UNIVERSITY OF CAMERINO

School of Science and Technology

Degree in Computer Science

Software and systems for industries

(Class LM-18)

A Framework for web presence sentiment analysis

Graduand :

Andrea Marcelli

Supervisor:

Prof. Fausto Marcantoni

Academic year 2016/2017

Abstract

(English)

The World Wide Web has gone a long way since its early days in development at the CERN Laboratories. Now holding immense relevance in everyday activities, being intertwined with most of the people lives on this planet, its network of links and applications allow users to reach any kind of content they want. The huge quantity of data hosted on the Internet can make the difference in a number of diverse applications, but, most importantly for the scope of this work, can be useful in determining with a certain degree of precision, the polarity of opinions held by communities, users and newspapers alike with respect to a certain individual or business. As a matter of fact, data can hold either a positive or negative meaning when stating something about a topic, or in even in more general terms, as the name of an individual or business can be cited alongside positive or negative content. This activity, called “Sentiment Analysis” has been actively researched upon in several influential papers. This work is based on the framework for Sentiment Analysis created by **Bing Liu** and cited in several researches. However, it aims at simplifying it by tailoring to a different, more specific context with respect to the more general one that a good general framework such as that developed by Liu should address. This work introduces the concept of “influence score”, that is the strength with which a text is said to be able to impact the reputation of the search subject. By using this companion concept, the simplified framework proposed in this work tries to address the issue of determining, in a naive way, the amount of impact that any content can have on either the individual or business at hand. The framework itself tries to follow a strictly scientific process by first analyzing the problem with use of Data Science principles and then by defining additional framework components and concepts to be used alongside said principles. The companion tool to the framework is developed to provide a first implementation of the concepts by examining results pages (SERPs) obtained by searching the web be means of a search engine in order to analyse each SERPs item separately, to obtain its sentiment and influence score and to provide, overtime, a monitoring functionality for the user. This document takes into account the different items at play that interacts within the framework implementation, such as Search Engines (Chapter 1), Data Science (Chapter 2), Explorative Algorithmic Techniques (Chapter 3), Web Reputation (Chapter 4), The Tool (Chapter 5), finally dwelling on future possibilities and additions to the work in Chapter 6.

(Italiano)

Il World Wide Web ha raggiunto uno stadio di sviluppo ed utilizzo estremamente avanzato rispetto a quanto era stato preventivato all'epoca del suo iniziale sviluppo ai laboratori CERN. Difatti, si può affermare che esso abbia raggiunto una notevole importanza nelle attività di tutti i giorni. Essendo le persone della maggior parte dei paesi poste a stretto contatto con esso, la sua rete di collegamenti e le numerose applicazioni native, esso consente ai suoi utenti di sfogliare qualsiasi tipo di contenuto essi vogliono. La grande mole di dati ospitata da Internet può essere impiegata in numerosi ambiti e diverse applicazioni ma, nel contesto di questa ricerca, rimane di maggior interesse la possibilità di utilizzare tali dati al fine di determinare, con un certo grado di precisione, la polarità delle opinioni espresse da comunità online, utenti o perfino giornali web circa un determinato argomento, individuo o business. Difatti, si può affermare che i dati possano essere caratterizzati da un orientamento, sia esso positivo o negativo, sia nel contesto specifico di un argomento di ricerca, che in termini più generali, quando tale soggetto viene citato in contesti caratterizzati da contenuto positivo o negativo. Questo genere di attività, denominata "Analisi del sentimento" (in inglese, Sentiment Analysis) è stata largamente studiata in diversi importanti studi in materia. Il seguente lavoro, si basa sul lavoro proposto da **Bing Liu**, che introduce un framework per l'analisi del sentimento, citato in numerose ricerche successive. Nonostante sia basato sul lavoro di Liu, il presente lavoro si propone di introdurre un framework semplificato e adattato a un contesto più specifico rispetto a ciò per cui il framework originale era stato inizialmente progettato. In particolare, si introduce il concetto di "punteggio di influenza" , definito come la misura con cui un particolare testo è in grado di influenzare la reputazione del soggetto della ricerca. Utilizzando questo concetto, il framework si propone di determinare, in modo semplice, la quantità di impatto che un determinato contenuto può avere sull'individuo o business in esame. Il framework si basa su di un processo scientifico che prevede una fase di iniziale di analisi dei dati attraverso l'uso dei principi della Scienza dei Dati e la definizione di ulteriori componenti e concetti interni al framework stesso. L'applicazione sviluppata come prima implementazione del framework analizza le pagine risultanti (SERPs) da una ricerca web via motori di ricerca analizzandone ogni elemento separatamente, in maniera tale da ottenerne i relativi valori di sentimento e punteggio di influenza per arricchire l'utente, come risultato di un'analisi continua, di una utile funzionalità di monitoraggio della propria reputazione. Questo documento prende, inoltre, in esame tutte le varie componenti che interagiscono tra loro all'interno dell'implementazione del framework, come i motori di ricerca (Search Engines, Capitolo 1), la scienza dei dati (Data Science, Capitolo 2), le tecniche esplorative dei contenuti (Explorative Algorithmic Techniques, Chapter 3), reputation online (Web Reputation, Chapter 4), l'applicazione (The Tool, Chapter 5) analizzando, in ultima istanza, le eventuali aggiunte e sviluppi futuri nel Capitolo 6.

1. Search Engines: finding the needle in the haystack

Defining the importance of finding *what* is needed, *when* is needed in a big scary haystack such as the World Wide Web can be examined from different points of view. This chapter, starts discussing the nature of the problem by going exactly at the heart of the web: the way it identifies its resources.

1.1. At the heart of the web

The World Wide Web has given us the ability to aim for any kind of knowledge at any given time. The average web surfer can search for a number of different keywords related topics and be given a set of related information in a matter of seconds. However, the reality of the web was not always like this. Finding any kind of information regarding a topic of interest often forced the user to know the exact wording of a website's title, in order to find the related resources. In even darker times, when no means of searching the web were at all available, a web surfer needed to know the exact address of a resource in order to access i. In fact, the World Wide Web once was - and still is, to some extent - just a mere collection of poorly organized nodes (i.e. documents). Since its inception, the World Wide Web presented its users with the peculiar memory challenge of remembering the exact character pattern of a page's web address in order to reach it. As everyone knows, all of its available content is uniquely identifiable by a unique string of text (also called a URL), hence someone approaching the web looking for some content needed to remember a combination of characters often including pretty foreign symbols to the casual user. This is made even more evident by taking into account the **RFC 3986** document, authored by Tim-Berners Lee, that defines URL's syntax as a combination of the following elements:

<scheme>://<username>:<password>@<host>:<port>/<path>;<parameters>?<query>#<fragment>

The document further goes into more details regarding the syntax specifics, by listing all of the characters that could - or could not - appear in the URL at some point. In particular, the characters allowed in its syntax can count an astonishing 20 reserved characters, and 40 unreserved characters including the whole english alphabet (consisting of 26 characters), digits from 0 to 9, as well as the "-" / "." / "_" / "~" symbols and some more characters, specifically labelled as being "unsafe" to be used in a URL. The following table summarizes the RFC's URL symbols:

Classification	Included Characters
Safe characters	Alphanumerics [0-9a-zA-Z], special characters $-_.+!*'(),, and reserved characters used for their reserved purposes (e.g. question mark used to denote a query string)
ASCII Control characters	Includes the ISO-8859-1 (ISO-Latin) character ranges 00- 1F hex (0-31 decimal) and 7F (127 decimal)
Non-ASCII characters	Includes the entire “top half” of the ISO-Latin set 80-FF hex (128-255 decimal)
Reserved characters	; / ? : @ = & (does not include blank space)
Unsafe characters	Includes the blank/empty space and “ < > # % { } | \ ^ ~ [ ]

Table: Characters and URL(s)

As hinted at by the above specifications, the combinations are potentially countless, making the basic naive web surfing model of proceeding by memory a very confusing approach. A URL containing any of the following characters (ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-._~:/?#[]@!$&'()*+,;=`) would for example still be considered a valid one, leading to surprisingly well formed urls, as peculiar as the following ones[1]:

http://www.thelongestdomainnameintheworldandthensomeandthensomemoreandmore.com/

http://3.14159265358979323846264338327950288419716939937510582097494459 2.com/

http://www.abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijk.com/

http://www.llanfairpwllgwyngyllgogerychwyrndrobwyll-llantysiliogogogoch.com/

URLs, by their nature, can then be difficult to remember. Of course, researchers understood the issue a long time ago and began investigating possible solution to this problem, often hitting dead ends or obtaining very primitive - early - results. The difficulty of solving the problem is made even more obvious when taking into account all of the different approaches that were investigated before Google - the real game changer - became indexing the web. The following paragraph will present a brief history of search engines.

[1] The World's Longest Domain Name, http://archive.oreilly.com/pub/post/the_worlds_longest_domain_name.html

1.2. Search Engines

Search Engines are at the top of every internet browsing session. Users assumes they have always been there, guiding us through the process of selecting a set of pages that would fit in with the query context.

1.2.1. Browsing the past

First of all, let's ask what is a search engine? A search engine is a program that searches the web for sites based on your keyword search terms. The search engine takes user's keywords and returns search engine results pages (which are commonly referred to as "SERPs"), with a list of sites it deems relevant or connected to your searched keyword.[2] As the story goes, the first comers not always sits at the winner's table. Most of the names that will be featured in this short chrono-story are long forgotten and gone, tangled by the spider's web (i.e. the web itself).

Before any search engine was formally defined, there were a number of websites that acted as collective directories for the indexable web. These websites became appearing as early as year 1990 and the early example was the **Archie**[3] website, which featured a index of downloadable directory listings as well as a query form from which to perform searches. The following year, the **Gopher**[4] protocol was very popular amongst researchers and, consequently, the Gopher index systems became to be searched upon by early search engines such as **Veronica and Jughead**[5] (1991). However, the most popular web directory was the **Yahoo! Directory**[6], launched in 1994, that began as a collection of favorable web pages, but soon became a searchable directory, given its size. Automatic crawling and indexing of the web was soon found to be a more viable alternative to static directory indexing and first appeared with the **World Wide Web Wanderer**[7] in 1993, shortly followed by a number of big players in 1994, such as:

- **Infoseek**[8] (employed by Netscape as their default search engine)
- **Altavista**[9] (the first to allow for natural language queries and the use of advanced searching techniques)
- **WebCrawler** (first crawler that indexed entire pages)
- **Lycos** (capable of identifying 60 million documents, more than any other search engines at the time)

---

[2] The history of search engines - an infographic, http://www.wordstream.com/articles/internet-search-engines-history
[3] The archie aearch engine, the world's first search, http://www.makeuseof.com/tag/the-archie-search-engine-the-worlds-first-search/
[4] RFC1436, the internet gopher protocol, https://tools.ietf.org/search/rfc1436
[5] Veronica and Jughead, https://bobbydotseo.wordpress.com/veronica-jughead-1991/
[6] Yahoo! Search Engine & Directory, https://www.livinginternet.com/w/wu_sites_yahoo.htm
[7] World Wide Web Wanderer, http://www.internet-guide.co.uk/WorldWideWebWanderer.html
[8] Where Are They Now? Search Engines We've Known & Loved, https://searchenginewatch.com/sew/study/2064954/where-are-they-now-search-engines-weve-known-loved
[9] THE HISTORY OF THE ALTAVISTA SEARCH ENGINE, https://digital.com/about/altavista/

The above search engines were fundamental in making huge steps in organizing the Internet, by gathering most of it in one place. They had, however, one big disadvantage: being early, tentative implementations, they often display a lack of “refinement” in their results, requiring their users to search through a long list of resources to find what they needed. Sometimes, it takes just the right idea to win a competition that has already started. Indeed, Larry Page and Sergey Brin were convinced they had one and began working on **Google** as Phd students at Stanford University in 1996. Their search engine would use backlinks for measuring the “authority” (i.e. reliability) of a resource for indexing purposes. A mechanism that still is at the root of the PageRank algorithm employed by Google today. The following years marked the arrival of popular search engines such as **Ask.com** (1997), **Msn** (1997), **Yahoo! Search** (2002), **Cuil** (2008) and **Bing** (2009).

As peculiar as it seems, no other search engine could match the algorithm that was developed for the Google search engine and the diverse factors examined by it to provide the best possible result. This infographic[10], showing the desktop search engine market share in the time period between January 2017 and August 2017, shows the preferences in usage of search engines by Internet users as favourable to Google (as expected):

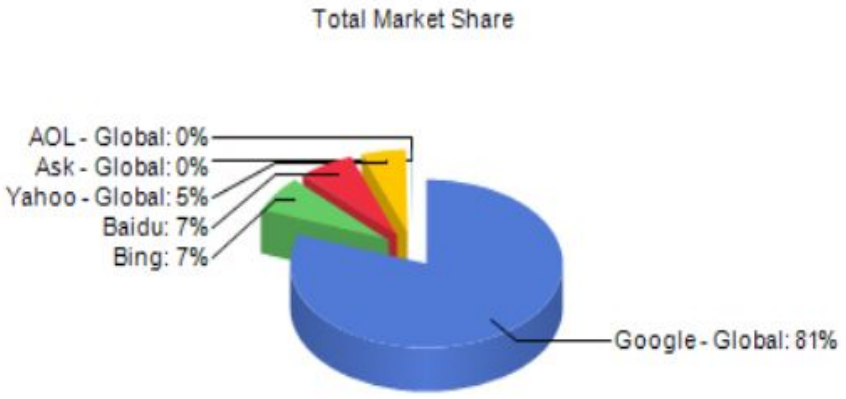


Image: desktop engine market share[3]

[10] Market Share Statistics for Internet Technologies, http://marketshare.hitslink.com/search-engine-market-share.aspx

Search Engines	Total Market Share
Google - Global	79.79%
Bing	7.21%
Baidu	6.52%
Yahoo - Global	5.14%
Ask - Global	0.16%
AOL - Global	0.05%
Excite - Global	0.01%

Table: market share statistics

In the next paragraph, some of the publicly available details of the Google's "page rank" algorithm and "web surfer random model" are briefly examined, alongside with a different, interesting approach to web search: the Hits search algorithm.

