Title: **3ºrd Report: Unveiling the mechanisms behind Google's search tool**

Author: **João Dias**

Date: **04/07/2020**

Index

Introductory Note:

Finding accurate and helpful information online can seem like simple task which most of us take for granted. A quick google search will most likely show us what we want to find in the first few results and links. However, making that a possibility is a task that is way harder than it seems. With over 2.5 quintillion bytes of data being created and uploaded to the internet every single day, filtering this information seems like an unfeasible task, especially when considering that there are almost 2 billion websites currently on the internet. So, the main questions are: how is Google able to do it? How is that amount of filtering possible? How do they seem to always know what to show? How are the expected results still found, even when the search contains misspelled words?

In the following sections of this essay, we will go into further detail about all these questions and their respective answers.

We will study how Google organizes the information in Section 1, and the algorithms and tools used by Google for the search tool will be analysed in Section 2.

1. Section 1 – Crawling and Indexing

Even before making a simple google search, a lot of search and organization over the whole web is made. In order to understand how search works, it is essential to have a better knowledge of how the information searched is gathered and organized.

1.1 Crawling

The **Crawling** process begins with a set list of URLs, fetched from previously made crawls and/or *sitemaps** given by the website owners. These URLs are subsequently visited by crawlers, who are then responsible for identifying all the URLs on the visited page and, afterwards, adding them to the list of URLs to visit. This list is a gigantic database of URLs that is stored at Google's servers, and is essential for when a search is made and the content of one of the URLs is useful. Due to its limited number of URLs, because of the amount of volume that it requires, crawlers focus a lot on trying to keep only updated versions and trying to avoid duplicate pages as much as possible. The crawl targets and the amount of URLs saved are all determined by a Google computer program which has objectives that change based on occasion.

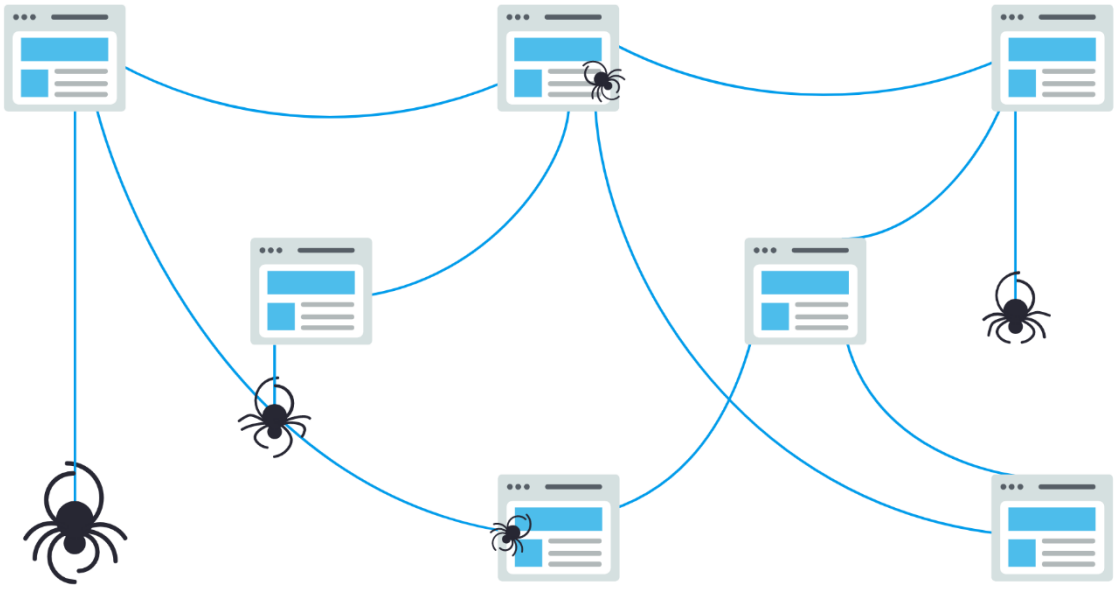

Figure 1 – Web Crawling

*sitemap : XML file that provides information about the files, pages, videos, amongst others, that are present on a specific website, giving extra information about each URL. For instance: when it was last updated, how often, etc. It allows for a quicker and more effective crawling process of the website, and it is useful for

newly created websites and websites that contain large amounts of files, videos or external links.

