domains	Domain rating
Ahrefs Rank	Traffic	Keywords	CPC	Position	H1 Tag
Meta Descriptio n	HTML Body Text	Title Tag	iFrame Elements	PageSpeed Insights	LightHouse Audit

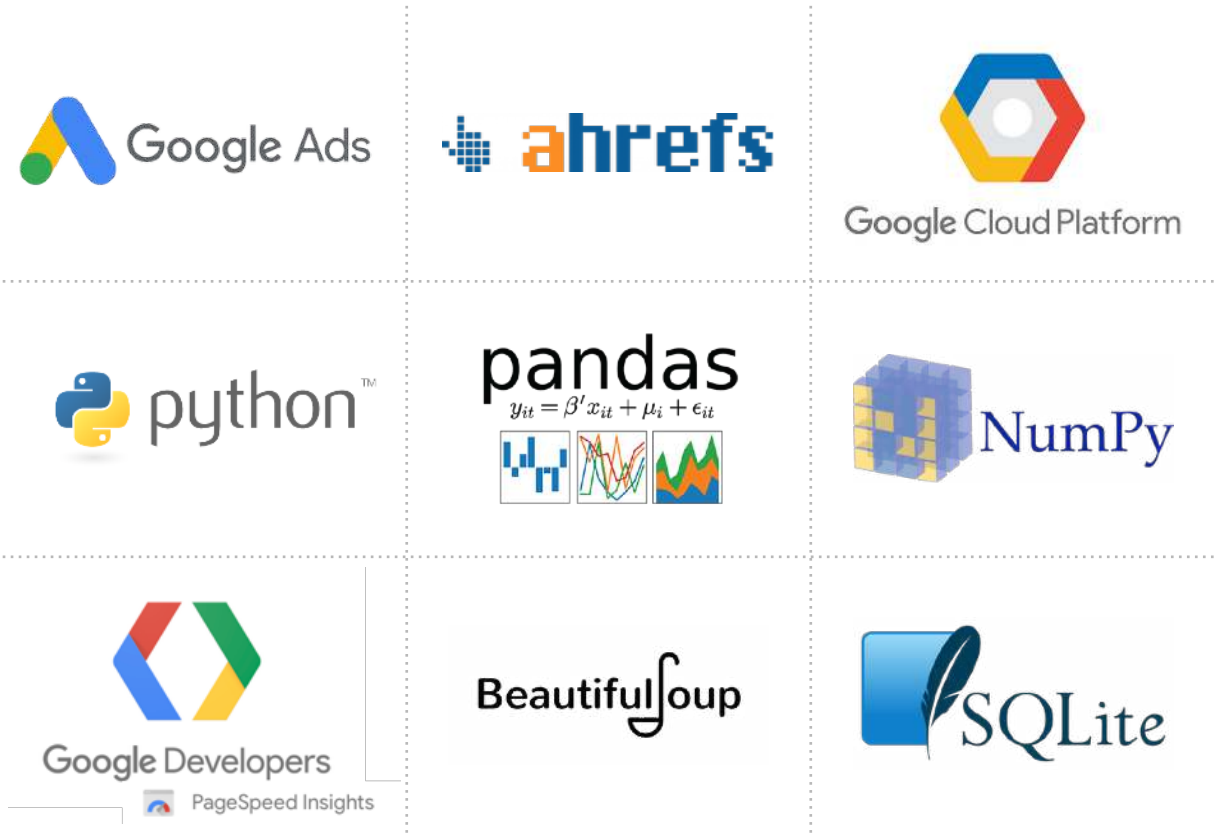
Google Keyword Planner
Ahrefs Keyword Explorer
AuthorityHacker Crawler
Google PageSpeed Insights

To calculate correlations was a pretty straightforward process. While we had a bunch of simple methods readily available (i.e. Pearson, Kendall, Spearman) using Python libraries (Scipy, Numpy, Pandas) for most analyses we only required a simple arithmetic (i.e. calculating HTTPS in URLs)

