SEARCH ENGINE OPTIMIZATION TOOL WITH WEB CRAWLER, PAGE DENSITY CHECKER, SEARCH DENSITY CHECKER AND SIMILAR PAGE CHECKER

Capstone Project

Yiğitalp Körpe

İSTANBUL, 2021

MEF UNIVERSITY

SEARCH ENGINE OPTIMIZATION TOOL WITH WEB CRAWLER, PAGE DENSITY CHECKER, SEARCH DENSITY CHECKER AND SIMILAR PAGE CHECKER

Capstone Project

Yiğitalp Körpe

Advisor: Asst. Prof. Dr. Berk Gökberk

İSTANBUL, 2021

MEF UNIVERSITY

Name of the project: Search Engine Optimization Tool with Web Crawler, Page Density Checker, Search Density Checker and Similar Page Checker
Name/Last Name of the Student: Yiğitalp Körpe
Date of Project Report Submission: 14/06/2021

I hereby state that the graduation project prepared by Yiğitalp Körpe has been completed under my supervision. I accept this work as a "Graduation Project".

14/06/2021
Asst. Prof. Dr. Berk Gökberk

I hereby state that I have examined this graduation project by Yiğitalp Körpe which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

Director
of
Information Technologies
Program

We hereby state that we have held the graduation examination of _____ and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature /Date
1. Asst. Prof. Dr. Berk Gökberk	……..
2. Director's Name	……..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name Date Signature

EXECUTIVE SUMMARY

SEARCH ENGINE OPTIMIZATION TOOL WITH WEB CRAWLER, PAGE DENSITY CHECKER, SEARCH DENSITY CHECKER AND SIMILAR PAGE CHECKER

Yiğitalp Körpe

Advisor: Asst. Prof. Dr. Berk Gökberk

JUNE, 2021, 26 pages

For this project, I built an SEO tool. I am working in digital marketing, and we are using various tools, and services frequently. But due to budget constraints or a variety of tools, we cannot reach each tool all the time.

Even these tools are not easy to reach, some of their features are fundamental for jobs we are handling every day. Therefore, we needed to find their free / less expensive versions or use them within their free limits. But since I learned some coding and saw programming possibilities, I see that some must-have features are not that hard to code or complicated. Therefore, I created a small program to help my career and my budget. This script helps generally SEO reporting.

This script has 4 main features. Web crawler feature can crawl the website and provide website's page details. Page density checker feature can report the word density of the page. Search density checker searches the input query on Google, reports top 10 results and their word density. Finally similar page checker crawls the website and runs cosine similarity test for each page of the website.

Key Words: Search Engine Optimization, Web Crawler, Page Crawling, Content creation, Digital Marketing

ÖZET

İNTERNET SİTESİ TARAYICISI, SAYFA YOĞUNLUK, ARAMA YOĞUNLUK KONTROLLÜ VE BENZER SAYFA DENETLEYİCİSİ İLE ARAMA MOTORU OPTİMİZASYON ARACI

Yiğitalp Körpe

Proje Danışmanı: Dr. Öğr. Üyesi Berk Gökberk

HAZİRAN, 2021, 26 sayfa

Bu projede bir Arama Motoru optimizasyonu aracı geliştirdim. Dijital pazarlama sektöründe bir çok araç ve servis kullanıyoruz. Ancak, bütçesel ya da başka sebeplerle ihtiyacımız olan araçlara istediğimiz zaman ulaşamıyoruz.

Dijital pazarlamada kullanılan bu araçlar kolay ulaşılabilir olmasa da, bazı özellikleri yapılan iş için kritik önem taşıyor. Bu sebepten bu sektörde çalışanlar bir çok bedava ya da fiyatı daha az rakipleri tercih etmek zorunda kalıyorlar ya da bu araçları ücretsiz versiyonlarındaki limitlere kadar kullanabiliyorlar. Ancak bu programda aldığım kodlama bilgilerine dayanarak, çoğu kritik özelliğin aslında programlama olarak çok zor olmayacağını ya da algoritmik olarak çok karışık olmadığını gördüm. Bu sebepten bu işlemleri yapabileceğim küçük bir program geliştirdim.

Bu program 4 ana özellikten oluşuyor. İnternet sitesi tarayıcısı özelliği internet sitesindeki sayfaları inceleyip o sayfalar hakkında bilgileri raporluyor. Sayfa yoğunluk analizi özelliği sayfadaki içeriği inceleyip kelime yoğunluklarını raporluyor. Arama yoğunluk analizi özelliği arama motorunda istenilen kelimeyi aratıp ilk sıralarda çıkan sonuçlardaki anahtar kelimeleri analiz ediyor. Son olarak, benzer sayfa analizi özelliği internet sitesindeki benzer sayfaları bulmak için bütün sayfalar için ikili cosine similarity testi uyguluyor.

Anahtar Kelimeler: Arama Motoru Optimizasyonu, İnternet sitesi tarayıcısı, Sayfa tarayıcısı, içerik yaratımı, dijital pazarlama

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

In this paper, I will detail my Capstone project which aims to help and boost a Digital Marketer's daily basis tasks. I am a digital marketer as well and we are using various tools, and services frequently. But due to budget constraints or a variety of tools, we cannot reach each tool all the time. Also, they are so expensive for personal use.

Even these tools are not easy to reach, some of their features are fundamental for jobs we are handling every day. Therefore, we needed to find their free / less expensive versions or use them within their free limits. But since I learned some coding and saw programming possibilities, I see that some must-have features are not that hard to code or complicated. Therefore, I would like to create a small program to help my career and my budget.

There would be 4 main features in this project:

1- Website Crawling: With this feature user can report what is the content of the website and see the similarities and differences for each page. This feature will report meta information such as title, description and keyword. It will also report its titles which have hierarchy like the biggest title named as H1 and the second one named as H2, and it continues like this until H6. For this project, since we would like to follow main usage, the script will report just H1 to H4 which will cover the 90%. This feature also reports the HTML responds from the server and the page so we can detect the problems and issues with this code.

2- On-Page Audit: In this paper and on the script, we will call it "Page Density Checker". This feature will get the content of the requested page, normalize the content and calculates the density of words in the content.