1.2 Indexing

When a crawler finds a webpage, a series of processes occur. These range from adding the URLs found to a list, which we explored in the previous section, to taking crucial keywords and signals of the website being crawled and adding them to a search index. Every word contained in a website is indexed. This way, when a web page is indexed, it is added to all the entries of the words it contains.

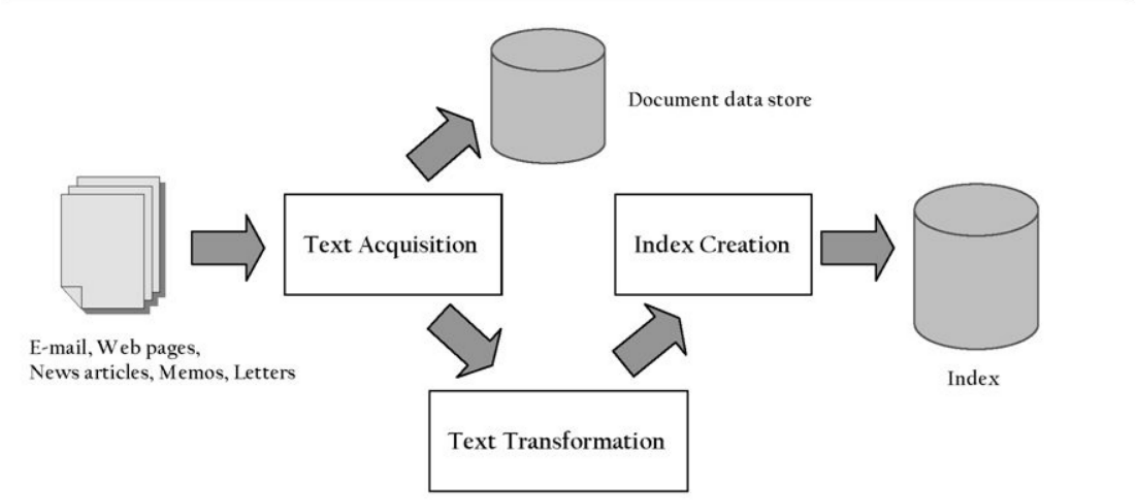

Figure 2 – Indexing process

By doing this, speed and performance are optimized when there is the need to find admissible documents for a search query. These high speeds are what allow for features such as the Knowledge Graph to be created.

1.2.1 Knowledge Graph

The **Knowledge Graph** fetches and aggregates information with the use of machine learning algorithms and data interlinking, in order to model out important information about a specific topic. One of the most common examples is when we Google search a relatively famous person.

Figure 3 – Example of Knowledge Graph

As we can see, the main basic information about that specific person is displayed in a box to the right of the other results. This allows for basic google searches to be quicker and for the desired information to be found easily. More examples of Knowledge Graphs being used in a helpful way can be seen in the following examples.