TERMINOLOGY

- **SERP:** A search engine results page (SERP) is the list of results that a search engine returns in response to a specific word or phrase query.
- **SERP Feature:** A result on a Google Search Engine Results Page (SERP) that is not a traditional result such as a link to an article or a website. Examples include rich snippets, knowledge graph, tweets, etc.
- **Keyword Search Volume (Volume):** The volume (or number) of searches for a particular keyword in one month.
- **Parent Topic:** When getting results for a particular keyword, the parent topic represents the highest volume keyword that the number one page ranks for.
- **Ahrefs Rank:** A metric developed and used by Ahrefs to order all websites from high to low based on the quality and size of their backlinks.
- **Domain Rating:** A proprietary Ahrefs' metric that shows the strength of a target website's total backlink profile (in terms of its size and quality).
- **Referring Domains:** The number of unique domains with a hyperlink pointing to a URL.
- **First Input Delay (FID):** Speed metric that represents the time from when a user first interacts with your site to the time when the browser is able to respond.
- **First Contentful Paint (FCP):** the time from navigation to the time when the browser renders the first bit of content from the DOM.

TECH STACK



- **Google Ads API:** A programmatic interface to a suite of Google Ads tools. In particular, we used it to access Google's Keyword Planner.
- **Ahrefs:** A paid software for backlinks, competitor and SEO analysis. Ahrefs offers the largest set of SEO data and runs one of the world's most active crawlers only second to Google itself.
- **PageSpeed Insights API:** An API access to Google's PageSpeed Insights which audits the content of a web page, then generates suggestions to make that page faster.

TECH STACK

- **Python + Libraries:** Python is an interpreted, high-level, general-purpose programming language. We used Python to make API calls, organize our data, build scripts to perform analyses and to build our own crawler. We took advantage of Python's many modules and libraries, including Numpy, Scipy, Pandas, Requests and Re (regular expressions module).
- **Pandas + SQLite3:** Pandas is a Python library providing high-performance data structures and data analysis tools. Â We used it in combination with SQLite3 database module for Python and Numpy.
- **Beautiful Soup:** Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML. We used it for some features of our crawler.
- **Readability.js:** A standalone version of the readability library used for Firefox Reader View. Given an HTML document, it pulls out the main body text and cleans it up. We used this python implementation.
- **Google Cloud Console:** We used the Google Compute Engine (GCP) to run our apps making API calls, as well as to crawl websites. GCP delivers virtual machines running in Google's data centers and worldwide fiber network.

THE PROCESS

Part #1: Keyword Research

This was a more challenging part than expected. We had to get at least 100,000 keywords so we can obtain at least one million SERP results.

At the same time, we needed to get keywords that made sense and aren't just plain words like *"if"*, *"and"*, *"you"* and so on, to get some statistically valid data.

First, we needed some seed words to generate keyword suggestions. Then we needed to pull out a random sample while making sure we get keywords that actually get searched for.

We aimed to ensure that at least half of all keywords are *"high-volume"* that is with minimum 1000 searches a month (as we worked with an assumption the results may be different for those).

Finally, we had to clean up the entire dataset. Here's how we did it:

- We downloaded 2500, most commonly used English nouns, verbs and adjectives and saved it to CSV file. We used popular dictionaries like Merriam-Webster to obtain them.
- Using Google Ads API, we accessed the Keyword Planner and seeded those words for suggestions. We used English for language setting and the US for location.
- Each word generated around 3,000 to 4,000. Overall we obtained around 797,936 keywords, after dropping duplicates we ended up with 573,278 keywords with different volumes.
- Finally, we cleaned it up of anything shorter than 3 characters, separated keywords with 1000 searches / month minimum volume and pulled out a random sample of 80,000 keywords from each subset, by calling the pandas .sample() method.