1.2.1.2 Directories vs. Search Engines (The Yahoo example)

An Internet directory can be regarded as an alternative Internet searching tool. Its implementation is quite different than that of commonly used search engines. Internet directories are catalogs of small World Wide Web portions, and the documents are usually selected and indexed by people rather than in an algorithmic way. Documents are divided in categories, and each category is further divided in subcategories. The **Yahoo!** Web directory

is probably the most famous and long lasting example of web directories, lists that were once used in an era before any modern search engine was proven to be really working. The Yahoo! Web directory was so popular because of its effectiveness, that was granted by a very simple distinctive feature: the human factor. With the ability of reviewers to rank pages by topics and arrange them in the right categories, directories had an earlier important advantage over the primitive automated search implementations that were in use at **Excite** and **Infoseek**.

The **Yahoo!** Directory, however, suffered from several drawbacks because of its human editors. Websites had often to wait long delays before being included in the directory, while those that had been included could also found that the editor had missed some important keywords in categorizing it.

When **Google** came along, it's algorithm could provide results so good that it actually proved that automated search through web crawling could rival, and even surpass, the **Yahoo!** directory's one. Searching with Google provided with many more relevant results that a human maintained directory ever could, thus ending its supremacy as the go-to search instrument.

The **Yahoo!** Directory continued to stay active up until 2014, following a slow and painful decline.

1.2.1.3. Other types of search engines

While Internet users are mostly familiars with Web search engines the likes of **Google** and **Altavista**, there exists other types of search engines, usually specialized in searching a specific subcategory of results or looking to broader the search with respect to more classical search engines. The following list is a brief summary of other types of search engines:

- **Meta Search Engines**: are search engines running a query through multiple search engines. Meta search engines work by restricting their searches to the subset of features commons to all search engines used during the search. While they are capable of showing a broader view of a topic, Meta Search Engines suffers from a variety of problems, including timeouts and the lack of advanced search features. Some notable Meta Search Engines includes **Dogpile, ixquick** and **Metacrawler**
- **Specialized Search Engines**: are usually specific to a collection of databases. They range from providing search functionalities over research databases, library databases and private database. These Specialized Search Engines match subject-specific databases to the searched topics and includes the likes of **Expedia** and **IMDB**
- **The Invisible Web Search Engines**: are search engines specifically created to search on the Invisible Web (i.e. the so called "Deep Web"), a portion of the web that cannot be crawled or indexed by usual search engines. An example of such search engine is **Profusion**

1.2.2. The Algorithms and the theoretical framework

At the very core of search engines lies algorithms, modelled after theoretical frameworks developed by researchers in the fields of Natural Language Processing and Information Retrieval.

1.2.2.1. Elements commons to all search engines

Web Search engines concept stems from the broader Information Retrieval (IR) concept of document search engines. “Search engines match queries against an index that they create. The index consists of the words in each document, plus pointers to their locations within the documents. This is called an inverted file” [11]. A search engine or IR system comprises four essential modules :

- **A document processor:** tasked with operations consisting of properly pre-processing document, deleting all the “text noise” (e.g. stopwords, punctuation), finally finding relevant elements to be indexed, creating the so called reverse index used when comparing a query with the indexed documents
- **A query processor:** is used to process the query input provided by the user. It can be comprised of several steps, that can be left unimplemented if the search engine values time of response over quality. At the very least, a query processor should be able to tokenize the input, turn it into a query. Once the query has been generated, the search engine is ready to match it against the reverse index
- **A search and matching function:** is usually implemented as a binary search and provides the functionality to match the query to the inverted index files
- **A ranking capability:** the search engine should posses the ability to properly rank pages by using an appropriate algorithm. The more parameters the algorithm takes into account in order to rank pages, the more accurate the final result will be

These basic search engines components will be further explored in *1.2.2.3. General modules of a search engine: a more in depth look*.

1.2.2.2. Elements search engines differ in

While search engines share many similarities in the way they work, by retrieving and indexing documents, the implementations by which they crawl ad index the internet is proprietary to each search engine. It’s these proprietary implementations that causes search engines to return different results. In particular, search engines can be said to differ in:

- **The quantity of a document they choose to index**. While some search engines are only interested about indexing the title and a few text line of each documents, other - bigger players - often choose to index every word of a document.
- **Index update frequency** is also a distinctive feature. Some search engines choose to update their index on a daily basis, while others might decide to do so every

[11] How a Search Engine Works, http://www.infotoday.com/searcher/may01/liddy.htm

couple of months. Google is an example of a search engine frequently updating its index.

- **Their field of view**, or, to put it simply, the amount of web they are capable of searching. As the growth of the web has reached a pinnacle in the last decade, it is not an easy feat for search engines to be capable of searching it all. Some research actually shows that search engines are capable of searching the 16% of the World Wide Web at best.
- **The way search engines compute the relevancy of a document** with respect to the user query. While all search engines uses statistics and probability (as well as some other internal heuristics), the actual implementations differ from search engine to search engine.
- **Truncation** of words. Some search engines, for example, might look for alternative forms a word such as plurals. Others might simply truncate the word to the word root (a process referred to as “stemming”).
- **Case sensitivity**. Some search engines are capable of distinguish between capital and lowercase letters. For example, search engines supporting case sensitiveness would return only capitalized results if the user’s query displays capital letters and usually return both lowercase and capitalized results for lowercase queries.
- **Keyword searching**. Most search engines usually implements keyword searching, meaning that they will search for documents containing the exact words used in the user query. Other search engines might, instead, choose a different strategy, by implementing **Concept searching**, which explore the query to retrieve documents related to the idea expressed by the user.
- **Related sites** are often provided by search engines to show more results that might be connect to the user query. Some search engines do that, while others don’t.

1.2.2.3. General modules of a search engine: a more in depth look

Each of the elements commons to the search engine’s core can be broken down to the ones already mentioned in paragraph *1.2.2.1. Elements commons to all search engines*. These elements are phases of the whole search process which, in turn, contains several steps that need to be followed to move to next phase. For each of the elements, the steps can be summarized as follows:

- **Document Processor**:
    - Preprocessing: at this step, all data are merged into a single consistent data structure that can be better handled by the following processing units
    - Identify elements to index: by following a strict set of internal rules the Document Processor is capable of tokenizing the text and identifying potential indexable elements
    - Deleting stop words: this step eliminates those terms that have little value in finding useful documents in response to a user's query. To delete stop words, an algorithm compares index term candidates in the documents against a stop word list and eliminates certain terms from inclusion in the index for searching

    - Term Stemming: at this stage, word suffixes are removed to reach two goals: An increased efficiency, as well as an increased effectiveness, A downside to stemming is that it may negatively affect precision
    - Extract index entries: the document processor extracts the remaining entries from the original document
    - Term weight assignment: weights are assigned to terms in the index file. The more sophisticated the search engine, the more complex the weighting scheme. The TF-IDF algorithm is often used for this task, as this algorithm measures the frequency of occurrence of each term within a document
    - Create index: the index or inverted file is the internal data structure that stores the index information and that will be searched for each query.
- **Query Processor**:
    - Tokenizing: the query stream is tokenized
    - Parsing: the system needs to parse the query first into query terms and operators, such as reserved punctuation (e.g., quotation marks) or reserved terms in specialized format (e.g., AND, OR)
    - Stop list and stemming: perform a stop-list and stem procedure on the query
    - Creating the query: a local query representation is created by using a model of choice, depending on the search engine implementation
    - Query expansion: some sophisticated systems may expand the query into all possible synonymous terms and perhaps even broader and narrower terms
    - Query term weighting: the final step in query processing involves computing weights for the terms in the query

1.2.2.4. Crawling and indexing

Search engines capabilities to provide users with a list of pages matching a query is strictly depending on its capability to crawl and index the web.

- Crawling, or spidering, is the process by which search engines discover new pages, changes to existing pages and dead links on the web. Crawling is performed by so called “bots” or “spiders”, small web programs capable of reading web pages. The bots choose which pages to crawl by following algorithmic procedures specific for each search engine and send back the information they find to the search engine. Crawlers are also capable of detecting links on a page, that will be crawled later to discover new content. The most famous example of search engine web scraper is of course **Google**’s owned **Googlebot** which is known to displaying more advanced features than other bots, in its ability to execute POSTS requests (web crawlers usually performs simple GET requests) and to execute Javascript and AJAX requests as well, seemingly behaving like a real web browser would. There exists multiple open source implementations of web crawlers as well, for example: Scrapy (a Python based tool), Nutch and Heritrix.
- Indexing, on the other hand, is the process by which search engines compute the crawled web pages to extract each word contained in the set to create an inverted index to be used upon a search operation on the search engine. Inverted indexes are usually preferred to forward indexes when creating the index. In fact, inverted

indexes provide direct access to documents containing a certain word when comparing it to the original query.

1.2.2.5. Relevance and popularity

Relevance and popularity are two important factors in determining how to order a result and what to display as well. However similar the two concepts may seem, they actually concerns themselves with different areas of text analysis. In particular, it can be said that:

- Relevance is a qualitative measure of how much the document's content returned by a search matched the user's query. The value increases with the number of occurrences of query items in the document. Relevance is often performed at a document analysis stage, when search engines look for query terms presence in important areas of the document (e.g. headlines)
- Popularity is an indicator measured through the amount of links (citations) occurring in a document returned by the search. Popularity of a document increases with every other document linking to it. Search engines usually evaluate popularity of a document during the link analysis phase, when search engines also evaluate the quality and **authority** of who is linking to the document.

1.2.2.5.1. Relevance

Relevance of a result is a very straightforward concept at first glance. However, it can be broken down to two different definitions:

- The document found through the search engines actually answers the question that prompt the user to start a search to begin with
- The user can somehow understand the reason why the search engine submitted the document to him in the first place

Both of these definitions comes with implementation implications. Clearly, the first definition is reminiscent of a more "black-box" approach, with respect to the second one. The user got the result he wanted and that is really the only important part. The second definition requires the user to have a visibility of how a search engine works, in order to understand why a document was retrieved following his query. While the first approach requires more computational power, it also yields more accurate results when appropriately tuned. The second approach, on the other hand, produces faster but weaker results, but, as the user is able to understand it, the query can be fine tuned to get more precise results in the following search iterations.

The first, non web search engines, were concerned with either applying the first or second definition to the letter, often suffering from many flaws when returning results. Before the Internet, in particular, the first definition was often used. Some more experimental search engines would even try to search for words correlated to single query terms, often returning better results but, in other, more dramatic cases, failing miserably by returning results which were incomprehensible to the user and contained no mention of the original query items at

all. The advent of the Internet and the World Wide Web caused a shift of focus towards a less relevance oriented paradigm moving the focus over a more performance oriented approach, causing a narrowing of search relevancy, as the task became that of navigating the user to the one document they seek, instead of that of precisely representing its interest. This caused search engines to adopt the second definition, as more experimental techniques went entirely missing from the scene. Thus, as long as the result “made sense”, the search engine is considered to be properly behaving and the result is regarded as being relevant. In order to measure if a search engine is returning relevant results there can be used several methods, either relying on user engagement measures the like of Click Through Raters (CTR), Conversions, Abandonment Rates or human reviews.

More likely, in the future, relevance of content will be evaluated through on a whole different new level with respect to the two approaches already listed. As search engines have already started providing their result based on custom user values such as their search history and their world position, it is very likely that relevance will evolve to mirror these changes as well, by becoming a somewhat malleable concept: a result will be relevant if it is relevant to that particular user, and not in a broader sense anymore.

1.2.2.5.1. Popularity

It is important to underline the very basic notion that, the popularity of a document often doesn’t mirror its relevance to the user query or that the document is associated with trustworthy sources. More often than not, in fact, search engines returns more mainstream results, in hope of satisfying a wider audience but, by doing so, they sacrifice accuracy - as already stasted in the previous paragraph about **relevance**. However, popularity of results still retains its importance as a distinctive factor when deciding which result to show first. Popularity is often associated with the concept of a page’s “received votes” or a page’s “authority”, notions stemmings directly from PageRank inventor and Google’s founder Larry Page and HITS inventor Join Kleinberg respectively. Popularity of web pages as “received votes” will be discussed in *1.2.2.8. Page rank: at Google’s foundations*, while popularity as “authority” of a web page will be further explored in *1.2.2.9. Hits*.

1.2.2.7. Search Engine Ranking Factors

Web-page ranking is an algorithmic technique used by search engines for ranking hundreds and thousands of web pages in a relative order of importance. To rank a web page different criteria are used by ranking algorithms. For example some algorithms consider the link structure of the web page while others look for the page content to rank the web page. Broadly Page Ranking algorithms can be classified into two groups Content-based Page Ranking and Connectivity-based Page Ranking[12].