3- Search Density Checker: With this feature, content creators can find keywords related to their topic and find competitors for that search query. Within this information, content creator can decide wording of the content and also decide not to pursue this topic or find perfect gap with their content and pushes the content further with distribution channels.

4- Similar Page: Content creation process can be long and can run by different parties such as vendors. This can cause similar or almost duplicated contents. With this feature, content creators and digital marketers can find their pages, similar information to website crawler such as title, HTML respond and meta

information but, in this case, user can see the keywords of the pages and also the cosine similarity values of the pages. With this cosine similarity users can find same or almost same pages and update them accordingly.

2. METHODS

2.1. Overview

There are 4 main parts of project. Since I aimed what I am using for Search Engine optimization in my daily life, all of these functions' goal is helping my daily work and save time. The main 4 functions are:

- Website Crawler: With this function, you can get desired website's pages' information. This information are page's HTML statuses such as 200 for healthy pages and 301 or 302 for redirections and 404 for broken pages, their meta information such as meta title, meta keywords and meta description, their page HTML title, and on page titles such as H1, H2, and H3.
- Page Density Checker: This function requests a single page from user with the page's language and how many keywords user would like to see. With these information script crawls the single page and gets its content information. The script removes language most used meaningless words, punctuations, after with stemming functions, script reaches the stem of the smallest meaningful part of the word, and this helps us to analyze the content. For analysis, the script runs density function and counts word's repetition. Since search engines understand the page's topic and content by their words and sentences, this helps us to see from Google's eyes. For reporting the program shows the most used keywords according to users input number.
- Search Density Checker: While content creating process, you need to check what is the most relevant words for your topic. By understanding this you can check Google results and see what people wrote about this topic and also, what Google thinks the most relevant words about this topic. For this proposes, this script requests the topic of the new content, number of the page the user would like to analyze and what language user would like to make this research. Within these inputs, script thinks the topic is a search query for Google and makes the search for the user. With input of number of the pages user would like to check, gets the top 10 or 5 or 20 results according to user's input and crawl those pages. Script runs page density analysis on those pages and return to user what the user should mention on their content.

- Similarity Checker: This function is the key for big old websites. While content creation process content creators focus on the topic they will talk about and generally they do not make research for the content they have already have. This causes cannibalization on content and none of content works how it should be. Therefore, this function requests a domain from user and crawls the website and gets the URLs of the pages. Script gets the most used keywords on the page and compare them each other. With Cosine Similarity, script creates a similarity value for each two pages. Within this similarity value, it creates a report with the most similar ones.

2.2. Functions

As I mentioned under 2.1 Projects Parts, there are 4 main parts/functions in this project. All of these functions, mainly aim make daily life easier and easily understandable. For this goal, I tried to make user interaction as less as I can. For example, website crawler just requests website URL. After that point, even if the script faces an error, it does not request user input. It shows when it happens and continues its job.

2.2.1. Object Oriented and Classes

For keeping pages and websites as an object, I have created 2 classes – Page and Website. If we need to talk hierarchically website object covers page objects. Website object holds domains URL which is "www.amazon.com.tr" for Amazon Turkey operations. Website object also holds Page URLs and page count. Page URLs is a list of Page objects. Further in this paper, I will mention it again but for small introduction, each time script finds a new page under the website, it adds it to the website objects page URLs list. By this way, I can reach all pages under the domain with the website object. Page count attribute is for tracking the pages and the size of the domain. I have created a class function as well to add new page under the website and adds one to the website page counter.

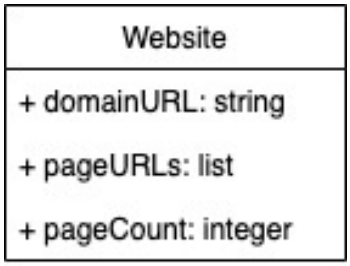

Figure 1: Website object UML diagram

For page objects, I would like to keep all page related information in the one page object. By this way, when the page once crawled and placed them right object attributes, further in the script there is no need to send a request to website and crawl them again. With this approach, script just crawl one page just once. In page objects there are 8 attributes. The first one and the most used one is the URL of the page. In the script, I am keeping it as the 'link' of the page. But since I do not want to lose the source of the page, the script keeps the source of the page as previous link. All requests to the websites have a HTML respond in the header (see the figure).

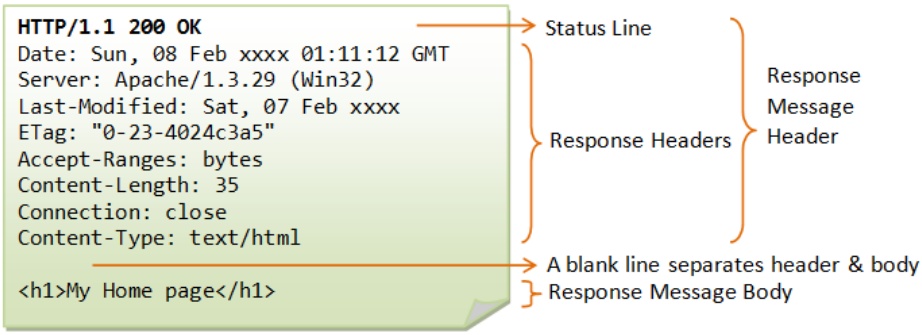

Figure 2: A sample HTTP request massage [1]

The most known one is 404 which means the page is no longer exist or removed. But 404 is not the only one. They started from 100 to 590 and each hundred has different meanings. But the most used ones are 2XX Success, 3XX Redirection, 4XX Client Errors and 5XX Server Errors. At this stage I would like to detail just few of them. 200 means OK which is the page, and the server works properly. In this case, the content of the page is accessible, and the script only crawl the page only in this situation. 301 and 302 really similar ones. Also, most of the cases users do not notice they got 3XX response. 301 and 302 means the page which the web client requested has been moved. 301 means it moved permanently, 302 means the page moved temporarily. One of the reasons we are using 2 codes for one

simple redirection is to tell search engines and crawlers that this page will be back or not. If we moved the page permanently, we expect to search engines remove the page from their indexing. To summarize, HTML response code is an important respond from the pages and the script's action will vary. Therefore, the script keeps this information within the objects.