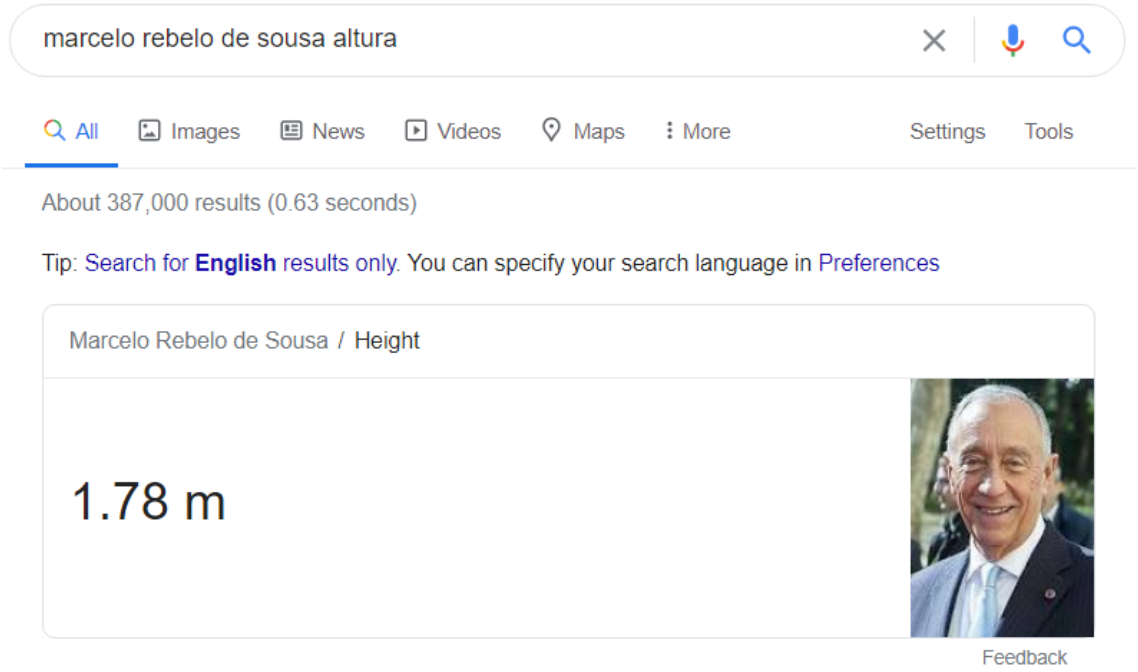

Figure 4 – Example of how direct and specific information is displayed

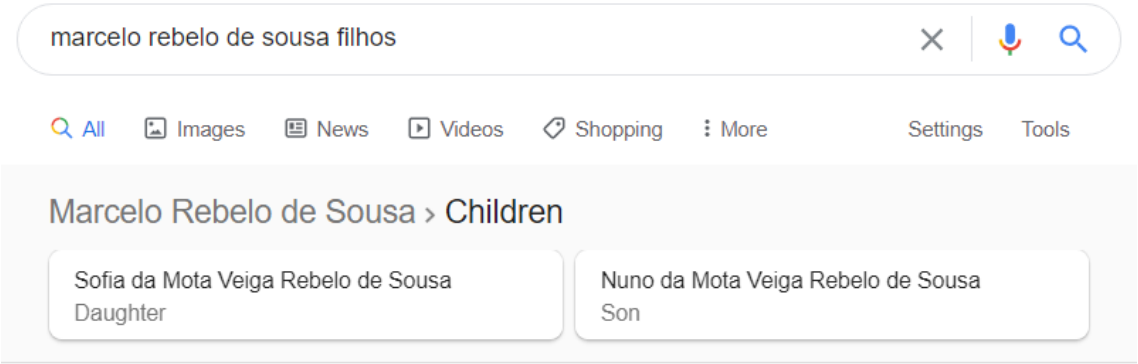

Figure 5 - Example of how direct and specific information is displayed

A lot of extra features, such as directions, traffic and meteorology information, are also used by Google.

2. Section 2 – Search Algorithms

2.1 Understanding what the user wants

In order for the results of a search to be accurate and precise, the user's query has to be understood. Language models are built and used by Google to try to decode what set of words should be looked up on the index. The search can be processed in very different ways from query to query: it can be a question, an object, it can be misspelled, etc. So, let us look at how each one of these situations is dealt with.

A lot of queries, despite being differently written, have the same meaning. To face these scenarios, Google came up with a system that took over five years to develop and that can recognize that, for example, the queries "How to discover friends on steam" and "How to find friends on steam" mean the same. Consequently, a lot of the pages displayed from one query will be the same as the ones displayed by the other.