[12] Comparative Study of HITS and PageRank Link based Ranking Algorithms: Pooja Devi, Ashlesha Gupta, Ashutosh Dixit, 2014

- **Content based page ranking**: in this type of ranking procedure, the algorithms takes into considerations the actual document's content. In this case, the factors influencing the page ranking can be summarized in the following list:
    - The number of words that are a match for the query terms
    - The frequency at which the exact query string appears in the document
    - The locations of the query terms. For example, the document headline and titles are very important areas
- **Connectivity based page ranking**: is grounded in the link analysis techniques oriented algorithms. These type of algorithms models the World Wide Web to be a directed graph and have been widely used. Amongst the link analysis algorithm there can be found:
    - PageRank
    - HITS

1.2.2.8. Page rank: at Google's foundation

Google's PageRank is the algorithm used by the search engine to rank pages, by determining their importance. PageRank is named after its primary inventor, Larry Page, who, alongside his colleagues at Stanford University (most notably, Sergey Brin), founded the company in 1998. Before dwelling into the details of the PageRank algorithm, it's needed to model the World Wide Web in a way that could be exploited. The World Wide Web can, in fact, be depicted as a directed graph, where web pages represents nodes, while links between them are representant of edges.

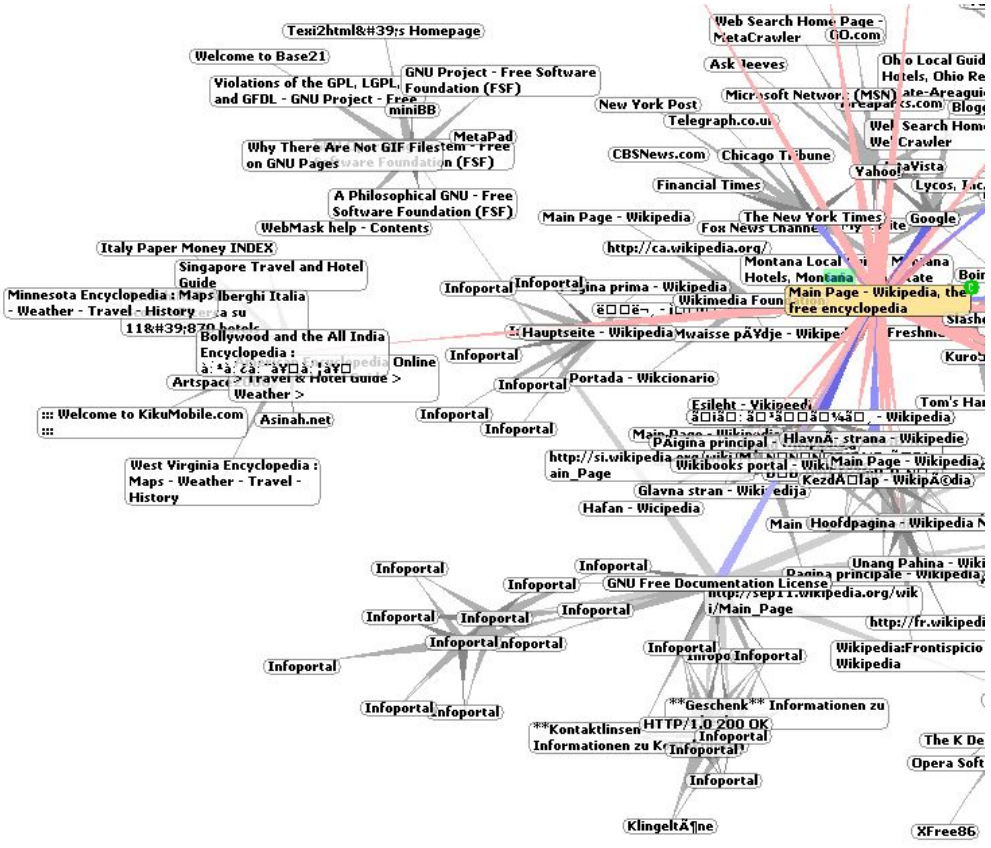


Image: portion of the World Wide Web as a directed graph [13]

[13] Graph drawing, http://www.wikiwand.com/en/Graph_drawing

The idea behind PageRank is at the same time simple and innovative. By looking at the graph structure and how pages are connected together via links, one of the first very noticeable things is that page linked to by other pages should somehow hold a certain importance in the pages linking to them. This concept goes even further by assessing another thing as well: creating a link to a page assesses its importance and, many inbound links to the page makes it generally renowned as an important page. On the other hand, if the page possessed only one link, but coming from an authoritative source, the page would receive an high authority value as a consequence. This is an implementation of the concept of **popularity** examined in previous chapters. This concept was developed in order to deliver better search results that would also be resistant to marketing attempts of deceiving search engines by creating web pages containing very popular keywords in order to appear first. Before seeing the formula in details it is worth mentioning the model behind it. In fact, the justification for the PageRank algorithm involves the so-called model of a "**Random Surfer**", modelling user behaviour as a random click of links with no regard towards content. The Random Surfer model captures the probability that a user will visit a certain page, depending on its PageRank, while the probability that the user clicks on one of the links is simply given by the number of links in the page. In short, this means that the probability for the random surfer to reach the page is the sum of all probabilities to reach the page by following links while this probability is also reduce by a random factor called the "damping factor". In the **Random Surfer** model, the damping factor is equivalent to saying that the user will not click indefinitely on the provided links but will sometimes be willing to change and choose other pages randomly.

The original PageRank algorithm has been described by Page and Brin in several papers and can be computed by solving the following formula:

$$PageRank\ of\ site = \sum \frac{PageRank\ of\ inbound\ link}{Number\ of\ links\ on\ that\ page}$$

OR

$$PR(u) = (1 - d) + d \times \sum \frac{PR(v)}{N(v)}$$

equation: page rank's calculations [14]

The above formula sintetize the **Random Surfer** model by taking into account the **damping factor d** as well as the probability of the user to click on all previous links leading to the document. It has to be noted that d is a probability value and, as such, varies between 0 and 1.

[14] THE MATHS OF GOOGLE, https://mathsbyagirl.wordpress.com/2016/01/29/the-maths-of-google/

Whereas this is the theoretical formula behind Google's success, its implementation is an actual approximation of it, given the size of the World Wide Web. As a matter of fact, each page is assigned an initial PageRank value and the PageRank of all pages are subsequently calculated. A good approximation of the PageRank value is only reached after a few iterations and, according to the author's publications, about one hundred iterations are needed in order to return a good approximation of the PageRank for the whole World Wide Web.

The following lists will try summarize the advantages and disadvantages of the PageRank algorithm:

- **Advantages**:
    - Low query time: PageRank is able to compute ranking at crawl time, making it very quick and efficient
    - Low susceptibility to localized links: given that PageRank theoretically computes ranking on the entire World Wide Web graph, it is less susceptible to localized links
    - Feasibility: PageRank algorithm is a feasible algorithm in today's scenario since it performs computations at crawl time
- **Disadvantages**:
    - User query relevancy: PageRank does not take into account the document relevance with respect to the user query
    - Rank "sinks": rank sinks occur for pages that have no outbound links, thus "sinking" the algorithm. These problem has been addressed in later versions of the algorithm.
    - Staticicity: as a static algorithm, PageRank suffers from a very simple problem: popular pages will tend to stay popular, while popularity of a website is not guaranteeing an equal query relevance.
    - Data amount: the amount of data in the Internet is vast and algorithm, in its non approximated form, simply cannot perform well enough
    - Spider traps: which are group of pages with no outgoing link from within the group to outside of the group
    - Dangling links: occurring when a page contains a link to a page with no outgoing links
    - Circular references: circular references in a website reduces the website front page's PageRank.

1.2.2.9. Hits

A popular alternative to Google's PageRank algorithm, the **HITS** algorithm by Jon Kleinberg, a professor at Cornell University, was being developed at the same time as Page and Brin were working on their trademark algorithm. The **HITS** algorithm, which is a shorthand name for ***hyperlink-induced topic search***, is implemented in the Ask search engine and is regarded as one of the more complete link analysis algorithm.

The algorithm was developed in order to propose a simple way to rank a page with respect to a user query, by proving the page to be **authoritative** in the context of the query. A page is said to be an **authority** if it contains valuable and trustworthy information towards the query subject. Generally speaking, these are the pages that the user would expect the search engine to return. However, the algorithm models the World Wide Web even further, by defining another category of pages, namely, the **hub** pages. These are pages that contains links to the authoritative pages, pointing the search engine in the "right direction". It is worth noting that:

- Good authoritative pages (**authorities**) and good hub pages (**hubs**) reinforce each other
- A page can be both a good **authority** as well as a good **hub** page

For every page, the **HITS** algorithm identifies an authority and an hub weight where a page holds a higher authority weight if being pointed to by pages with high hub weights. This is also true for pages with high hub weights, which points to pages with high authority weights. Given a set of query related pages, the algorithm tends to form a bipartite subgraph of the World Wide Web as depicted by the following image:

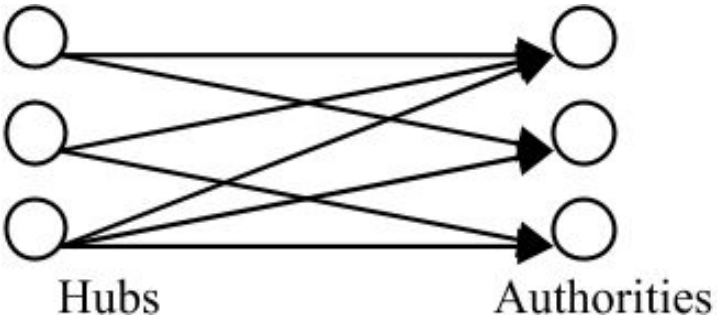


Image: hubs and authorities[15]

In the **HITS** algorithm, ranking of a web page is computed by an analysis run on the resulting subgraph obtained from the resulting set of documents with the highest occurrences of the query terms. A brief explanation of the algorithm can be given as follows: the initial result set, usually called a "base set", is set to contain an heterogenous set of pages with few links to each other. The base set is extend by including all edges coming from or pointing to the set.

[15]https://www.semanticscholar.org/paper/An-Improved-HITS-Algorithm-Based-on-Page-query-Sim-Liu-Lin/e7e182659614da4f92daca8d8455fd11350f198a

This operation yield the subgraph that will be analysed by the **HITS** algorithm. This graph is very likely to contain **authoritative** sources for the query and, from here on, the **authority** and **hub** scores for each page belonging to the set are computed following a weight initialisation. The final **authority-hub** scores are obtained after “infinite” iterations of the algorithm and need to be normalized to avoid divergency.

The following lists will try summarize the advantages and disadvantages of the HITS algorithm:

- **Advantages**:
    - Highly relevant to the query: the ranking measure given by assigning **authority** and **hub** weights is highly relevant to the query
    - Ranking can be combined: the ranking measure can be also combined with other informational retrieval based rankings
    - Query sensitive: is query sensitive, meaning that the page ranking will always take into account the relevancy to the user query
- **Disadvantages**:
    - Query time inefficient: as HITS calculates page ranks at query time it takes more time to compute a response
    - Irrelevancy of hubs: it is a situation occurring when a page contains links to a large number of separate topic, receiving in turn a high **hub** rank without being equally relevant to the user query
    - Mutuality: mutually reinforcing relationships between hosts can also be used to “spam” the algorithm
    - Topic drift: is an issue occurring when irrelevant pages in the initial set are strongly connected, yielding to irrelevant pages in the base set as well
    - Low feasibility: as HITS computes the pages rank values at query time it is less feasible for today’s search engines with respect to its others counterparts

In general, it can be said that the **HITS** and **PageRank** algorithms share similar principles by making use of the link structure of the World Wide Web graph in order to compute page’s relevance score. The main differences can be summarized by the fact that, differently from the PageRank algorithm, HITS only operates on a small subgraph. Furthermore, HITS is query sensitive while PageRank is not. In either cases, the highest ranking pages are displayed to the user by the search engine.

1.2.2.10. TF-IDF

TF-IDF is the shorthand name version of “Term Frequency - Inverse Document Frequency” and is often used in the areas of Information Retrieval and Text Mining to compute the importance of a word to a document in a collection of documents. This weight is a statistical measure increasing proportionally to the times the word appears in a document but that it is limited in its growth by the frequency of said word in the collection. Uses of **TF-IDF** often includes:

1. Basic ranking of functionality, computed by summing the **TF-IDF** weight for each term of the query
2. Stop words identifications

TF-IDF is usually employed by modern web search engines in conjunction with ranking algorithms such as PageRank and HITS to retrieve a first collection of documents that matches the user query and upon which to run the ranking algorithms.