Since each time requesting the HTML status code of the page, I am having the body (content) of the page as well. In the program, the script needs unfiltered and filtered ones at some stages. For this reason, I keep the unfiltered one as 'raw content'. For filtered content, script parse the raw content to major and important parts. For this purpose, I am using Beautiful Soup and get title, meta title, description, H1, H2, H3 and paragraph tags such as p tag for HTML. After decomposing all elements adding all these HTML elements with the key to the element object to the Page object content dictionary. For each page, script collects all crawlable links such as for HTML under 'a' tag collects href links and adds to page object's links list. Since for some pages script will run page keyword density algorithm, since I would like to reach this information from the object, I chose to implement them as keyword attribute to the Page class.

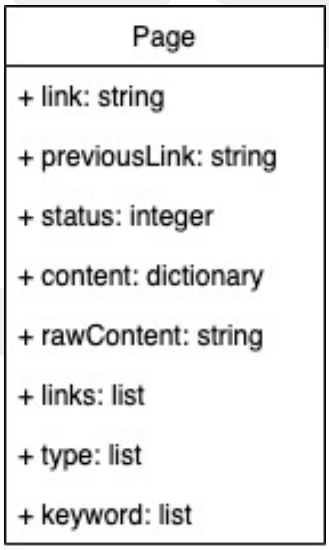

Figure 3: Page object UML diagram

2.2.2. Website Crawler

As I mentioned before, all of functions in this project based on daily needs. In a digital marketers' daily life, most of the time we are looking for new tools or spending time on tools to see what they are capable. Mainly this function based on budget limits for companies. While working on a SEO project, first move always try to learn what are the own assets of the company. We check client's website and text manually. We are looking for a good way to understand their current position so we can find where to improve or finding gaps to fill. Therefore, we need a tool to crawl the clients' website and report at least main structure such as headlines, titles, subtitles, meta keywords that website owner has added to their website. All of this information leads us to understand the logic and the strategy behind the brand.

Therefore, fundamental website information about the website is the key for marketers. We are using different paid tools for these jobs. Generally, website crawling is not their main or only job, but they also provide these details about the website.

For example, SEMrush [2] is a good, well-known, advance SEO tool and it can help us to find content keywords or traffic estimations. Not just for anyone on the internet. Therefore, you can use this tool for exploring or detailing your competitors as well. For doing that you can command the tool to crawl your website. One of its magic touch is provides some errors or warnings about the website. If you have bad-broken pages, it reports them and where to reach it. Or if you have bad user interface to crawl, it reports and gives you a site score accordingly. They keep adding new checks, they have added even the latest update of the Google these days. I am writing this document on May 31, 2021, and on their website [3] they are stated that they have more than 120 on-page SEO checklist. But according to your package, there is a limit to number of pages to crawl. If you are free user, you can crawl just 100 pages per month. In this day, there are 4 packages and monthly limit goes up to 1 million number of pages to crawl. As they stated obviously free user is free, but 100 pages is not enough even for smallest websites. Therefore, you need to buy one of their packages and they are starting from the 120 united states dollars to custom solutions. Their smallest package lets you to crawl 100,000 number of pages which is not too small but not big. If consider we need to check site heaths at least weekly, you may need to update your package.

One another good example is Screaming Frog [4]. If we compare Screaming Frog and SEMrush, we can easily say that SEMrush is for keyword and content analysis and Screaming Frog is for website audit. Screaming Frog provides details such as broken links, internal and external linking and other on-page SEO results. As I mentioned before, SEMrush provides more keyword insights or keyword trends. Yes, it provides website page related results as well, but it is not its strongest asset. Therefore, for on-page search engine optimizations, marketers generally [5] prefer Screaming Frog. As you can notice just for on-page SEO, we start to have 2 totally different market leaders. We are not going to detail their alternatives in this paper but as you can imagine there are lots of alternatives for each of them.

Screaming Frog is a software, so you need to download and install it to your mac, windows or ubuntu computer. By this way it uses our computer and network for site crawling which is a good think and a bad think at the same time. Results and the process depends on your network and computer stability. If the link on the one of the pages, block to crawlers or your location, the results say this page is not reachable. But it could be reachable for other countries or locations. The opposite could happen as well, like SEMrush is a SaaS and provides results with its' servers, so the results could be variable and there is no right answer. You need to choose which tool you are going to use with your current status.

Screaming Frog lets users see a lot of on-page information. [6] Like analyzing title and descriptions of the page or Audit redirects. But just like SEMrush, of course they are not giving this program for free. At least all of it. You can crawl up to 500 pages which free version of the tool. If you have a bigger website, you need to buy pro version which is 149 pounds for a year [7]. Also, some good features such as ignore robots.txt or JavaScript website rendering require pro version.

With all these examples, I was thinking to create a new website crawler to solve my daily jobs and created a simple website structure report creator. Since I would like to limit the user interaction with the program, this function of the project only requests website URL and keeps asking the website for the correct format. After getting the first URL correct, the script purse it to find website's domain. From now, there is a loop with 2 lists - crawled pages and to be crawled pages. Even for the first URL which entered by the user, script adds the URLs to the to be crawled URLs list. The loop will be finished when all to be crawled list element has been crawled and processed successfully.

The script takes each element of the list and starts the process. For the Web Crawling the process starts from getting the HTML response from the web servers. If the URL is a valid URL which means if it is a web URL, script gets the status code and the content of the page. After getting all of this information, script checks the page HTML status if it is available page content. Because for example if the page returns 404 or 502, the page content becomes inaccessible, and the script cannot complete further steps. Therefore, there is a control in this point for page HTML respond. In this step, script also controls for the domain and checks if the page is already crawled or not. Since at this stage, I do not want to crawl all world wide web, the code parses the URL and checks if its same with the starting page. If current page domain is not same with the starter one, the script eliminates the current page and does not let it go further process. It is same with the previously crawled pages. As I mentioned before, there is a list of objects for crawled pages. And further in these steps we will see that the script adds the crawled page objects to this crawled page list. But even if this case, I do not want to crawl pages more than one. Therefore, there is one another control in this stage as well. So, if current page is in the crawled page list, the control eliminates the page as well.