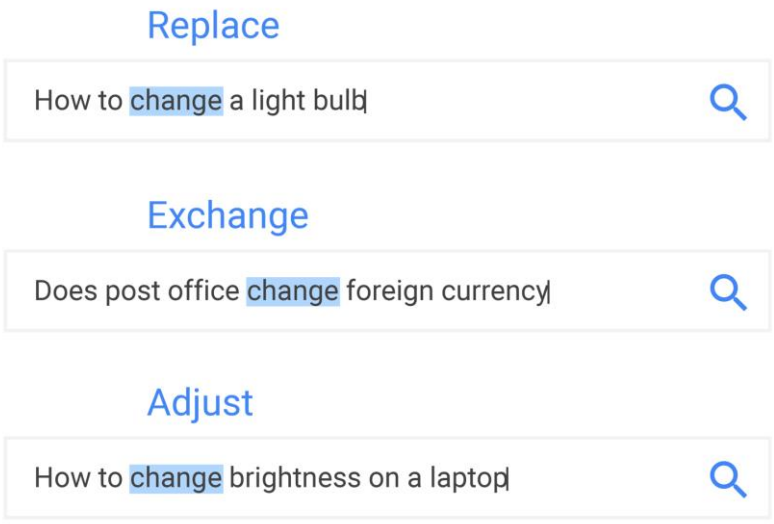

Figure 6 – Words have different meaning

If a query is written in a language (e.g. Portuguese), it means that the user is most likely looking for results in that same language.

Keywords such as “review” or “schedule” can be identified and understood as the user wanting a specific information of the query. These can also be heavily influenced by the time in which they are searched. Keywords can be “*trending*”, meaning that if you search for “Primeira Liga”, you will want to be shown information about the 2019/2020 league and not the one 10, 20 or 30 years ago. This is something that Google refers to as “***freshness***”. Information can change quickly, and whilst some topics might still be relevant after a week, other ones may lose relevance within just a couple of hours. A page that contains a recipe for a chocolate cake from 2012 might still be relevant today, but not so much the score of a football match played 20 years ago, unless it is directly searched.

Despite still being improved roughly on a daily-basis, the algorithm is already very developed and effective when it comes to recent searches. Google’s performance allows for the users to have an immediate response to everything and anything, from basic information, such as the weather and the daily news, to bigger concepts, for instance, historical or scientific facts. Consequently, this great experience translates into the general public having a preference for Google over other search engines.