THE PROCESS

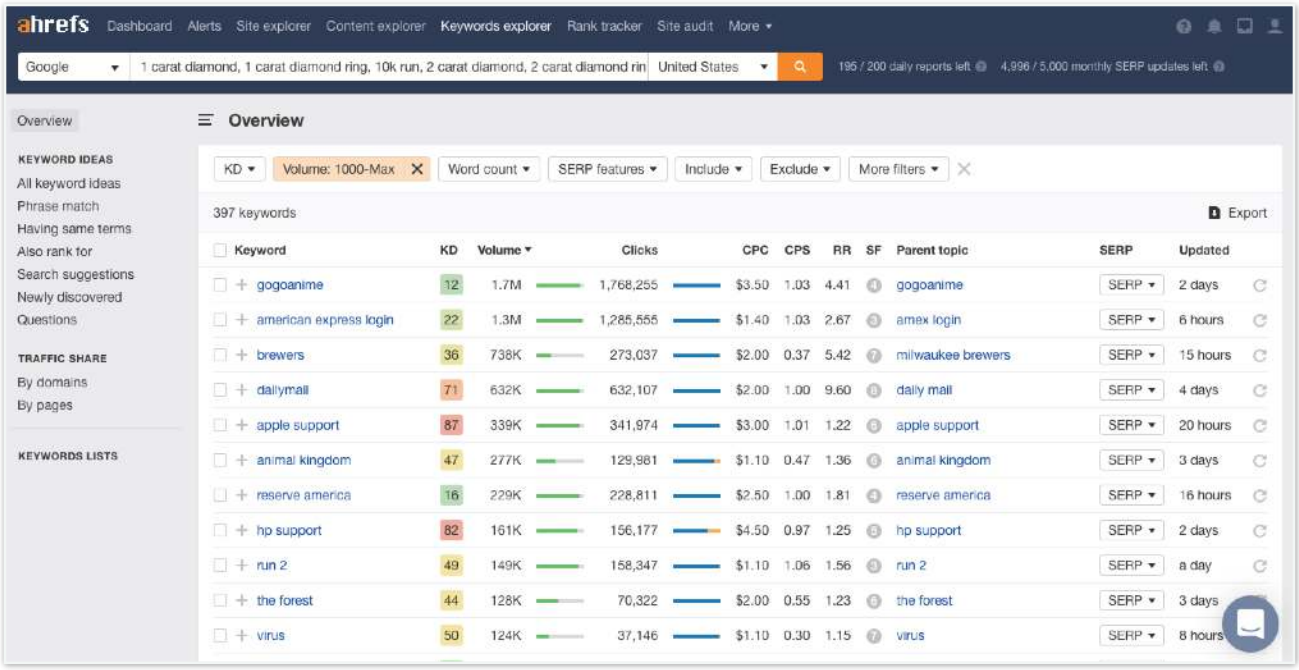
	A	B	C	D	E	F	G	H	I
3	Keyword	Currency	Avg. monthly search	Min search volume	Max search volume	Competition	Competition (indexe	Top of page bid (low	Top of page bid (hig
4	chest	EUR	N/A	10,000	100,000	High	96	0.41	1.63
5	assistance	EUR	N/A	10,000	100,000	Low	3	0.2	0.66
6	breathe	EUR	N/A	100,000	1,000,000	Low	1	3.32	9.31
7	buy	EUR	N/A	10,000	100,000	Medium	50	0.24	1.92
8	writing	EUR	N/A	100,000	1,000,000	Low	3	0.76	2.03
9	cookies	EUR	N/A	100,000	1,000,000	Medium	57	0.92	2.4
10	contribution	EUR	N/A	10,000	100,000	Low	0	0.12	1.78
11	worker	EUR	N/A	1,000	10,000	Low	5	0.15	2.35
12	chocolate	EUR	N/A	100,000	1,000,000	High	80	1.18	3.07
13	conclusion	EUR	N/A	10,000	100,000	Low	0		
14	chocolates	EUR	N/A	10,000	100,000	High	94	1.32	2.69
15	dresser	EUR	N/A	100,000	1,000,000	High	100	0.44	1.53
16	zelda breath of the w	EUR	N/A	10,000	100,000	High	76	1.78	1.78
17	dark chocolate	EUR	N/A	10,000	100,000	High	100	0.08	1.24
18	write	EUR	N/A	10,000	100,000	Low	2	0.54	1.71
19	the legend of zelda	EUR	N/A	100,000	1,000,000	Medium	59	1.78	1.78
20	best chocolate	EUR	N/A	1,000	10,000	High	100	0.8	4.18
21	white chocolate	EUR	N/A	10,000	100,000	High	100	0.29	1.89
22	chocolate covered p	EUR	N/A	10,000	100,000	High	100	0.34	1.56
23	breath of the wild	EUR	N/A	10,000	100,000	High	75	1.78	1.78
24	truffle	EUR	N/A	100,000	1,000,000	High	80	0.77	3.92
25	chocolate candy	EUR	N/A	10,000	100,000	High	100	0.84	3.4
26	chocolate maker	EUR	N/A	1,000	10,000	High	99	0.25	4.53
27	chocolate gifts	EUR	N/A	1,000	10,000	High	100	1.66	3.43
28	chocolate truffles	EUR	N/A	10,000	100,000	High	100	0.92	1.77