Since the script made all required controls for this step, it passes the page to the next steps. Since we have crawled the page beforehand the control, we got the raw HTML response of the page as a page raw content attribute. For further analyses we need to parse the content and prepare it. For this reason, with beautiful soup, I am creating a title value with the title value of the page. For all title, description, meta keywords, h1, h2, and h3 elements selecting with parsing and creating variable steps continue. With all these information, the script appends this information to the page content dictionary in the order with the HTML terminology as the key of the dictionary values. For example, As I got the title information of the page and created the title value with it, the script appends this information to the page object's content dictionary as title key and title variable value as value of the key. These steps continue for each determined HTML tag and by this way the script fills the content information of the page object.

Since the script got the content of the page, it adds the current page object to the crawled page list and for finding upcoming URL, checks the raw data for the 'a' HTML tag and their href information. For next URLs it needs to fix - normalize them.

For normalizing the founded URLs, the script checks following rules:

- The new URL must not be the domain or the first provided URL to the script. Since all regular websites provides a link to their website homepage link, before giving any loop I would like to remove them.
- Some internal linking contains mail option or JavaScript functions. Therefore, if the link the script got from the href of the page contains any JavaScript tag or mail or telephone command, script eliminates the page.
- Some web developers can provide internal link without the domain of the website. Since the basic server systems based on the folders, it can direct to the page directly. For example: some websites use internal linking as '/shop' instead of 'www.domain.com/shop'. In this case, the script needs to add the domain and the https information.
- Since I would like to reach new pages and crawl HTML pages of the website, if the URL links to any kind of image the script removes them such as jpeg, jpg, png, gif.
- The page should not be in the to be crawled or crawled page lists.

If found URLs follow these rules, the script appends the URLs to the to be crawled page list and the loop continues until the script crawl all pages in the to be crawled list.

At the end of the loop, the script exports the crawled pages as a csv for reporting and analysing. This report contains the URL, status, previous page, meta title, meta description, keyword, and HTML titles (H1, H2, and H3) of the pages.

2.2.3. Page Density Checker

Page word density just like website crawler is one of the main check points for digital marketers. While creating a new content or auditing old content of the page, marketers need to check the main topic, and most used words of the page since it is one of the magic functions of search engines. Sometimes even if the topic of the page is let's say table lamps, but on the page, you keep mentioning bulbs. It creates conflicts and hard to understand for search engines. By this mistake your lamp page starts to get position on bulb results (search queries) but since the user searched for a bulb and does not want to see your lamp page. Therefore, they do not click to your page or even if they clicked on the page, they are leaving

it quickly as they can. This gives bad signs to the search engines about your page, and they push your page deep internet dumpsters.

For avoiding these, digital marketers keep an eye on search queries that the page is performing but event in this stage it could be a late respond. Because search engines started to test your page for those queries and the results started to affect your page and domain scores. Also, digital marketers need to keep an eye on new contents. Their new content's topic could be about lamb but if it mentions bulb more than lamb search engines will think it is related to bulbs. You can easily avoid this kind of word density errors just with these crawlers.

Just like the main idea of building these scripts, I would like to minimize the user interaction with the program. Therefore, Program requires just the URL of the page you would like to crawl, language of the page and how many keywords you would like to see as a result. Language is important at this stage since we would like to see common words, we would like to eliminate most used meaningless words for our experiment such as 'the', 'I'.

When the script got the URL from the user input, it checks if it is valid or not. Valid URL means accessible at this stage of the script. With the valid URL, script gets HTML titles (H1, H2, H3 and H4) and paragraphs with p HTML tags. With getting all of these data from the page and appends them to the 'dataset' of the crawled page. Since we got the data from just a page from the world wide web, and since the page is not designed for running these tests on the page, we need to format and parse the texts we got. In the script I am calling this stage text normalizing and this function includes:

- Remove punctions: since we got the texts from a page which designed for reading for humans, it does not design for running these tests. Therefore, the script needs to remove punctions from our dataset. For this function, script checks each text in the dataset and replace punctuations with blank string. Therefore, if there is apostrophe, it does not add and mixed words.
- Sentence to words: We will not check the meaning of the sentences. We would like to check most used words in the page. Therefore, we can simply divide sentences, paragraphs to words and run our tests on them.
- Remove HTML Characters: For HTML pages, there are some HTML tag to change the appearance of the text. For example, with <bold> tag you can make the text bold, or you can create and edit its appearance with the

CSS codes. But since we would like to analyse just readable part and script removes HTML tags from the dataset.

- Lower Case: With natural languages sentence cases and capital letters make different. But for our script, we do not want to any case sensitivity, the script makes each word lower letters. By this way, the script avoids case sensitivity errors.

After this text normalizing stage, the script removes unwanted words, and the dataset is ready for stemming. Since we would like to reach the stem of the words and avoid plural or other forms, I am using an external library called Porter Stemming. I simply send the word to the function, and it returns the stem of the word. I am replacing the stem with the first word. For example, I do not want to see lamp and lamps in the output of the keyword analysis. At the same time, it does affect the results as well. Let's say if there are 4 lamp and 6 lamps on the page. Search engines understand the word and connect them with their stem. By this way the total lamp should be 10 but the program shows them separately. For fixing this I have used stemming. After text normalizing and stemming, word pool (dataset) is ready for analysis. With the counter function, all list of keywords is ready to list the results and returning via python console.

2.2.4. Search Density Checker

Search Density checker, just like other functions it aims to solve daily life problem of digital marketers. For search engine optimizations we cannot let content creators free. While working on a new page, blog post or any content for the website or related to search engines, content writers need to follow common sense and answer common questions. Therefore, they need to follow keywords and research related to the topic. For example, if a content creator creates a new blog post regarding to new updates on Apple and Apple's IDFA changes. Content creator needs to check other websites, and other common search queries as well. And also, they need to adjust the text accordingly. It will also affect other steps as well. If general search query for id for advertisers is IDFA, the content creator needs to place IDFA keyword in the content.

For finding all these keywords, content creators check popular tools. I have mention some of them previously such as SEMrush or Ahrefs. But this could be not enough, and

creator would like to check deeper. In this case, they will write the query to one of search engines such as Google, and they will check first few pages. Creators will check page titles, text and most used words.

With this function of the script, I would like to help content creators first Google Search. As all other function, I would like to keep the user interaction minimum. Therefore, the script asks just for the query. This could be the topic or the main keyword of the content they are working on, how many results should the script check, which language they would like to do the search or the content, and how many keywords they would like to see as a result.