In the field of search engines **TF-IDF** is thus not a single method but a collection of techniques that aims at comparing query terms and documents by providing a way to determine the matching. In the case of search engines, **TF-IDF** can be used to summarize a document using a few meaningful keywords. A more simplistic approach would involve simply counting the terms, but that would be error prone by including non meaningful terms, like, for example, stop-words. The **TF-IDF** approach naturally deals with such problems ad is able to identify good keywords candidate amongst those contained in the document (i.e. those with a high **TF-IDF** score). The computation usually involves two terms: the normalized Term Frequency (**TF**) and the Inverse Document Frequency (**IDF**):

- **TF: Term Frequency**, measures how frequently a term occurs in a document. The number of times a term appears in a document is normalized due to the probability of the term appearing more often in longer documents with respect to shorter ones. The computation of the Term Frequency of a term *t* for a certain document is then computed as such: TF(t) = (Number of appearances of term t in a document) / (Total number of terms in the document).Typically, the more often a term occurs in the document, the larger its TF coefficient.
- **IDF: Inverse Document Frequency** measures the importance of a term. This is particularly useful when facing stop-words and extremely recurring (but, non important) terms. This terms may appear a lot of times, thus a way to weight them down with respect to the rarer ones is needed. Conversely with respect to the TF coefficient, IDF is bigger for terms that are rarer across different documents. The coefficient can be obtained by computing the following formula: IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

It is worth noting that **TF-IDF** differs from **Sentiment Analysis**, despite being somewhat often used in combination with it and both algorithms being regarded as text classification techniques. While **TF-IDF** concern itself with expressing the importance of words in a collection of documents, sentiment analysis is used to classify the text based on its “sentiment”, being it negative or positive.

1.2.2.11. Beyond the algorithms: influencing factors

Apart from the famous core algorithms which have been explored in the above paragraphs, there are a number of other hidden ingredients to the secret receipt that seem to hold a variable importance to the ranking of pages from year to year. These factors are subject to continuing speculations by Web Marketer that often try to exploit them in order to better their page's ranking. An estimation of the amount of different factors contributing to Google's PageRank algorithm is often said to amount to up to two hundred different parameters.

A bi-annual survey conducted by Moz.com is an attempt at closely modelling the changes in the PageRank additional factors, and the most recent version, dating back to 2015, lists over 90 different ranking factors in the following categories[16]:

- **Domain-Level, Keyword-Agnostic Features**: Domain name length, TLD extension, SSL certificate, etc.

- **Domain-Level, Link Authority Features**: Based on link/citation metrics such as quantity of links, trust, domain-level PageRank, etc.

- **Domain-Level Keyword Usage**: Exact-match keyword domains, partial-keyword matches, etc.

- **Domain-Level Brand Metrics**: Offline usage of brand/domain name, mentions of brand/domain in news/media/press, toolbar/browser data of usage about the site, entity association, etc.

- **Page-Level Social Metrics**: Quantity/quality of tweeted links, Facebook shares, Google +1s, etc. to the page

- **Page-Level Link Metrics**: PageRank, Trust metrics, quantity of linking root domains, links, anchor text distribution, quality/spamminess of linking sources, etc.

- **Page-Level Keyword & Content-Based Metrics**: Content relevance scoring, on-page optimization of keyword usage, topic-modeling algorithm scores on content, content quantity/quality/relevance, etc.

- **User Usage & Traffic/Query**: Data SERP engagement metrics, clickstream data, Visitor traffic/usage signals, quantity/diversity/CTR of queries, both on the domain and page level

- **Page-Level, Keyword-Agnostic Features**: Content length, readability, Open Graph markup, uniqueness, load speed, structured data markup, HTTPS, etc.

These fluctuation of parameters is not relative to the above items only, but also to the links - which are at the very base of the link-analysis algorithms - as well. In fact, ".. From our

[16] Search Engine Ranking Factors 2015, https://moz.com/search-ranking-factors/survey

experimental data, we could observe that the top 20% of the pages with the highest number of incoming links obtained 70% of the new links after 7 months, while the bottom 60% of the pages obtained virtually no new incoming links during that period…"[17], but that, more importantly, "…The link structure of the Web is significantly more dynamic than the contents on the Web. Every week, about 25% new links are created. After a year, about 80% of the links on the Web are replaced with new ones. This result indicates that search engines need to update link-based ranking metrics very often…"

1.3. With many data, come great possibilities

The Internet and the World Wide Web has given us the ability to obtain an incredible amount of information from many different sources. Search engines usually returns up to ten pages of results for very broad queries, amounting for more data than most of the casual web surfers might need to use. As a matter of fact, for the casual users, these data often come in overwhelming quantity, but to the data scientist, they can actually amount to an incredible gold mine of useful information. For example, it has, in recent years, been proven that the careful examination of user tweets from the famous website Twitter can actually predict many things, amongst which it is of particular relevance to cite a recent study, which claims the unusual ability (for a website) to "predict riots faster than the police"[18]. The era of social networks has also opened the gates for the discovery of trends in the collective opinions of people interest in a specific subject, being it of current or past relevance. As a result, people behavioural patterns and thoughts have become somewhat more open to the inspection of government agencies, marketing experts, web criminals and such. All of these people uses Data Science to scrape the content of the web looking for the kind of information that can be profitable for their business.

A brief list of the information that is possible to get by scraping internet data can comprehend:

1. A local, static copy of any existing web site
2. User opinions from social media
3. Newspapers articles published on the web
4. The currently most trending topic

The above list contains some interesting entries. In particular, the possibility to obtain information about the Web Reputation of some topic of interest seems to be extremely interesting. Such topic could, for example, very well be a person. Web Reputation is in fact often offered as a service by many computer science firms, in order to improve and manage the reputation of a juridical entity (e.g. a persona, a firm) on the web.

[17] J. Cho, S. Roy: Impact of Web Search Engines on Page Popularity In Proceedings of the World-Wide Web Conference (WWW), May 2004.
[18] Twitter can predict a riot faster than police, https://www.theverge.com/2017/6/28/15882272/twitter-riot-predict-faster-than-police-cardiff

1.3.1. Meaningful results: web reputation

Web reputation is the opinion of the Interned towards any kind of opinion susceptible topic of relevance. It is an assets to many interested parties and is discussed to more extend in chapter 3. Web Reputation.

1.3.2. What does the “query” say

Running a query on a search engine amount to asking a cluster of servers, located in a web farm somewhere in the world, to run an algorithm in order to retrieve the most related lists of web pages that could satisfy the query itself. What is actually returned might slightly vary from search engine to search engine, but usually consists of – at least – the following basic elements:

1. A list of pages (of course)
2. A brief description of each page in the list
3. A clickable anchor pointing to the page's location in the web

These results pages are usually known as **SERPs** (Search Engine Results Pages). The mesmerizing thing about them is that each one is unique, even upon returning results for the same query. This mostly occurs because of search engines results customization, aimed at providing their user with a more customized experience, depending on factors such as user’s location, browse history and social media activities. **SERPs** are usually also presented with varying graphical setups, due to the search engines constantly experimenting with them.

SERPs usually consists of different kinds of listings that are common to all commercial search engines, such as Google, Bing and Yahoo! . In particular, they may show a combination of the following:

1. Search results indexed by the search engine’s spider
2. Search results added to the index by a human
3. Paid results, that are paid for to be listed on the page

It is possible to further divide these listings into two groups. The first group contains the first two items of the list and is called “**Organic results**”. The second group consists - of course - of the remaining list item, the group of “**Paid results**”.

SERPs might display a different amount of organic results, depending on the type of search performed by the user. If the user is looking for just plain informational subjects, the result page will return more organic results with respect to other type of search, the reason being: an informational search consists of a request of information regarding a topic with no commercial value at all. On the other hand, queries consisting of keywords indicating the intent of completing some kind of monetary transaction are very likely to include paid results in their **SERPs**, with less organic results being displayed in the page as a consequence.

It is very important to point out that most people never go past the first page of results when looking for a search result. This has dire consequences on both search engines and web marketers. Where the firsts strive to return the right answer to the user query in the first top three results of the first page, web marketers desperately try to achieve greatness for their client by pushing his / her content into the first page of results.

1.3.3. An interesting question: is it possible to obtain more information from the returned result set?

As previously examined, **SERPs** contain short amounts of very valuable information regarding each site that matched the user query. The information can be exploited to, for example, gain an insight on a company's competitors Internet presence, or, more interestingly to examine the results set by looking for different kind of indicators. These indicators can be divided in two distinct groups:

- **Unexpanded SERPs items**: indicators that can be deduced by analysing the resulting **SERPs** in detail by not visiting any of the resulting urls
- **Expanded SERPs items**: indicators that can be deduced by expanding the **SERPs items** even further, by analysing each page's content, reached by following the provided urls

These two categories differ primarily in computational time, and amount of information yield. More specifically, **Unexpanded SERPs items** can be computed at crawls time without the need to spend additional computational power, while the **Expanded SERPs items** obviously need to be separately analysed and are subject to pre-processing operations that can hinder the time needed to provide a response. The **Unexpanded SERPs items** indicators are:

- **Social networks**: the presence or absence of social networks in the **SERP** is an indicator of a good social media activity and thus an actively discussed topic
- **Related searches**: the related searches list can also suggest the kind of user searches usually linked to the query, providing additional information such:
    - The level of interest in the topic, approximated from the quantity of available suggestions
    - User opinions on the topic, approximated by analysing the additional query items provided in the list
- **Types of results**: the different types of results might indicate a good (or bad) media coverage, depending on the presence (or absence) of media results such as images and videos
- **Result positioning**: the varying page ranking of a document inside different **SERPs** obtained by running the same query over different period of times can provide indications such as:

    - A drift in users search interests
    - Stronger competition
    - Area of improvement

These general indicator can provide a first partial view of both the user's opinion on the query subject as well as some broader indications concerning the type of content to be added in order to ensure a better **SERPs** positioning and a larger user base reached.

However, to get into even more detail, the **Expanded SERPs items** need to be taken into account. These indicators include:

- **Sentiment of document main corpus**: the sentiment of the central document text can be deducted from the text by means of sentiment analyser to derive an estimation of the document sentiment
- **Influence over query subject**: the influence of the text on the query subject can be regarded as the impact (positive or negative) that the text can have on the query subject

These indicators can provide very valuable information linking to the "opinion of the web" towards a certain subject, being it a person, a company or a brand. Summarizing, the above two groups of indicators yield the possibility to gain knowledge of how the World Wide Web's opinion is directed and on its fluctuations over time, with the resulting data ideally summarizable in a graphical manner through charts and tables. This data, combined in a tool to execute and store information, could provide every entity interested in its Web Reputation management with the needed insight.

The data collected in such a way are part of process of gathering information from textual sources on the web, called "Text Mining" which can be regarded as a subset of the broader set of techniques known as "Data Science".

2. Data Science: a brief explorations of the principles behind data

In the nineteenth century the human society was entering the so called Industrial Revolution, exploring its real potential side by side with great mechanical innovations. The most famed entrepreneurs of this era were able to spot a great deal of potential in the technology of the time and this ability was fundamental for the incredible wealth they were able to amass thanks to their factories. The Industrial Revolution had its shares of positive and negatives sides and, while the technology reached a peak that allowed engineers to produce better machineries, the next logical step was that of producing smaller and faster technology. Thus, the industrial era had come to an end as the world was entering the so-called **Information era**. Researchers started to gather and analyse data in order to better their understanding of humanity, the world and the whole universe. As with the Industrial era, the Information era also featured positive and negatives consequences on the world. On the good side of things, the technological innovations had a great positive impact on society. However, on the bad side of things, the world started being overflowed with information due to researchers continually pushing to gather more data to analyse. Of course, data is not generated for the sake of research only. With every tweet, posts and document, Internet users contribute to the growing mass of data by creating an unprecedented amount of information. The flow of data travels both ways: people are not only creating information at incredible rates, they are also consuming it at an ever growing rate. What is being seeked today is knowledge, that one piece of information that can add to the puzzle and solve whatever issue at hand. Researchers now have an enormous set of data, powered by machines capable of harvesting information 24/7. The remaining problem, is to make sense of the information as such data could be used to make predictions and model reality. This is what data science seeks out to do, by using a scientific approach to solving the problem[19].

2.1. A scientific approach to data analysis

Data science can be viewed as a combination of logical steps to help gaining a better understanding of the data. As explained by Ozdemir S. in his book "*Data Science. Guida ai principi e alle tecniche base della scienza dei dati"*, there are several types of data and levels as well as different steps involved in the approach to data analysis with Data Science. The following paragraphs will introduce said concepts in this work.

2.1.1. An introduction to Data Science

It is most useful to define what "data" is in the first place. Data can be defined as a collection of information that can be in either an *organized* or an *unorganized* fashion.

[19] Ozdemir S. (2017), *Data Science. Guida ai principi e alle tecniche base della scienza dei dati*, Apogeo.

- **Organized data**: represents information usually organized in a table like structure (i.e. with columns and rows) where each row is a single “observation” while columns represents its “characteristics”
- **Unorganized data**: are data characterized by having no formal structure, usually present in text, audio clips and other formats. This type of data need to be better analysed in order to gather any useful information and often need to be turned into **organized data** to be properly handled by software

Data science, then, is simply the science of making sense of data by extracting information to gather valuable knowledge to be used for:

- Decision making
- Future projections
- Gaining a better insight of past and present events
- Creating new products and services

In order to achieve any of the above feats, data need to be *acquired*, *examined* and *transformed*. The hidden information can then be used by Data Science to find new hidden meanings that would otherwise remained concealed.

2.1.2. Data types

As already mentioned, there exists different types of data. In particular, it is possible to summarize a list of the usually found data types:

- Organized and unorganized data
- Quantitative and qualitative data
- The different data levels

Different data types do not only determine the kind of methods used to to analyse and extract the information but can as well tell the researcher a lot about the kind of setting they are attempting to model. Understanding the type of data is a very important task as without any real understanding of the nature of the data the models and techniques applied might not be suitable for the case at hand.