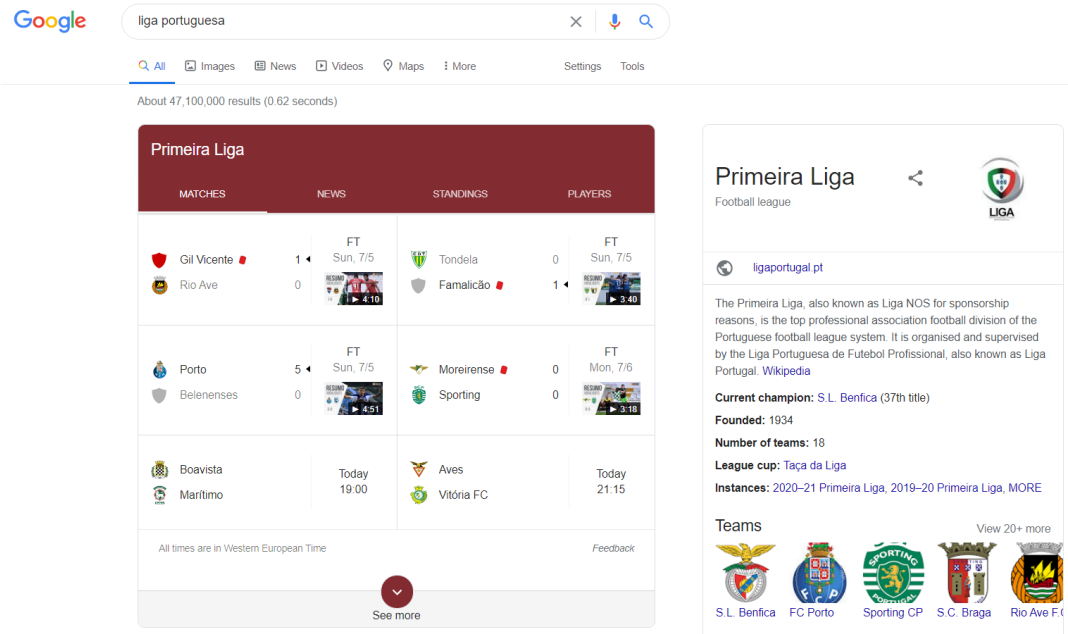

Figure 7 – Results are adapted to the ‘*freshness*’

Another feature that we take for granted these days is the “***did you mean***” feature. Spelling errors happen to us all the time, so it is important that search engines have the ability to fix our mistakes and try to understand what set of words our query was meant to have. This feature shows a suggested search option that not only allows for the correction of the query but can also lead the user to different options that broaden the search field.

At the moment, there is not much information about what is behind this feature. However, it is thought that it most likely consists of a mixture of existing algorithms and continuous machine learning development, that provide a feature as accurate as this. Let us analyse two of these algorithms.

2.1.1 K-Gram Index

The **k-gram index** algorithm focuses on suggesting the substitution of a word for another that it believes to be similar. So, what are “K-grams”?
K-Grams are k-length subsequences of a word. The k values change depending on the situation, but usually k = [1 (unigram), 2 (bigram), 3 (trigrams)] are the most used.
A **K-gram index** is a list of words that contain the k-gram string. If our bigram k-gram string was “it”, our k-gram index would look something like this:

*[notw**it**hstanding, heterosexuali**ty**, incompatibili**ty**, hosp**it**alization, electroposi**t**ive, corrigibili**ty**, controllabi**li**ty, ...]*

With the K-gram index of every k-gram created (figure x), it then finds the **k-gram overlap** between all the lists by using the **Jaccard coefficient** (figure y). This coefficient will pick possible candidate terms and scan them throughout the lists. The candidate with the higher coefficient will be the one picked as the most credible correction. If we look at the example in figure x, the correction picked would be apple