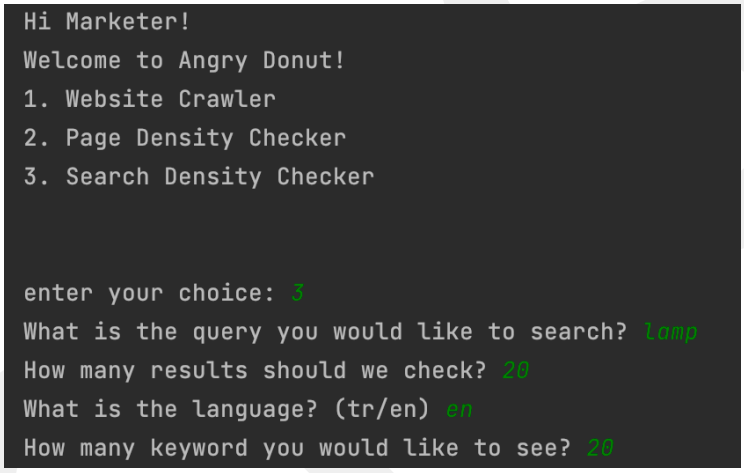

Figure 4: Screenshot from Search Density checker

After these inputs, script checks the Google with requested query and gets top requested page counts and show them to the user. Even if you select English for your search, Google's results will vary according to your search location. For example, right now I am located in Izmir and as I have added a screenshot below, it shows Ikea's Turkish lamp page for the first result.

Figure 5: Search density Google results output

Since we would like to create a content for this topic and these results perform best for this query. We can investigate what they are doing good. For this, the script will crawl these pages and report their word density.

I am using similar crawling techniques for these pages crawling as well. At first place, the script gets their HTML titles (h1, h2, h3, h4) and paragraphs information as shown p tag for HTML. The script creates a word pool with this information and removes punctions, separates the words, removes HTML characters, and make them lower case. After normalizing the texts, they are ready for analysis and reporting their frequency. The script prints the results like shown below.

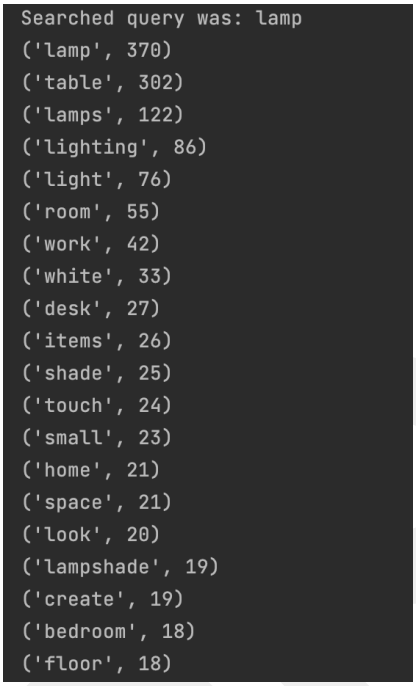

Figure 6: Search Density keyword results

As we can see there is a strong relation between lamp and table words. This could make content creator think pursue the connection between these two words. The connection may be there is a model of lamp, we call it table lamp for work desks. We are also seeing that lamp word seems related to the “work” word as well. This could be related to the table lamb. Starting from this point the content creation could decide a new section to their content. As a digital marketer, I strongly recommend mention table lambs and types of lamps such as tabletop, desk, room, work, bedroom, floor. And I have reached all of this knowledge without making any additional search, we can see these right in the results of the script. For better visualization, I have created a word cloud with the results.

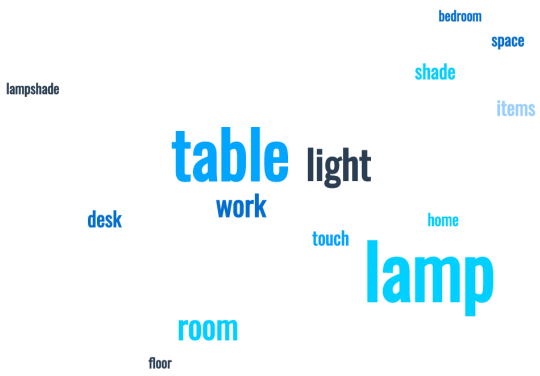

Figure 7: Search Density “lamp” query script output as a word cloud

2.2.5. Similar Page Checker

Other features generally focused on new content creation. Page Density helps us to understand the keywords of the pages, page density let user find successful pages and their keyword analysis. But similar page checker unlike others, focuses on old pages and the whole website. For websites, copying same pages and having similar page results kills and cannibalizes the website’s search engine performance. Since the search engines need to order pages with their success, not well performer pages will affect good ones and search engines thinks the website is try to fraud and let the website lower positions. For avoiding these simple errors, pages of the website need to be creative and unique. Even for internal linking. Content creators need to handle different topic on each page and should not let them create conflict each other. For making this content creator can check the sitemap, make a research for the page. But generally, these kinds of jobs outsource by the companies and each content provider creates similar pages. And none of highly functional SEO tools show similar pages. Therefore, I have created a script to crawl the website, finds all internal linking, visits them, creates keyword density report with their content and gives them similarity value. With all URLs and similarity values creates a CSV document as an export.

Just like any other feature, this feature is also aiming for minimum user interaction. Therefore, it just requests the domain or one page of the website. With that URL, script parses the domain and starts the crawl.

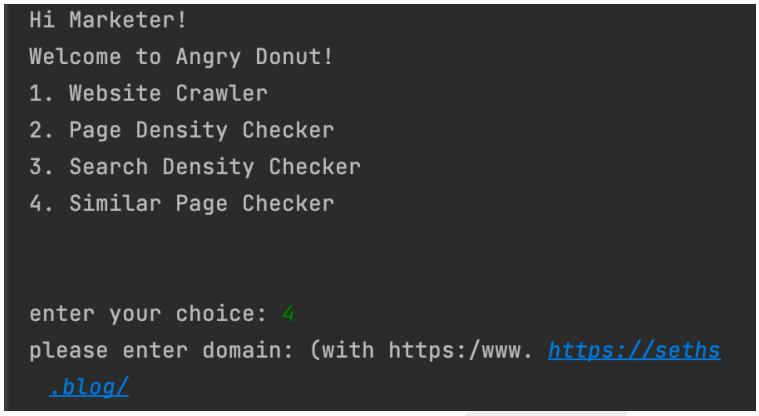

Figure 8: Similar Page user input

The script visits the page and controls the HTML respond code of the page. If the code is available for crawling, it gets the content and extract internal linking. And each page, runs the page density algorithm and gets the word density of page. With that word density results, the script implements the keywords of the page as page attribution – page's keyword. The script appends each crawled page to the crawled page list as a page object. By this way, the script could reach the object and its keyword attribution easily.