2.1.2.1. Organized and unorganized data

Organized data are much easier to make sense of and compute and are usually *organized* in a table like structure. Most models were conceived to work with this type of data and does not usually behave as expected when dealing with the aleatory nature of **unorganized data** as machines - as humans do - are better at understanding structured information. Nevertheless, a great amount of data is of the unorganized type. Most estimations actually indicate that up to 80 to 90 percent of data worldwide is *unorganized*. These data exists in different formats and are usually underestimated as they are not a clear source of information - at least, not at first sight. However, this is simply due to the intrinsic nature of **unorganized data**, that are trapped in a “free” format that doesn’t allow for a simple insight

on the information hidden within the data. Data Science is then often charged with the task of employing pre-analysis and pre-processing techniques to - at least - partially structured data, so to be able to proceed with further analysis and elaboration.

The pre-processing of data is used to transform data in a format that could be used by a model and to extract new characteristics starting from existing features. Given a simple text, it is possible to, for example:

- Count words / statements
- Spot special characters
- Compute its length
- Extrapolate the main topic

These simple characteristics can then be organized in a table, thus becoming **organized data**.

2.1.2.2. Quantitative and qualitative data

Distinguish between qualitative and quantitative types of data probably is the most common way to describe a data set. Most often than not, when referring to quantitative data, the data set is probably structured (i.e. **organized**) but that can also not be the case and this is way the pre-analysis step is important in marking a distinction. These two types of data can be specified as:

- **Quantitative data**: can be described by numbers and thus support the ability to perform mathematical operations on them
    - Continuous data: these are data that need to be measured by some kind of metric. They can have an infinite array of values
    - Discrete data: the are data that can be counted and thus have only determined - finite - values
- **Qualitative data**: cannot be described by numbers or simple mathematical operations. On the contrary, they are usually described with the use of natural language

2.1.2.3. The four levels of data

When dealing with **organized data** it is possible to focus the analysis on the features of a specific entry. Its characteristics can then be examined through the four levels of data, that are the following ones:

- Nominal level
- Ordinal level
- Intervals level
- Ratio level

This framework originates in psychology and was created by psychologist Staley Smith Stevens in an attempt to “describe the nature of information within the value assigned to variables”[20]. By use of the definition presented in the next paragraphs is possible to further structure data and to obtain more information as a direct consequence. It is important to note that, as shown below, different kind of mathematical operations and functions are available for each level of data. Moreover, each level inherits the mathematical operations defined at preceding ones.

2.1.2.3.1. Nominal level

The first level of data is called “nominal” as in “name” and is characterized by data that can be described by name or category only. These categories hold some distinctive traits, in particular:

- cannot be nor ranked or compared quantitatively
- feature a mutually exclusive characteristic as one item cannot be a member of two categories at the time
- they are also exhaustive, meaning that there is a category for any possible case

The most notable examples of data belonging in the set are data about gender and nationality. These data are not numerically described and are - therefore - qualitative data.

- **Mathematical operations**: as the data in this level are of qualitative nature, most mathematical operations would have no meaning when confronted with this typology of data. The only mathematical operations that can fit into this level are the equality between items and the membership of an item to a set of items.
- **Central tendency**: is a measure needed in order to find an element describing the tendency of values towards a specific “central element”. More often than not, it is possible to describe this tendency by use of mathematical average, median or moda values. When faced with data at the nominal level, however, it is customary to use the moda parameter to find the tendency of the dataset. This value will then be the item with the most occurrences. Other ways of finding the centre of data, as the mathematical average and median values, cannot be used in this level for its missing support for mathematical operations.

At the nominal level, data usually describe categories and, as they are described by means of natural language, there might occur errors. Furthermore, only having the moda value describing data tendency the additional information that can be inferred from the original set are very limited.

[20] Kirch, Wilhelm, ed. (2008). "Level of measurement". *Encyclopedia of Public Health*. **2**. Springer. pp. 851–852. ISBN 978-1-4020-5613-0. doi:10.1007/978-1-4020-5614-7_1971. Retrieved 14 August 2015.

2.1.2.3.2. Ordinal level

While at the nominal level of data there is no possibility to order items in any natural way, it is not possible, within that level, to to perform many mathematical operations. At the ordinal level the variables are still classified into categories. Categories at this level also retains the features of mutual exclusiveness and exhaustiveness but also feature an additional characteristic:

- categories have a logical order that allows for ranking

Nonetheless, the scenario already changes for data at the ordinal level which can surely be ordered. Nevertheless, while it is possible to order the single items, it is not possible to sum or subtract them yet.

- **Mathematical operations**: while inheriting the mathematical operations of the preceding level, the ordinal level makes the following available:
    - Ordering: it simply is the natural ordering of data. It can sometimes be tricky to find, depending on the data being described
    - Comparison: makes it possible to compare two elements and to define which one is better than the other
- **Central tendency**: at this level the median value is normally used to define data tendency towards a central value.

2.1.2.3.3. Intervals level

At this level data starts to display more interesting features as they can now be expressed by means of more advanced methods allowing for more mathematical operations to be featured in this level. At this level, data are still organized in categories with the following additional features adding to the ones at previous levels:

- equivalent distance between categories

Moreover, what sets this level of data apart from the previous ones is the possibility to express to use the mathematical function of subtractions between elements.

- **Mathematical operations**: at this level all operations allowed in previous ones are inherited, while new operations can now be employed:
    - sum: it makes sense to be able to sum two or more elements of a dataset belonging to this level of data
    - subtraction: it makes sense to be able to subtract two or more elements of a dataset belonging to this level of data
- **Central tendency**: at this level the most appropriate measure used to find the central element is the arithmetic mean as it is now possible to express the mathematical sum
- **Measuring data variability**: it is important to also be able to track the dispersion of data. The most common measure of data dispersion is the standard deviation which can be intuitively defined as the “medium distance from a value with respect to the

mean". Standard deviation alongside the arithmetic mean allow to describe a dataset with just two plain numbers

Furthermore, data at this level does not have a real zero point. On the contrary, the zero point is totally arbitrary, making it nonsensical to multiply or divide elements.

2.1.2.3.4. Ratio level

This data level stands out as the most complete and solid data level of the list. In fact, it inherits all the features of previous levels.

- **Mathematical operations**: at this level the division and multiplications of elements are added to the supported operations
- **Central tendency**: arithmetic mean maintains a meaning at this level but the *geometric mean* provides a more suitable way of measuring the central tendency of the data set
- **Measuring data variability**: at this level, measures of data variation such as the *studentized range* and the *standard deviation* are more commonly used

In this case, the presence of a real zero point ensures the possibility to perform the multiplication and division operations as well as stating a simple fact: a value of zero actually means the absence of value.

2.1.2.3.5. Summary

As clearly deductible from all the observations on the differences between the different data levels of the framework it is of clear importance to clearly define the data set at hand by choosing in which of the four level the analysis should operate. This is needed because of the different kinds of mathematical operations allowed at each level:

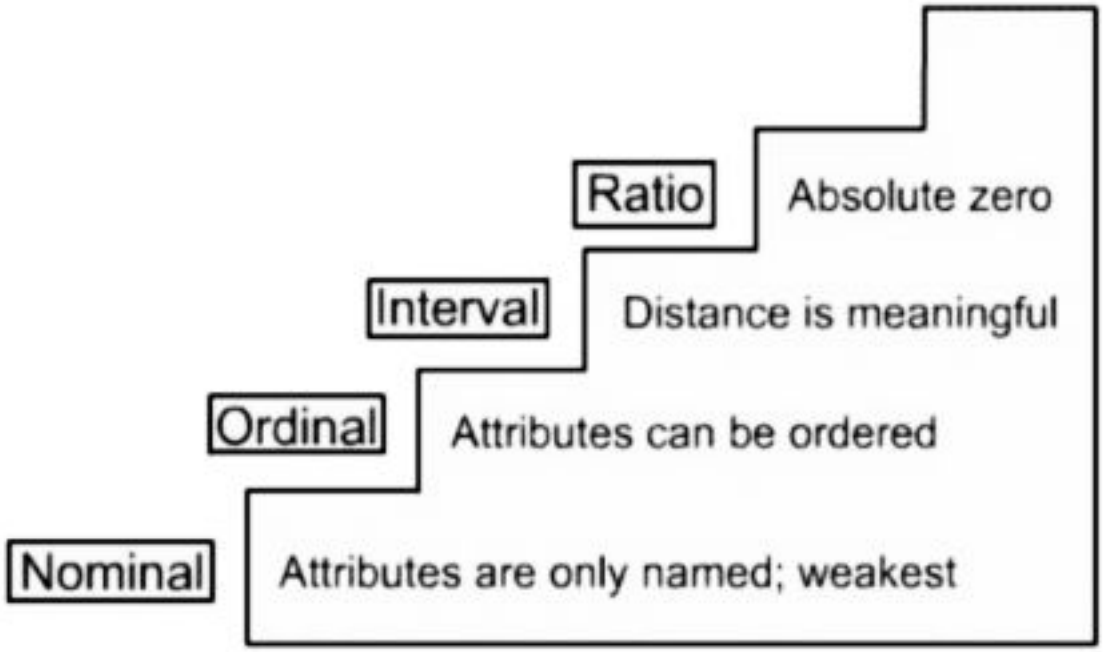


Image: levels of measurements[21]

An erroneous evaluation of data could mean an over imposition of a structure that doesn't match the true data level thus causing errors when applying unsupported operations.

2.1.3. Data science: a five steps scientific approach

Data science follows a structured process organized in distinct steps to facilitate the analysis and ensure data integrity. The five steps approach of data science involves:

1. **Posing an interesting question**: this is probably the most fundamental step, as without an interesting question there would be no problem to solve and no data to exploit. An interesting question is such that might create new value but should also have a feasible nature. Some questions could involve gathering data in both the public and private sector and that could not be feasible in some cases. As a consequence of what is asked in the aforementioned question, the problem outline should aldo be framed in order to translate the request into a well defined problem
2. **Fetching data**: fetching the data from reliable sources is the next logical step and requires identifying where to extract the information from and through which means.
    a. If data are on the web then they are most likely free for anyone to use and web scraping techniques could be employed to obtain data
    b. if data are of private nature a request to examine and use them should be made to the owner of the data set which could in turn refuse to accommodate the requests

[21] Statistics – Understanding the Levels of Measurement, http://www.kdnuggets.com/2015/08/statistics-understanding-levels-measurement.html

The process of fetching data would also require answering a few questions on its own. For example: What parts of the date are useful? What kind of resources are needed in order to collect such data?

3. **Exploring data**: at this point data should be categorized in order to define their type (i.e. level of data they belong to) and their domain should be studied in order to gain a better understanding of their nature. This step involves identifying data as well as transforming it into different types in order to better prepare it for the next step (i.e. Modelling). In fact, raw data is very rarely usable out of the box as there might be missing or corrupted information in the collection, this is the reason why pre-processing steps as the one mentioned in the previous paragraph should be taken
4. **Modelling data**: this step involves finding and applying the right statistical, machine learning models or algorithms to data to uncover the information hidden in the raw data
5. **Presenting findings**: the final step involves presenting data in a simple and clear way to the user, as all the analysis would otherwise prove meaningless to stakeholders

The following analysis will go into more details and will attempt to illustrate each step in more depth.

2.1.3.1. Posing an interesting question

This step is similar in many ways to the software engineering step of understanding the problem as well as gathering the user requirements. However, at this stage, the real challenge is in understanding the data science problem behind a user's request. The process involves the tedious process of sitting with a customer in order to draw as many information from him, so to be able to define the problem. Once the data science problem have been agreed between parties, it is very important to refine the analysis in order to prioritize the questions that the client finds to be the most desirable to be solved. Following this, it's time for yet another question: where to gather the data from?

2.1.3.2. Fetching data

Being able to get the data requires access to a data source of some kind, be it a database or the web itself.

- **Web scraping**: is a set of techniques whose final outcome is a set of data referring to a specific topic. They often makes use of search engines to find suitable documents from which to extract information from and are somewhat debated as belonging to a gray moral area as some experts see them as being borderline illegal if used for the wrong endings. Data extracted in such a way are usually **unorganized** and difficult to understand without any pre-processing

- **Databases**: on the other hand, stores data in a very **structured** way and often have a more complete data set that what could be obtained by merely scraping the web's surface. Data extracted from databases, however, still needs to be pre-processed to some extent

At this stage it is important to choose what data is actually needed by the application and in what format it should be extracted from the data source. If data is related to personal information it also important to anonymize them so that they don't bear any reference to the entity they refer to in the original source.

2.1.3.3. Exploring data

After the previous stage, the data that was collected from the source(s) are in the raw form and cannot yet be employed to obtain valuable information. Raw data should be explored in more details in order to:

- to better the understanding of each item by examining further with the client in order to ensure that the value is not being misinterpreted
- to uncover missing or malformed data that could harm data integrity and to decide what course of action to take when confronted with said issues (e.g. malformed records could be simply thrown away or default values could be used instead, to replace missing or corrupted ones)
- to define a data type, by choosing between the different available levels of data in order to uncover the available operations and transformation on data at the specific level
- to check that all the needed information has been gathered

At this point, after cleaning the data up it is time to explore them with a different objective in mind: what significance does this data set hold with respect to the original **question**? Which part of the data is significant in answering it? The answers to these questions can be better gathered by transforming data into an **organized** data set from which to infer more information. This last pre-processing step leads to data modelling.