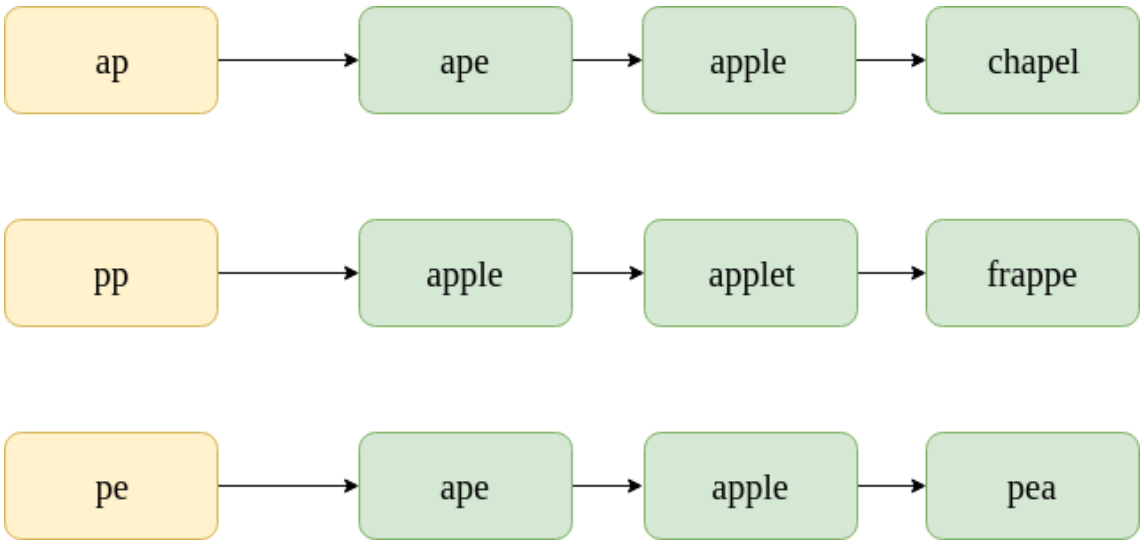

Figure 8 – K-gram index example

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Figure 9 – Jaccard Coefficient

2.1.2 Levenshtein Distance

The **Levenshtein distance** is a string metric that, when given two strings, measures the least amount of edits an algorithm would need for it to be able to correct a string into the other. The distance between two strings, a and b, with the length |a| and |b| respectively, is given by the following:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

"Here, 1(ai≠bi) is the indicator function equal to 0 when ai≠bi and equal to 1 otherwise, and leva, b(i,j) is the distance between the first i characters of a, and the first j characters of b.

It is important to emphasize that the first element in the minimum corresponds to deletion (from a to b), the second to insertion, and the third one to match or mismatch. This depends on whether or not the respective symbols are equivalent." - Wikipedia

How exactly is this applied?

1. We start by creating a matrix (m, n) in which the cell's size is equal to the m-character prefix of one with the n-prefix of the other string.
2. The matrix is filled from the top left corner to the bottom right one.
3. A horizontal jump is equivalent to a delete, and a vertical jump corresponds to an insert.
4. Cost is set to 1 for each operation.
5. The diagonal jump costs 1 if the two characters in the row and column do not match. If they do, it costs 0.
6. The final number in the lower right corner is known as the **Levenshtein distance** between the two strings.

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

Figure 10 – Levenshtein Distance

2.2 How Search Results are selected

Besides matching the strings in the user's query with the index of the relevant pages on the web, it is the search algorithms that prioritize the quality and reliability of such pages. The algorithm will assign a rank to each page based on various factors. Some of the factors that are taken into consideration when giving a rank to a page are: a page that shows expertise, trustworthiness, and which keyword is mentioned a lot. Pages that have a high rank will usually appear farther up in the search result, meaning that the first pages linked for a search are going to be the "best" and better ranked pages. The algorithm responsible for ascribing the ranks to the pages is called **PageRank**. We are going to analyze how it works.