The script finishes the loop when it reaches all possible internal URL it could reach. This function is almost same with the web crawling function except keyword analysis for each page and implement these keywords to the page object. After these steps, the function exports the crawling results just like it does website crawling but for this function page objects have keyword density information as well. So, the output CSV file contains "URL", "Status", "Previous Page", "Title", "Description", "keyword", "H1", "H2", "H3", and "Keywords" variables for each crawled page.

After providing keywords and page URLs, the script analysis the pages and keywords for similarity. For this purpose, I am using cosine similarity. Cosine similarity simply creates vectors for words and compare the similarity of the pages. The function returns a number between 0-1. If cosine similarity value close to 1, that means pages are similar to each other. If the cosine similarity value close to 0, they are they do not have similarity.

Since with the cosine similarity, the script needs to compare the page with each other. This causes massive algorithm complexity problem. For example, we are working with a small website which contains just 10 pages. For each page, the page needs to be compared to other pages. Homepage needs to be compared with other 9 pages, the second one on the list needs to be compared with the other 8 ones. It causes factorial algorithm complexity and

if we consider that we will work with big websites, such as websites contains 1000 pages or more. For making this complexity easier, I tried to avoid double works and keep the code simple as possible. Therefore, I used pandas dataframe for flexibility and handling. Each time the script calculates the cosine similarity, it adds URLs and the cosine similarity value to the dataframe. After finishing all work, it exports as a CSV file with the URLs and the cosine similarity.

URL1	URL2	Cosine Similarity
https://www.tarihikisiler.com/	https://www.tarihikisiler.com	1.0
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kategori/dergide-bu-hafta	0.537379576852221
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kategori/kitap-yazilari	0.5227041235019606
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kategori/tarihi-olaylar	0.42079859191446645
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kategori/haftanin-karakteri	0.41096893941382656
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kategori/roportajlar	0.48262942543047715
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kategori/mekanlar-ve-hikayeleri	0.4454956024141497
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/siz-de-yazinizi-gonderin	0.2067714499159616
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kitap-yazilari/turk-alman-iliskilerine-kisa-bir-bakis-fahrettin-kerim-gokay	0.46130753195243007
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/haftanin-karakteri/adnan-menderes	0.4451218069383224
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/dergide-bu-hafta/fezaya-bir-adam-firlatildi	0.44457390590813156
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/mekanlar-ve-hikayeleri/kulturlerin-bulusma-noktasi-moda	0.4148747644711034
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/roportajlar/altemur-kilic	0.4558559341335495
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/kitap-yazilari/bir-sehzadenin-hatirati-sehzade-ali-vasib-efendi	0.49139444157966183
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/mekanlar-ve-hikayeleri/istanbulun-sirin-koyu-cengelkoy	0.4216538832740323
https://www.tarihikisiler.com/	https://www.tarihikisiler.com/ozel-gun/28-mayis-hamburger-gunu	0.4300911078154467

Figure 9: Similar Page Cosine Similarity report

3. RESULTS

3.1. Overview

For this project, my aim was creating something can be useful for digital marketers and help their daily basis jobs. I could not find the possibility use these functions in my daily life but when I discuss this script's capabilities with my coworkers, and they would like to try and see the results as well.

3.2. Website Crawler

For results, I would like to select some of the most visited websites. Since we will not make deep dive analysis for these websites, I have added limitation for the crawler and when it reaches 500. Iteration it stops and returns the results.

For first example, I choose "https://www.fandom.com/" which is the 13. of the most visited websites in the US. [8] For starting the code requests the URL of the page.

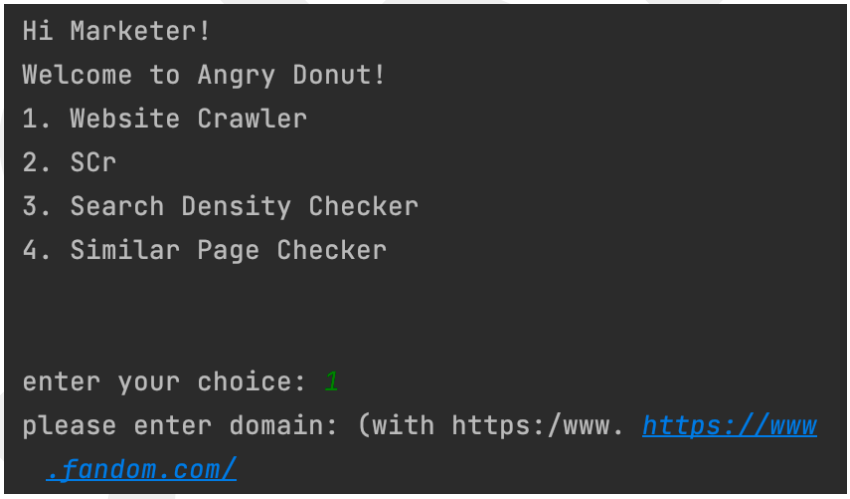

Figure 10: Website Crawler user interface

One thing I have noticed, after crawling the homepage, the script found 245 pages to crawl which means the website has solid internal linking and you can reach pages easily. After iteration 10, the script found 713 pages which is really big number and explains why this website is one of the most visited websites. When the script reaches the 500-crawl limit, I have placed for this test, the length of to be crawled list was 6783 pages to crawl. But the

crawled page count is 250 which means half of the iteration got the same URL, and eliminated the duplication.