2.1.3.4. Modelling data

Data modelling makes large use of both *statistics* and *machine learning* techniques in order to create a model for the data to accurately predict or analyse data.

> **Def 2.1.3.4. (model)** *a **model** is a simplified representation of reality - or, in this case, of data. More precisely, a model attempts at mathematically describing data by acquiring inputs and producing outputs based on the data that was previously collected.*

In particular, it can be said that "models are implementations of theory, and in data science are often algorithms based on theories that are run on data. The results of running a model

lead to intuition, i.e., a deeper understanding of the world based on theory, model, and data." [22]

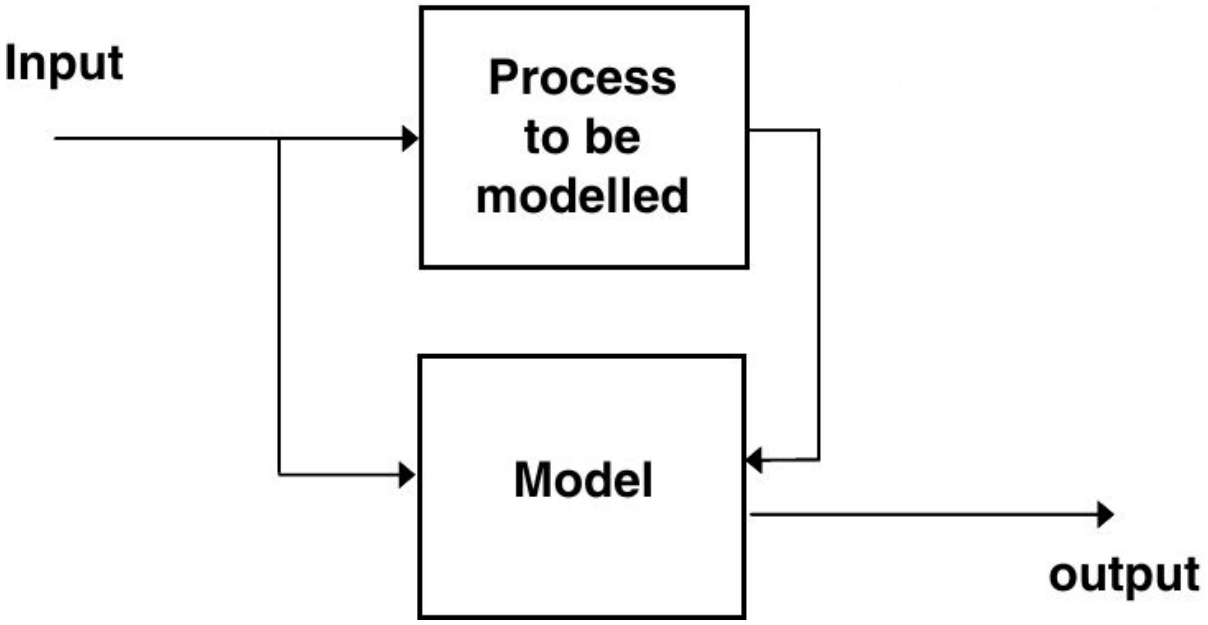


Image: the process of modelling data and returning a result

The choice of a model to represent data has to be done by understanding the available techniques to better describe the reality analysed by the task. In general, either the statistical or machine learning approaches provide with enough modelling power to create a functional project model depending on the focus, whether it's a "optimization and performance first" approach (i.e. machine learning) or an "inference first" approach (i.e. statistics). In both cases, the first step in building a model is deciding which algorithm to use. Below, some of the commonly used algorithms:

- **Decision Trees**: builds classification models in the shape of a tree like structure by breaking use of decision rules used to to correctly predict the outcome of an input variable. The final result is a tree featuring *decision nodes* and *leaf nodes*
- **Random Forests**: empowers the concept of a decision trees by using multiple decision trees simultaneously. Each decision tree computes an output value depending on the input and the final value is said to be a "vote" from the decision trees
- **Naive Bayes**: is a family of probabilistic algorithms based on popular Bayes' Theorem. It's an algorithm both simple and reliable and it has been used with many purposes regarding NLP (Natural Language Processing) problems
- **K-Nearest Neighbors**: is a simple yet versatile algorithm that is often used for classification by use of a distance function between values (i.e. points) where a case is classified by "a majority vote of its neighbours"[23]
- **Neural Networks**: simply are a set of techniques based on the mathematical models of the brain. Neural networks operate in similar fashion to the brain's neural network

[22] Sanjiv Ranjan Das, data science : theories, models, algorithms, and analytics

[23] K Nearest Neighbors - Classification, http://www.saedsayad.com/k_nearest_neighbors.htm

- **Support Vector Machines**: are based on the concept of hyperplanes in a high dimensional space, generally used for classification and other tasks

Defining a model is often perceived as being the straightforward task of selecting a number of features from the data set and choosing the corresponding algorithm to use for better describing the dataset. However, selecting a model is half of the job. The need to assess its correct functioning is measured by the use of metrics that should be defined in accordance to the classifier and the type of problem to be solved. By defining the type of work the algorithm need to do, a family of metrics can be identified. In particular:

- **Classification** algorithms measures their success when they correctly predict the class (i.e. label) of a text instance, while they are considered to be wrong if they cannot correctly predict it. The most common metrics to measure the classification efforts of a model are defined as *log loss*, *mean consequential error*, *mean average precision and recall*, *accuracy*, *multi class log loss*, *hamming loss*, *mean utility*, *matthews correlation coefficient*[24]
- **Regression** algorithms are considered to be "right" when predicting a value close to the actual one and "wrong" when predicting a value that is far from the actual one. The error metrics associated with this kind of algorithms measures the distance between the predicted and actual values by use of *mean squared*, *mean absolute*, and *mean median* errors metrics
- **Ranking** algorithms should give a high rank to relevant elements. Thus, they are "right" when performing said task correctly, and "wrong" when assigning high rank to non relevant elements. Several measures of effectiveness have been devised, amongst which the *mean average precision, mean reciprocal rank*, *kendall's tau*, *spearman's rho*.

2.1.3.5. Presenting findings

After the creation of a model and the gathering of results there is, however, still the need for a clear and neat graphical representation of the insights that have been gained thanks to the previous steps. Even though communication is a very underestimated skill it is very important to communicate findings by depicting them by the most appropriate media, being it images, charts, presentations slides or very thorough reports. All of these media need to the list the most important conclusions that can be drawn based on the exploratory analysis phase and the model that was built to represent the problem settings. When using charts to illustrate the research findings on a subject it is of uttermost importance to choose the right one to convey the results with clarity and effectiveness. Each chart is better suited to display specific results and correlations between variables, for example:

- **Dispersion charts**: are used to highlight correlations between variables, where each single point represents a single observation. This type of chart can help finding a correlation between variables but it however cannot provide any additional information on the causality relationship between them

[24] Metrics, https://www.kaggle.com/wiki/Metrics

- **Line charts**: are one of the most used types of charts employed in data presentations. They are most notably used to show variable variations over time
- **Bar charts**: are generally employed for comparison amongst variables of different groups. Commonly, the x axis describes a categorical variable while the y axis describes a quantitative one
- **Histograms charts**: are used in order to show the frequency distribution of a set of continuous data
- **Box-plot charts**: are commonly used to depict a value distribution. They are created by drawing the following five values:

    - the minimum value
    - the first quartile
    - the medium value
    - the third quartile
    - the maximum value

2.1.4. Applied data science: extracting info on the web

The problem is an obvious one. Given the great amount of data on the web, how can Data Science help in gathering information from data? Depending on the scope of the analysis, if the objective is to obtain measureable attributes describing various qualities of the articles, the study can be reduced to the problem of "*news analysis*". This can be defined as:

Def 2.1.4.1 (news analysis) *the measurements of qualitative and quantitative attributes of news articles, especially concerning the measurements of characteristics such as the* ***sentiment****, relevance, novelty and authenticity of an article.*

The activity of news analysis is comprehensive of a set of techniques employed to classify documents and information by means of specific metrics. While this is a broad field, is also considered equally rooted in the fields of *Information Retrieval*, *Machine Learning*, *Statistical Learning Theory*, *Network Theory* and *Collaborative Filtering*.

The following paragraphs will discuss the practical use of data science in the settings of news analysis as well as various techniques for extracting the relevant information from data.

2.1.4.1. News Analytics

News analytics can be thought of in terms of its components. In particular, this branch of text analysis can be broken down to the following three levels of scope: text, content and context.

1. **text**: this component simply deals with analysing and extracting information from text sources
2. **content**: this second layer expands the analysis domain to that of media content in the form images, format and so on and so forth

3. **context**: referrers to the existing relationships among information items

News analytics algorithm grows in complexity at each level. This means that algorithm only dealing with the textual level will be less complex with algorithm embodying the content level that, in turn, will display less complexity to algorithms also dealing with the contextual level. A famous document analysis algorithm including all three levels is the PageRank algorithm which was already mentioned earlier in this work. In the case of news analytics, however, the analysis is often conducted at the textual level. Before dwelling more into details about the application of data science in the context of news analytics, the paragraph will continue by presenting the general algorithms (2.1.4.1.1. Algorithms) and models (2.1.4.1.2. Models) to be used in this case.

2.1.4.1.1. Algorithms

The algorithms used in the field of *news analytics* are related to the tasks of gathering data from the web, pre - processing it and analysing it.

- **gathering data**: In order to get data from the web two algorithms are particularly useful. *Crawling* algorithms and *scraping* algorithms. Recalling from the research's previous definitions of a *crawler* is obvious that they are needed in order to explore the web, moving from one hyperlink to another, choosing the path by using heuristics, whilst a *web scraper* simply downloads the content of a chosen web document making it locally available for further processing
- **text pre - processing**: before any form of analytics can be applied, text obtained from the web needs to be cleaned by carefully:
    - removing all HTML tags: all tags such as, for example, the <body>, <p>, <h1>, &nqt; tags are removed from the document corpus
    - tokenizing text: text is tokenized by splitting statements into a word list
    - expanding text abbreviations to their full form: words abbreviations are removed and abbreviated words are turned into full form
    - handling negations: negations is handled by detecting negation words and by tagging the remaining part of the sentence with markers in order to inverse its meaning
    - stemming of words: is a process by which words are replaced with their roots in order to be treated equally by the algorithms. Stemming is of course language dependent

- **analysis**: the analysis can be done through several algorithms depending on the actual information to be gathered from data. Algorithms often used in news analytics most notably includes:
    - TF-DF
    - sentiment analysis
    - influence evaluation
    - word count / word clouds

Sentiment analysis and **Influence evaluation** techniques will be further explored later in the paper.

2.1.4.1.2. Models

There are various models available to extract information from documents. In paragraph *2.1.3.4. Modelling data* various models were briefly described for a general data modelling process. However, the analysis will now focus on the more specific task of text classification. Formally, this task can be defined as the solving the problem of inferring a class *Y* for an input *X* by predicting it based on previous data. Text classification is then the task of appropriately label input *d* (a document) with a class (or label) *c*. Focusing on text classification tasks this paper takes into account the different kind of classifiers. More precisely, each type of classifier shows a different way of doing classification:

- **generative classifier**: computes its predictions by modelling each class. A generative classifier will construct a model of each class and try to fit values in the respective category by comparing them with classes and assessing the best "fit" for every value. More formally, generative classifiers return the class that most likely generated the observation
- **discriminative classifier**: do not construct the model but simply draw a decision boundary directly and calculate on which side the values will fall by learning what features from the input are the most useful in distinguish between all classes

It is important to stress that the type of data as well as the type of problem to be solved is determinant in the selection of the specific classification model to be used. This paragraph will cover two classifiers: the **Naive Bayes** classification model (*generative classifier*) and the **Support Vector Machines** (*discriminative classifier*):

- **naive bayes**: is probably the most widely used classifier. The model's implementations are based on two main assumptions. In particular, naive bayes classifier are based on:
    - the **bag-of-words** representation of documents (i.e. every document is seen as an unordered set of words where position of words is ignored but frequency of appearances is tracked). This type of model allow to express a document as a set of features without loss of generalization. This set of features is often seen in the form of a vector, called the "features vector"
    - the (naive) assumption that the features of a document are independent of each other. In practice the independence assumption is often violated but the classifiers still tend to perform well. However, *strong* violations of the assumptions still cause Naive Bayes classifiers to perform poorly

  Naive Bayes classifiers are probabilistic classifiers and will return the class that has the most probability to describe the document. They are based on the idea of

Bayesian inference that implies the possibility to derive a “deduction” (i.e. assign a class) for an observation (i.e. the input) based on the data.

This idea is mathematically expressed by the following formula:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

From the above idea, given a document *d* and a class *c* belonging to a set of predetermined classes, the classifier allows to estimate the correct class by solving the following: [25]

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

The above equation simply states that the probability of a document *d* to belong to a class *c* is estimated by selecting the maximum *posterior probability* of the document belonging to each different class. This can be mathematically expressed by expanding the previous equation into the following one:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$

As for all classifiers, the naive bayesian needs to be trained on an initial corpus with pre-classified text in order to be effective. The training sets must be *representative* of the entire population. The training set provides the *prior probabilities* that allows the classifier to perform the analysis. Whenever a new value appears that needs to be classified but has not been previously presented to the classifier via training sets, the models account for it by use a smoothing value used to avoid wrongly attributing a zero probability to belonging to any class when evaluating a document containing said unprecedented element. There are several available implementations of the Naive Bayes classifier model:

- Multi-variate Bernoulli Model: are characterized by their binary implementation of the *feature vector* by use of a binary interpretation of a document’s token,

[25] Speech and Language Processing, Daniel Jurafsky & James H. Martin.