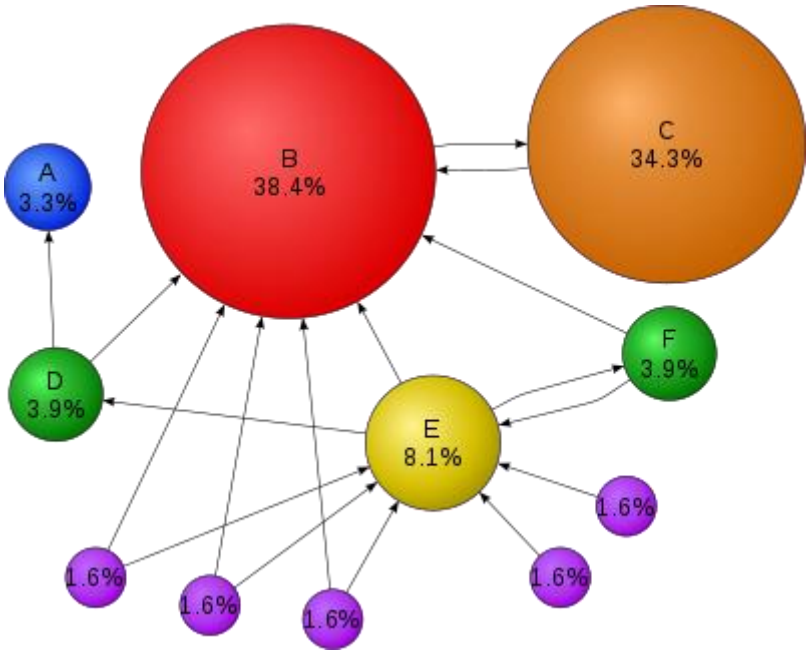

Figure 11 – PageRank Example

The rank attributed by the algorithm is partially established by the number of other pages that link to that specific one. Not every link has the same value though, a link from a trustworthy and high-quality page will be much more valuable than a link from an ordinary and not very known page. This also prevents the creation of a lot of web pages that link to yours in order to get a higher page rank. The number of links a page has also has an impact on how valuable those are. If a page links to 100 others, those links will have a lesser value than if that same page only linked to 10 others. The “age” factor is also pretty important. If a page and the links are up for a couple of years, those links will also be deemed more valuable.

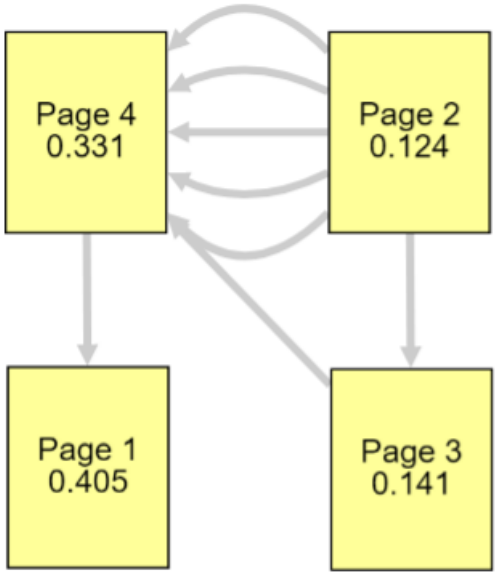

Figure 12 – PageRank Example

For a better visual understanding of the PageRank algorithm, I created a video facilitate it’s understanding: https://youtu.be/ynyQ_N9jXTA

PageRank PR ( ) is bestowed by an outbound link is equal to the page’s own PageRank rank divided by the number of outbound links L( ) and it can be obtained by:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)},$$

For a better understanding of how the algorithm might be implemented, a python script was created. The purpose is for the user to see first-hand an application of the algorithm. It has a brief explanation of what the algorithm is (figure 1), what arguments the function needs and what they mean (figure 2), and the user is able to see the PageRank values of the inputted graph of nodes (figure 3).