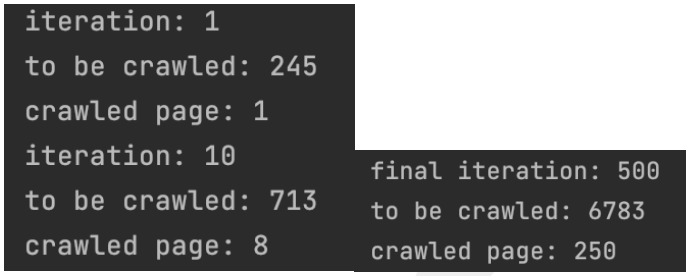

Figure 11: Website crawler Fandom.com 1., 10. And 500. Iteration results

When we checked the output of the script, the first we can see the URLs it crawled, Status code, previous page URL and the title of the page. Since the first URL was the starting URL the Previous Page field is empty.

URL	Status	Previous Page	Title
https://www.fandom.com/	200		Fandom
https://www.fandom.com/topics/games	200	https://www.fandom.com/	Games | Fandom
https://www.fandom.com/topics/movies	200	https://www.fandom.com/	Movies | Fandom
https://www.fandom.com/topics/tv	200	https://www.fandom.com/	TV | Fandom
https://www.fandom.com/video	200	https://www.fandom.com/	Videos | Fandom
https://www.fandom.com/explore	200	https://www.fandom.com/	Explore | FANDOM
https://www.fandom.com/signin	200	https://www.fandom.com/	Sign in | Fandom
https://www.fandom.com/register	200	https://www.fandom.com/	Join FANDOM Today | Fandom

Figure 12: Fandom.com web crawler URL, Status, Previous page and title results

When we check other columns of the pages, such as Description, keyword, H1, H2 and H3. They are using same description for their main pages, and they do not use meta keywords which is the recommended option for meta keywords. But when we check H1, H2 and H3. They are not using these title hierarchy efficiently.

Description	keyword	H1	H2	H3
The entertainment site where fans come first. Your daily source for all things TV, movies, and games, including Star Wars, Fallout, Marvel, DC and more.			[']	['The God of Mischief Who Would be King: The Psychology of Loki', 'The Evolution of 'Far Cry'', 'The Best Detective G
The entertainment site where fans come first. Your daily source for all things TV, movies, and games, including Star Wars, Fallout, Mc		['Games', 'Games', 'Games']	[']	['The God of Mischief Who Would be King: The Psychology of Loki', 'The Evolution of 'Far Cry'', 'The Best Detective G
The entertainment site where fans come first. Your daily source for all things TV, movies, and games, including Star Wars, Fallout, Mc		['Movies', 'Movies', 'Movies']	[']	['The God of Mischief Who Would be King: The Psychology of Loki', 'The Evolution of 'Far Cry'', 'The Best Detective G
The entertainment site where fans come first. Your daily source for all things TV, movies, and games, including Star Wars, Fallout, Mc		['TV', 'TV', 'TV']	[']	['The God of Mischief Who Would be King: The Psychology of Loki', 'The Evolution of 'Far Cry'', 'The Best Detective G
The entertainment site where fans come first. Your daily source for all things TV, movies, and games, including Star Wars, Fallout, Marvel, DC and more.			[']	['Recommended on Fandom', 'Honest Trailers', 'The Loop', 'Fandom 5 Trivia', 'Hey Fandom!', 'Reviews', 'Interviews', '
			['FANDOM is the fan's voice in entertainm	['Loki', 'Ratchet & Clank: Rift Apart', 'Record of Ragnarok', 'King's Bounty IIWhat You Need to know', 'Alex Kidd in Mir

Figure 13: Fandom.com web crawler Description, Keyword, H1, H2 and H3 results

3.3. Page Density Checker

For page Density Checker, I would like to find a different example and good example. Therefore, I decided to select one of the most popular Search Engine Optimization blog page. As the following screenshot, I have entered the page URL, the language of the page and how many keywords I would like to see for this paper.

```
Hi Marketer!
Welcome to Angry Donut!
1. Website Crawler
2. Page Density Checker
3. Search Density Checker
4. Similar Page Checker


enter your choice: 2
please enter a page URL: https://searchengineland
  .com/google-announces-its-own-version-of-app
  -tracking-transparency-349272
Please enter the language of the page (tr/en): en
How many keywords would you like to see? 10
```

Figure 14: Page Density Checker user input screen

Within few seconds, the script crawled and run its algorithms and returned the results. When I checked the page, the content is about the Google's announcement regarding Advertising ID. For understanding the results, basically Google announce that they will follow Apple's lead for adverting ids and app privacy. According to this information, what we see as keyword density results make sense.

```
URL was:  https://searchengineland
  .com/google-announces-its-own-version-of-app
  -tracking-transparency-349272
keywords for this page:
('user', 13)
('googl', 12)
('advertis', 10)
('app', 9)
('2021', 9)
('smx', 8)
('privaci', 7)
('help', 6)
('track', 5)
('opt', 5)
```

Figure 15: Page Density Checker Search Engine webpage keyword result

3.4. Search Density Checker

For this research, we can continue our previous example and continue our research about IDFA. For making this search, I needed to input our research keyword “IDFA” as search query, 10 top results can help us to understand the main topic, language will be English, and 10 keywords can sum the functionality of the script.

```
Hi Marketer!
Welcome to Angry Donut!
1. Website Crawler
2. Page Density Checker
3. Search Density Checker
4. Similar Page Checker


enter your choice: 3
What is the query you would like to search? idfa
How many results should we check? 10
What is the language? (tr/en) en
How many keyword you would like to see? 10
```

Figure 16: Search Density Checker user input screen

Within seconds, the script returns the 10 results for our query and provides keyword density results. As far as I see, MMPs run good strategy and placed ahead of Apple. And most of keywords related to user, data, apple which is expected if we think the topic.

```
These are the top results for:  idfa
['https://www.adjust.com/glossary/idfa/',
 'https://www.idfa.org/', 'https://www.appsflyer
 .com/mobile-attribution-glossary/idfa/',
 'https://www.idfa.nl/', 'https://www.invoca
 .com/blog/what-is-idfa-and-why-apple-killed-it
 -everything-marketers-need-to-know',
 'https://tinuiti.com/blog/data-privacy/apple-ios
 -idfa-guide/', 'https://en.wikipedia
 .org/wiki/Identifier_for_Advertisers',
 'https://developer.apple
 .com/app-store/user-privacy-and-data-use/',
 'https://developer.apple
 .com/documentation/appstoreconnectapi
 /advertising_identifier_idfa_declarations']
Searched query was: idfa
('idfa', 102)
('user', 85)
('app', 79)
('apple', 69)
('data', 67)
('users', 64)
('tracking', 63)
('ios', 56)
('device', 43)
('ad', 42)
```

Figure 17: Search Density Checker results for “idfa” query

3.5. Similar Page Checker

Similar Page Checker is the trickiest one, since we consider the algorithm complexity. The script needs to crawl the website and compare them with each other. For this paper, since we would like to see the functionality, I have added a 50-limit and run the code for “https://www.macrumors.com/” website which is well known source for Apple news.