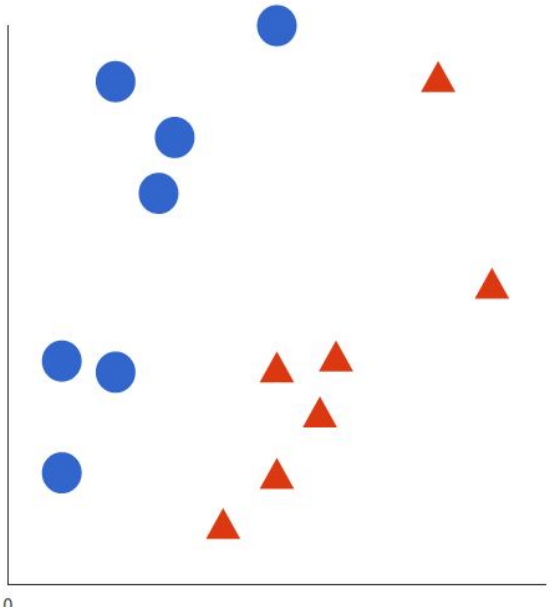
which are associated either with a 0 or with a 1. The Bernoullian model penalized the non-occurrence of a feature and generally offers better performances than the other Naive Bayes implementations when used with shorter documents
    - Multi-nomial Naive Bayes: utilizes term frequencies to characterize documents or the TD-IDF variant to account for stop words and to better identify the importance of words through a set of documents. This model tend to outperform the *Multi-variate Bernoulli model* for larger documents
    - Gaussian Naive Bayes: is used when dealing with continuous non categorical data under the assumption that the probability distribution of the features follow a normal (Gaussian) distribution

  The above models works on the assumption that data has been correctly preprocessed before being submitted.

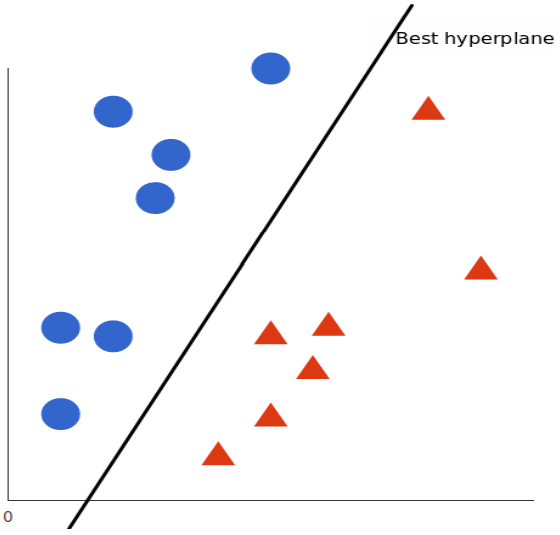
- **support vector machines**: are classifier techniques that describes textual content as a vector in a high-dimensional space, where the number of dimension is described by the number of word in a dictionary. With this particular technique, each data item is plot as a point in a n-dimensional space (n being the number of features describing data) where the value of each feature is represented as that of a specific coordinate. The principle states that text sources belonging to the same category will be plotted in the same region of the space, thus the classification procedure is performed by finding the *best* hyperplane dividing the classes. Visually, it is better illustrated by an example[26] :

    1. given two categories (red and blue) and two features (x and y) the SVM classifier will work by outputting either red or blue for a given pair of coordinates (x,y)

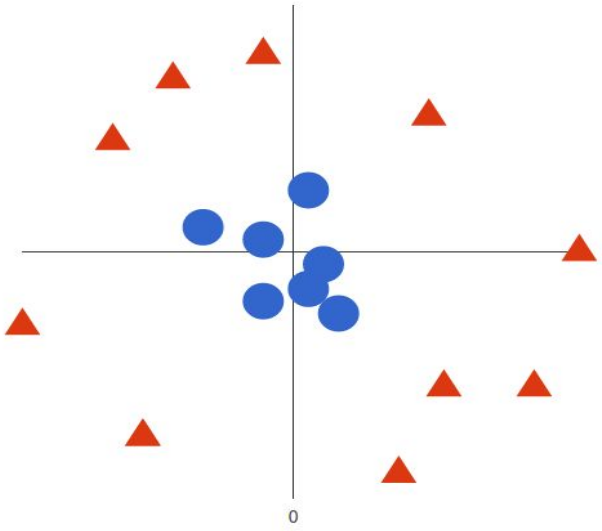
[26] An introduction to Support Vector Machines (SVM), https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/



2. The **Support Vector Machine** classifier, then, is tasked with outputting the n-dimensional hyperplane (where n is equal to 2) to best separate categories. This line is also called the *decision boundary* as it draws a separations between values falling on either side of the boundary



The catch with the Support Vector Machines model is that, however, given a set of *n* features, SVMs find an *n-1* dimensional hyperplane to separate between categories. This has dire consequences over the model performances that do very badly with non linearly separable datasets. This implies the need to transform the non linearly separable datasets into linearly separable ones but, as it turns out, this becomes very difficult to do for datasets with lots of features. An example of a non linearly separable data set is depicted in the following image:



Fortunately, the Support Vector Machines model is implemented using a so called **kernel**. The kernel trick works by transforming features vectors by only considering them dot-product wise, thus avoiding expensive dimensional transformations, hence allowing Support Vector Machines to work with non linearly separable data. For the specific task of text classification, Support Vector Machines are used by selecting vectors features representing every text in the dataset by using world frequencies or the TF-IDF values as for the Naive Bayes models. The final step is choosing the most appropriate *kernel function.*

Def 2.1.4.1.3. (kernel functions) a function that takes two input vectors in the original dimensional space and produce the dot product of the vectors in the feature space

Very intuitively, kernel function can be thought of as similarity functions that given two objects can output a similarity score. Kernel functions can be selected by choosing between:

- Linear kernels: defines the kernel function to be the similarity between the features vectors *x* and *y* (where *c* is an optional constant)

$$k(x, y) = x^T y + c$$

  The linear kernel is the simplest kernel function and often performs well for classification problems. This kernel function is generally used when the number of features is larger than the number of observations

- Polynomial Kernels: are well suited for problems featuring normalized training data sets. The function is defined as the dot product between the feature

vectors *x* and *y* in combination with additional terms *c* (i.e. constant term), *alpha* (determining the slope) and the polynomial degree *d*

$$k(x, y) = (\alpha x^T y + c)^d$$

- Gaussian Kernels: are example of radial basis function kernels. Here, the function features an additional parameter *sigma* which should be fine tuned for better performances

$$k(x, y) = \exp\left(-\gamma \|x - y\|^2\right)$$

This kernel function is generally employed for problems characterized by a larger amount of observations with respect to the number of features. Gaussian Kernels suffers, however, from poor computational speed and that should be taken into account when processing very large amount of data (i.e. observations)

2.1.4.1.3. Putting it all together

This chapter ends by summarizing the procedures related to the Data Science approach towards *news analysis*. Based on the five steps presented in *2.1.3. Data Science: a five steps scientific approach*, the following actions should ensue:

1. **Posing an interesting question**: as already defined, *news analysis* favours the extraction of information related to several possible source material attributes. In this case, the focus is on the "*sentiment*" expressed by the source. The question can then be simply stated as "Is the sentiment about the subject positive or negative?"
2. **Fetching Data**: for simplicity's sake, let's define the context of the problem to a set of documents listed by a single website. It will also be assumed that no actual database access is provided.
    a. The first sub-activity in this case is to gather all of the documents related to the subject by either use of a customized *crawler* or through a search engine.
    b. Secondly, it immediately follows that web scraping on all documents related to the subject that are listed by the fictional website should be performed, thus obtaining a set of raw data.
    c. The second steps calls for a required exploration of raw data. A good exploration should then determine the following:

        i. Does the text contain any opinionated information about the subject?
        ii. How long is the text?
        iii. Is it possible to identify an author and time of creation of the text?
        iv. At what level of data should the text be categorized?
        v. How can data be represented?

While some questions are related to the specific case at hand, it can be generally replied the following answers to question iv: textual sources such as those embodied in news articles and similar are unstructured qualitative data clearly belonging to the *Nominal Level* of data. However, when defining the a model for the data, it is worth noting that their nature will most likely by transformed for the sake of the algorithms that will receive and compute them.

3. **Modelling Data**: when creating a model a number of features belonging to the data source should be selected. What is it that makes the various source items differ from one another? In the first place, the following features can be identified as being distinctive:

    a. Authorship
    b. Time of publication
    c. Url
    d. Sentiment

    From this list a model with four different features can be constructed rather easily. By examining data, the following characteristics are on display:

    a. Data can be linearly organized in a vector like structure (i.e. *feature vector*)
    b. The number of features can be regarded as very low
    c. The number of classes to which to categorize data to is very low (i.e. text can have either positive or negative connotation)
    d. Classes are well defined (i.e. they don't overlap with each other)
    e. Features are independent of each others
    f. Data can be represented by use of the bag-of-words model

    All of the above make the **Naive Bayes** models and algorithms good candidates for use with this set of data. As already mentioned, the Naive Bayes is a probabilistic classifier. Naive Bayes classifiers need to be trained with appropriate training sets in order to be able to later classify values accordingly. As with all machine learning devices, Naive Bayes goes through the phases of:

    - Training
    - Testing
    - Validating

    In order for Naive Bayes to be on par with more advanced machine learning methods such as the previously mentioned **Support Vector Machine**, the input needs to be appropriately pre-processed by appropriate *tokenization* (optionally employing the n-grams technique), *removing stopwords*, *stemming* or *lemming* of words. Given that the training phase was correctly carried out, the testing phase is carried out on previously unpresented data and the performance of the classifiers are validated by measuring it with the appropriate validation metrics chosen for the model. Given the

nature of the algorithm, the correctness of the model is evaluated by choosing from a list of possible classification metrics (e.g. precision and recall, accuracy, etc.). Finally, when the model is fine tuned and performs well on new data, the focus switch on presenting finding in an efficient way.

4. **Presenting findings**: in order to successfully present findings the appropriate charts should be chosen when presenting data. According to the focus of the analysis - which in this case is on the sentiment of text - either *line* or *bar* charts are good candidates for presenting results.

3. Explorative algorithmic techniques

In this chapter the analysis will focus on two techniques that can be used to extract qualitative information from a textual source:

- **Sentiment analysis**
- **Influence evaluation**

Lastly, the chapter will explore additional measuring techniques that could be useful when dealing when in the context of Sentiment Analysis related tasks.

3.1. Sentiment Analysis

Text classification can occur based on several parameters. A researcher might be interested in analysing the category that the source material belongs to, the kind of tags that could be generated in order to better describe (i.e. classify) content and, more interestingly, the understanding of the sentiment and opinion of a subjective text (i.e. a text that most likely contains an opinionated view about any subject). The comprehensive set of techniques devoted to such analysis of data is formally called **Sentiment Analysis**.

Def 3.1.a. (sentiment analysis) *as set of techniques adopted to capture the general mood of text by using either Natural Language Processing, Statistics or Machine Learning. Sentiment analysis focuses on people's opinions, emotions or attitude towards a subject and is - as such - an important tool to understand how people feel about something.*

On a bigger scale, sentiment analysis is not limited to cover the "sentiment" of a specific source only, but concerns itself with a more sophisticated concept describing the "opinion" embedded in text. The introductory definitions for the concepts of "opinion" and "sentiment" thereby presented will further be expanded and refined later in the paragraph.

Def 3.1.b. (opinion) *an opinion includes the concepts of sentiment, evaluation, appraisal, target of the opinion*

On the other hand, the concept of sentiment is simply defined by the following

Def 3.1.c. (sentiment) *the underlying positive of negative feeling implied by an opinion*

The distinction between the two concepts implies that usual definitions of Sentiment Analysis often suffers from an excessive generalization consequently struggling to capture the inner complexity of data. In order to better understand the issue at hand and apply the right algorithms a more structured definition of the problem should be provided. Sentiment Analysis can be defined at different levels of the *spectrum*. Given the common way Natural Language Processing recognizes the structure of text by means of its different parts (i.e.

words, *clauses*, *phrases* and *sentences* are all part of a *document*), the analysis can focus with a broader or more specific spectrum width[27]:

- **Document**: at document level, the analysis aims at classifying whether a document is more positively or negatively oriented on overall. This task, often referred to as *document-level sentiment classification* assumes that each document expresses opinions on a single entity (e.g. a single product or subject)
- **Sentences**: at this level, sentences are analysed to determine their polarity (i.e. whether they are connotated by a positive, negative or even neutral meaning).
- **Clause**: researches have also analysed the possibility to use the polarity of clauses for better results, but analysis at this level is often believed to be poor and thus rarely used
- **Phrases**: analysis at phrase level splits a phrase into more components, called *entities* and *aspects*. For this, *phrase-level document analysis* is more often referred to as *aspect-level document analysis*. Aspect level analysis performs finer-grained analysis by targeting the opinion itself, on the idea that an opinion consists of a *sentiment* as well as a *target*, following the definition of opinion given in *Def 3.1.b.* In many applications, opinion targets are described by entities and (or) their different aspects, thus the objective of *aspect-level document analysis* is that of uncovering sentiments related to entities and (or) aspects of a target

As can be clearly seen in the above list, Sentiment Analysis can be performed with a high or fine granularity. The finer the granularity gets, the more sophisticated the Natural Language Processing modules needs to be in order to solve the subproblems. Analysis at aspect level is the most difficult and requires solving different kinds of subproblems that will be presented later in the document. This work will introduce the formal Sentiment Analysis framework proposed by author **Bing Liu** In his book "**Sentiment Analysis and Opinion Mining**". Possible implementations, enhancements or approximations of the framework will be discussed in the succeeding paragraphs.