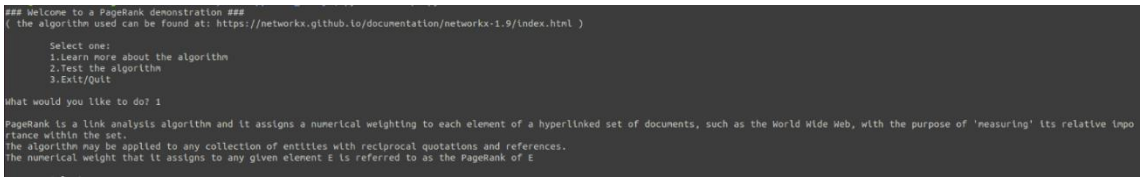

Figure 13 – Intro to the algorithm

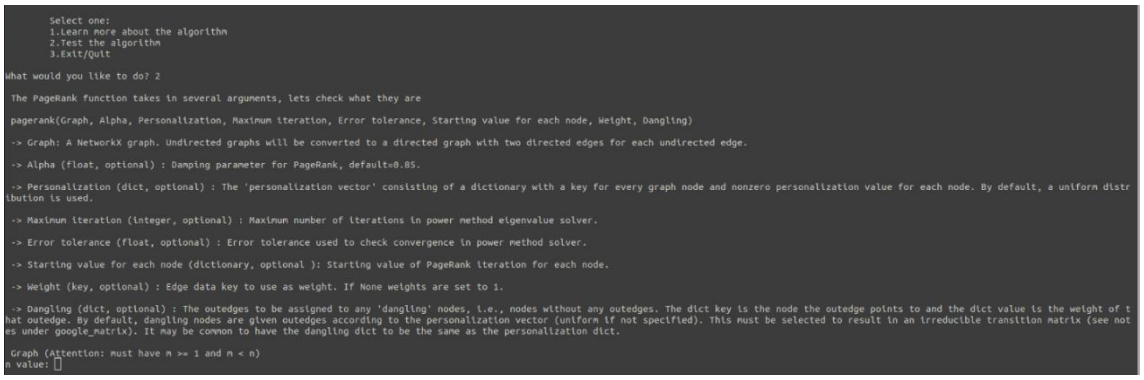

Figure 14 – What variables go into the funciton

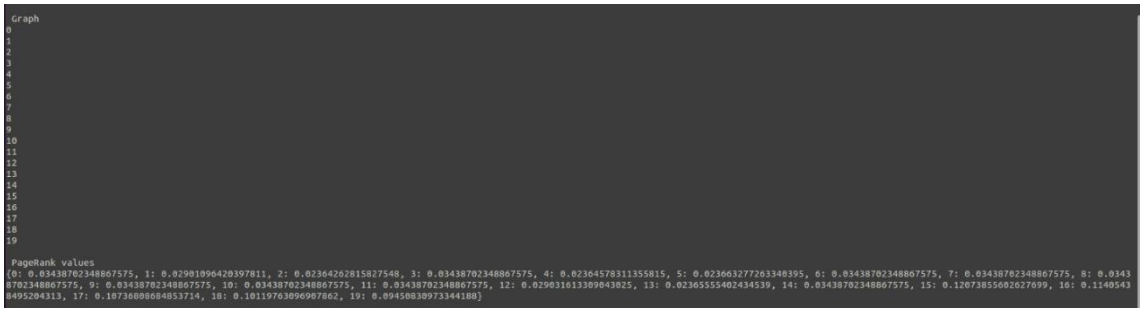

Figure 15 – Final PageRank

3. Conclusions

As thousands of websites are created and uploaded every single day, the web is constantly evolving. We may not notice it, but that evolution has its impact on our search results. After analyzing what happens “behind-the-scenes” when it comes to Google search, it is safe to say that everything is thought-out carefully and according to the users’ needs. Not only do they pay attention to detail and practicality, they also are able to deliver a good service with very effective results. However, as we have seen throughout this essay, this is a tool that constantly needs work, in order to adapt, improve, and be consistent with the daily development of the web.

The essential information about Google search is covered and explained in this essay, with the help of the video and with the script, but since it is so complex, there are more details to it, which can be found online. Also, a lot of information is simply not known, not provided, or not available to the public.

4. References

- https://www.google.com/search/howsearchworks/crawling-indexing/
- https://moz.com/beginners-guide-to-seo/how-search-engines-operate
- https://en.wikipedia.org/wiki/Sitemaps
- https://support.google.com/webmasters/answer/156184?hl=en
- https://www.google.com/search/howsearchworks/crawling-indexing/
- https://moz.com/beginners-guide-to-seo/how-search-engines-operate
- https://www.shoutmeloud.com/google-crawling-and-indexing.html
- https://thenextweb.com/podium/2019/06/11/what-is-a-knowledge-graph-and-how-does-one-work/
- https://computer.howstuffworks.com/google-algorithm.htm
- https://neilpatel.com/blog/how-google-search-engine-really-works/
- https://www.google.com/search/howsearchworks/algorithms/ https://en.wikipedia.org/wiki/PageRank#:~:text=PageRank%20(PR)%20is%20an%20algorithm,in%20their%20search%20engine%20results.&text=PageRank%20is%20a%20way%20of,how%20important%20the%20website%20is.
- https://www.link-assistant.com/news/google-page-rank-2019.html
- https://itenterprise.co.uk/how-does-googles-did-you-mean-algorithm-work/