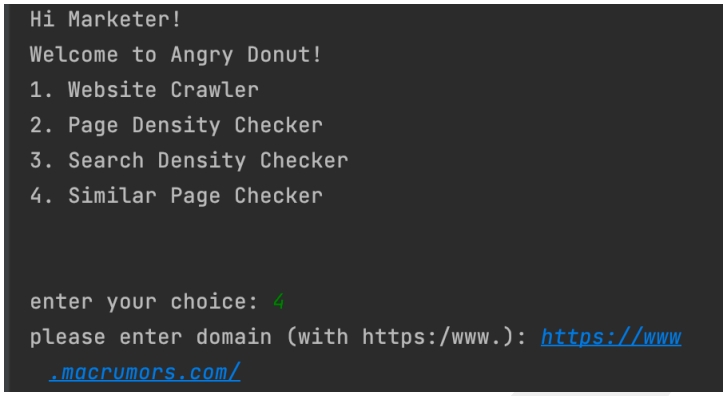

Figure 18: Similar Page Checker user input screen

Their website was more impressive than I had expected. After first crawling the script found 343 pages to crawl, after 10th iteration, the number was 845. This means they have a good page quality and internal linking. When the script reaches the 50-limit placed by myself, the script found 2127 pages to crawl and 46 pages successfully crawled which means only 3 of iteration end up with elimination which is amazingly high.

If we check cosine similarity numbers for the Macrumors.com, there is a big similarity between pro and SE models' pages such as watch SE and watch or iPhone 12 and iPhone 12 Pro.

URL1	URL2	Cosine Similarity
https://www.macrumors.com/roundup/apple-watch-se	https://www.macrumors.com/roundup/apple-watch	0.9475464102
https://www.macrumors.com/roundup/iphone-12	https://www.macrumors.com/roundup/iphone-12-pro	0.9106181338
https://www.macrumors.com/roundup/homepod	https://www.macrumors.com/roundup/homepod-mini	0.8776293977
https://www.macrumors.com/push	https://www.macrumors.com/how-to	0.8709816611
https://www.macrumors.com/guide	https://www.macrumors.com/how-to	0.8685640079
https://www.macrumors.com/roundup/macbook-pro-13	https://www.macrumors.com/roundup/macbook-pro	0.863087178
https://www.macrumors.com/push	https://www.macrumors.com/guide	0.8530616654
https://www.macrumors.com/roundup/apple-watch-se	https://www.macrumors.com/roundup/watchos-7	0.8432746325
https://www.macrumors.com/roundup/airpods	https://www.macrumors.com/roundup/airpods-pro	0.830882101
https://www.macrumors.com/roundup/ipad	https://www.macrumors.com/roundup/ipad-mini	0.830508208
https://www.macrumors.com/roundup/apple-watch	https://www.macrumors.com/roundup/watchos-7	0.8267231865
https://www.macrumors.com/roundup/apple-tv	https://www.macrumors.com/roundup/tvos-14	0.8088004493
https://www.macrumors.com/roundup/ipad	https://www.macrumors.com/roundup/ipad-air	0.7961241496
https://www.macrumors.com/roundup/macbook-air	https://www.macrumors.com/roundup/macbook-pro-13	0.7951967392
https://www.macrumors.com/roundup/iphone-11	https://www.macrumors.com/roundup/iphone-xr	0.7777272804
https://www.macrumors.com/roundup/iphone-12-pro	https://www.macrumors.com/roundup/iphone-13	0.7670274005
https://www.macrumors.com/push	https://www.macrumors.com/review	0.7625310583
https://www.macrumors.com/roundup/iphone-12	https://www.macrumors.com/roundup/iphone-13	0.7608913833
https://www.macrumors.com/how-to	https://www.macrumors.com/review	0.7577140075

Figure 19: Similar Page Checker results for macrumors.com

But if we check the reported 1035 rows for cosine similarity, average value is 0.3184 which is really low and good for the website. They really focus on creating a unique content.

3.6. Future Work

Since the topic and the roots of the project based on daily tasks of digital marketers, these scripts need to be updated frequently based on latest updates and features. But while writing this paper, I have noticed that if I would have 6 more months to work on this project, I would aim on the visualisation of results, and it would be great if I can run NLP algorithms for keywords and contents.

Visualisation could be useful for end users because it could help them understand easily and the script become more user friendly. It could be creating word clouds with keyword outputs or network visualization with connections.

Natural language processing for keywords and content can help the script a lot. Right now, the script cannot understand the words or the connection of the words. With this feature, the script can group keywords and even maybe recommend different keywords for the content.

REFERENCES

[1] «HTTP (HyperText Transfer Protocol),» [Online]. Available: https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

[2] «SemRush,» [Online]. Available: https://www.semrush.com/

[3] "SemRush," [Online]. Available: https://www.semrush.com/kb/31-site-audit#2.

[4] «Screaming Frog,» [Online]. Available: https://www.screamingfrog.co.uk/seo-spider/

[5] «SEMRush vs Screaming Frog Crawling Tools,» [Online]. Available: https://onpage.rocks/semrush-vs-screaming-frog-crawling-tools/#:~:text=While%20SEMRush%20does%20give%20you,5%20different%20websites%20per%20month.&text=Screaming%20Frog%2C%20on%20the%20other,many%20pages%20you%20can%20crawl

[6] «SemRush User Guide,» [Online]. Available: https://www.screamingfrog.co.uk/seo-spider/user-guide/

[7] «Screaming Frog Pricing,» [Online]. Available: https://www.screamingfrog.co.uk/seo-spider/pricing/

[8] «Top 100: The Most Visited Websites in the US,» SEMrush, [Online]. Available: https://www.semrush.com/blog/most-visited-websites/