We took out more keywords. The reason is that we were going to seed them into Ahrefs Keyword Explorer and Ahrefs has different data than Google.

That means some of the low volume keywords wouldn't have enough Ahrefs data, and some high-volume keywords would show different volume in Ahrefs, so we wanted to make sure we get keywords that show solid volumes from both Google and Ahrefs.

For anyone looking to replicate this or conduct a similar study, please note that setting up Google Ads API access can be an incredibly painful process that requires approval too - so budget your time and resources in advance.

THE PROCESS



#2: SERP Results and Ahrefs Data

The next milestone was to obtain SERP results for each of the keywords, as well as the relevant Ahrefs metrics, such as Volume, Referring Domains and so on.

We used Ahrefs Keyword Explorer 3.0 for this. There's no API access so we had to do this part manually. Luckily the feature allows you to upload a ***.csv*** file and export up to 25,000 rows in one go (even though the actual number of exported rows were much lower each time)

As you can imagine, this took a while but eventually we managed to export 1,1 million rows with SERP data in ***.csv*** files.

We used our paid account for this. Ahrefs wasn't aware of us doing this study.

THE PROCESS

Part #3: Building Our Crawler

We wanted to collect additional metrics and data, such as H1, Meta and Title tags, wordcount and presence of some other HTML elements like images and videos.

We built a crawler using python library Beautiful Soup. That was a feasible solution to collect properties like Meta tags as it was quite reliable but insufficient grabbing the relevant content part.

The problem here, and I assume other studies faced it too, was that simply scrapping all text tends to inflate the word-count.

That's because a general library like Beautiful Soup isn't built to establish which part of the text is what's supposed to be read so it usually grabs other elements of the page too.

We needed something more accurate so we started to look for a reliable solution like the one that powers the reader view in web browsers.

After doing some manual testing and comparisons, we settled on ***Readability.js***.

Finally, we deployed our crawler with Google's Compute Engine and let it run for some time. We were able to scrape about 90% of the URLs in many concurrent processes.

THE PROCESS

Part #4: PageSpeed Insights and LightHouse Audit

In a similar fashion, we built an app to make API calls for Google PageSpeed Insights, that implements the Lighthouse Audit. We opted for the SEO audit.

This was quite a resource-consuming task, as Google sets a limit of 25,000 API calls per 24 hours. Additionally, one audit takes about 40 seconds on average.

40 seconds times million equals about 11,111 hours or 462 days.

Needless to say, with 1 million URLs it would take months to accomplish. Instead, we only pulled the data for the top and the last rank so we have can make a more simple comparison.

We broke the entire task into 50 concurrent processes and let it run until completed.

Part #5: Analysis

At this point we ended up with 3 databases. One for each stage. We used pandas to load all data into efficient data frames and performed the analysis, getting overall stats for each rank and comparing it.

For more information about this study, contact AuthorityHacker or Michal Ugor at hello@mchlgr.com