3.1.1. A framework for Sentiment Analysis

The most important indicators of sentiment in text are particular words that might help identifying the overall *polarity* (or *orientation*) of the document. These indicators are called *sentiment words*. These words are often compiled in lists - instrumental to Sentiment Analysis - called *sentiment lexicons*. Classic Sentiment Analysis has often based its approach on sentiment lexicons only but it is shown that it is often not enough for most cases. For example, when trying to extract the polarity of a review for a product, there might be several product features touched by the reviewer, each of them with a positive or negative connotation. Classic techniques are guilty of overlooking said details by working at higher granularity level (i.e. document level) than optimal (i.e. aspect level) - although it can be said that not all applications might need the high level of accuracy provided by this approach.

[27] Bing Liu,(2012), "Sentiment Analysis and Opinion Mi
ning, Morgan & Claypool Publishers".

Sentiment analysis is a Natural Language Processing problem - albeit a highly restricted one - and, as such, it is affected by most of the sub problems concerning this area of research, in particular: negation handling, word sense disambiguation and coreference resolution[28]. One important thing to bear in mind when facing Sentiment Analysis is that it deals with *opinion* that are, by nature, different from *factual information* as they are subjective in nature. Furthermore, an opinion can be stated on different aspects of the *entity* subject to the opinion itself.

Def 3.1.1.a. (entity) An *entity e* is a product, service, topic, issue, person, organization, or event. It is described with a pair, *e: (T, W)*, where *T* is a hierarchy of *parts*, *sub-parts*, and so on, and *W* is a set of *attributes* of *e*.

This observation is what prompts the following formal definition for opinion:

Def 3.1.1.b. (opinion) An opinion can be defined as a quintuple:

$$(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$$

Where e_i is the name of an entity, a_{ij} is an aspect of e_i, s_{ijkl} is the sentiment of aspect a_{ij} of entity e_i, h_k is the opinion holder, and t_l is the time when the opinion is expressed by h_k. The sentiment s_{ijkl} is positive, negative or neutral, or expressed with different strength or intensity levels.

The above definition stress the importance, in this framework, to model sentiments after all the other members of the opinion quintuple, thereby taking into account, the specific entity *e*, aspect *a*, opinion holder *h* and the time the opinion was expressed *t*. While extremely thorough, the above definition is not too overly complex in order to not increase the difficulty of the problem to be solved. *Def 3.1.1.b*, despite some limitations,. Is usually more than enough for most applications as it provides both *qualitative* and *quantitative* information.

3.1.1.1. Tasks of sentiment analysis

The discussion in the precedent part of the paragraph is instrumental in defining the tasks of Sentiment Analysis with respect to this framework:

- **Objective of sentiment analysis**: given an opinion document *d*, discover all opinion quintuples $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ in *d*.

From the above definition, Liu easily derived the main tasks related to Sentiment Analysis in relation to the five components of the quintuple. To better the modelling of Sentiment

[28] Bing Liu,(2012), "Sentiment Analysis and Opinion Mi
ning, Morgan & Claypool Publishers".

Analysis into a structural step-by-step process, the framework introduces the following additional definitions:

Def 3.1.1.1.a. (entity category and entity expression) An *entity category* represents a unique entity, while an *entity expression* is an actual word or phrase that appears in text indicating an entity category

Def 3.1.1.1.b. (aspect category and aspect expression) An *aspect category* of an entity represents a unique aspect of the entity, while an *aspect expression* is an actual word or phrase that appears in the text indicating an aspect category

Def 3.1.1.1.c. (explicit aspect expression) Aspect expressions that are nouns and noun phrases are called *explicit aspect expressions*

Def 3.1.1.1.d. (implicit aspect expression) Aspect expressions that are not nouns or noun phrases are called *implicit aspect expressions*

Finally, based on the above discussion, the framework tasks can be broken down to the following steps, where, given a set of opinion documents D, sentiment analysis essentially consists of the tasks below:

- **Task 1** (entity extraction and categorization): extraction of all entity expressions in D and categorization of entity expressions into categories for each unique entity e_i
- **Task 2** (aspect extraction and categorization): extraction of all aspects related to entities, while categorizing them into clusters where each aspect expression cluster of entity e_i represents a unique aspect a_{ij}
- **Task 3** (opinion holder extraction and categorization): is the task of extracting the opinion holder (i.e. the person expressing in the document) and categorize them
- **Task 4** (time extraction and standardization): account for extracting and standardizing the times at which the opinions were stated
- **Task 5** (aspect sentiment classification): sits at the core of Sentiment Analysis, determining the polarity of the opinion on aspect a_{ij}
- **Task 6** (opinion quintuple generation): is the final task of producing all opinion quintuples expressed in document d after completion of the above tasks

3.2. Influence Score

Despite the definitions and theoretical framework for **Sentiment Analysis** that has been introduced in this chapter, the Natural Language Processing techniques that need to be employed for it to be effective require additional effort that might not be justified in front of the computational loss of speed when processing large sets of data. To counter the issue,

this work proposes an approximation techniques that loosen the Sentiment Analysis Framework definition by employing a set of text analysis rules: **Influence Score**.

3.2.1. Defining the problem

The framework defined in *3.1.1.* works very well in all general cases concerning the Sentiment Analysis procedures. However, it can be discussed that since the framework definition needs to account for general tasks, it might *overfit* specific - more restricted - problems. For example, let us imagine the following setting: *A user is curious about the articles and posts in which is name could came up. The fact that some articles might be talking either about him or simply linking him indirectly with non positive facts (or statements) is of concern for him. So, he would like to know if there exists any article, in the indexable web, which may be talking about him with an unpleasant tone.* This problem is more general than what previously examined. This can be better seen by breaking the problem setting into smaller logical units:

1. The user is curious about articles regarding **his** persona
2. The user is interested in knowing if his persona is mentioned in ***positive*** or ***negative*** articles
3. By effects of above points, the user reputation might be **directly** or **indirectly** affected by either *positive* or *negative* content

The above points simply states that it can be identified an important subproblem to Sentiment Analysis. This problem, can referred to as “Directed Sentiment Analysis” and is (informally) defined as follow:

- **Directed Sentiment Analysis**: is the problem of identifying good or bad influencing documents with respect to a specific subject (or entity) and to provide a thorough summary of the findings.

It is trivially deduced that the “directed” feature of Directed Sentiment Analysis refers to the target towards which the effort is *directed* to. This simplify a number of considerations stated in the framework discussed by Liu [1]. The next paragraph will discuss the modifications to the framework derived from this simple problem setting redefinement but, before dwelling into further discussione, let's introduce the following short-handed formula to indicate the Sentiment Analysis computation by the framework.

Def 3.2.1.a. (sentiment analysis function): a sentiment analysis function is defined simply as a function capable of translating an input $x \in D \vee x \in S_d$ (where D is the set of all documents to be examined and $d \in D$) into either a positive or negative output value y. The function is indicated in the following way:

$$\varphi(x) = y$$

It is worth noting that the $\varphi(x) = y$ equation can result into either a positive or negative output that could in turn be mapped to match numeric values, like, for example, 0 for a negative output and 1 for a positive one.

3.2.1.1. Additional measures

This short paragraph introduces an additional measure used in the simplified framework that will be presented later in the chapter. The function, called **Jaccard Similarity** index is generally used in the field of Information Retrieval and Natural Language Processing to provide a comparison between documents, so to identify similarities between them.

Def 3.2.1.1. . (jaccard similarity in a set of documents): given a set of documents D and an initial document $A \in D$, the jaccard similarity index between A and D' where $D' = D - \{A\}$ is defined as the average value of all jaccard indexes for A and any document $B \in D'$ or, mathematically:

$$J_A(A, D') = \frac{1}{n} \sum_{i=0}^{n} J(A, B_i)$$

$$s.t.\ B_i \in D'$$

The previous definition provides a simple way to obtain an average "similarity" indicator of a document with respect to the remaining documents. This can be combined with all documents of the document space D and can be used to draw some interesting observations. In fact, by examining these values, it is possible to gain a sense of what the heterogeneity of the document space D is on average. The algorithm itself is pretty simple and is based on the definition for **jaccard similarity in a set of document** and on the definition for the **document space jaccard similarity** thereby introduced:

Def 3.2.1.1. . (document space jaccard similarity): given a set of documents D and $A \in D$, the average jaccard similarity $\overline{J}$ corresponds to the median value identified between the computed $J_A(A, D')$ values for the set of documents D.

Def 3.2.1.1. . accounts for all $|D|$ members of the set to be examined during the computation of the jaccard similarity in a set of documents. $J_A(A, D')$ computations involves confronting sets modelled after the bag-of-words model commonly used in calculating the similarity between two documents and is, as such, a pretty expensive procedure as it involves operations such as tokenization and stemming of all sets. In a real life scenario this is still feasible but yet not advisable for computational time sake. The amount of jaccard similarity values that need the be computed for a set is simply equivalent to a simple combination of $\binom{n}{k}$ where n is the number of sets and k is, simply, two. A simple mathematical example is quite enough to understand that this operation should be approximated in order to not hinder computational time. For example, computing $J_A(A, D')$ where $|D|$ is equal to 10 simply requires 45 operations. The number of operations involved, however, steeply increases

when the number of elements starts being quantitative relevant. Simply wanting to examine the average jaccard similarity of a set of 100 documents requires to compute $\left(\frac{100}{2}\right) = 4950$ operations! This, of course, demands some form of limitations be put into place when dealing with growing numbers and will be taken into account when including this calculation into the framework. These considerations are also implemented in the algorithm for computing the **document space jaccard similarity**:

Alg. 3.2.1.1.a. (document space jaccard similarity): given $A \in D \, s.t. \, D$ represents the document space, the algorithm computes the **jaccard similarity in a set of documents** for the top ten documents of the document space, reorders the list increasingly and then outputs the median value for the whole document space.

$$
\begin{aligned}
&for\ each\ A \in D\ do: \\
&\quad D' = D - \{A\} \\
&\quad Values.push(J_A(A, D')) \\
&end\ for\ each \\
\\
&Values.sort() \\
&Median\ index = \frac{|D|+1}{2} \\
&return\ Values.at(Median\ index)
\end{aligned}
$$

3.2.1.2. Adjusting the framework

Having defined the problem settings to a more specific environment this work introduces new definitions for Directed Sentiment Analysis, as well as a lighter definition for a number of framework items.

As discussed, the stress on Directed Sentiment Analysis is on the adjective “directed”, meaning that there’s no need to take into account all entities that could be possibly extracted from an opinion document. There’s, however, the need to focus all efforts on the entity the analysis is interested in. This particular entity, is called a **target entity**:

Def 3.2.1.2.a. (target entity) A target entity $\overline{e}$ is the entity for which to identify an opinion. A target entity can be any type of entity as defined in *Def 3.1.1.a.*

Target entities can be about anything the user is interested in. Usually, however, they are individuals or organizations that hold some importance in the user’s eyes. When confronted with long articles a number of language specific issues may arise, but, more in particular, the analysis might be faced with a text featuring *indirect* references to the topic of interest. It is very common to avoid referring to a subject by directly mentioning it at all times, as it is stylistically unwise to do so. However, as much as this may be of value for the future career of a writer and the quality of his work, the need to use *synonyms* in place of the actual topic name (i.e. target entity) leads to obvious problems related to feature extraction. This work

proposes to solve the problem by using a list of terms related to the topic of interest, simply called a **synonym list**.

Def 3.2.1.2.b. (synonym list) A synonym list $L_{\bar{e}}$ for a target entity $\bar{e}$ is a series of items $l \in L_{\bar{e}}$ such as each item l is a synonym for $\bar{e}$.

A synonym list allows for a simpler feature extraction as it can be tailored to each specific target entity or, more generally, to any kind of related titles. For examples, let us imagine the Sentiment Analysis is **directed** towards the following subject: a politician from Italy who happens to be appointed as minister of the interior, a former lawyer and a number of other titles. As it often happens in papers, also on the web there exists possible variations of its full name. For the sake of the example, the politician's name will be "Andrea Marcelli". The synonym list, in the context of this example, could result in something like this (when combined with the entity):

Marcelli,
A. Marcelli,
Minister of the interior,
Minister Marcelli,
Minister Andrea Marcelli,
Minister A. Marcelli,
Lawyer Marcelli,
Lawyer Andrea Marcelli,
Lawyer A. Marcelli

As the number of possible variations are finite, having a list of synonyms for each titles (or synonym) is possible and can help in identifying *indirect* and *direct* references to the target entity. Similar processes have been employed in various works derived from Liu's framework, in particular with regard to sentence level analysis and the use of techniques to identify subjective sentences. This work, however, chooses to follow more relaxed guidelines with respect to the framework's, thus not evaluating the subjectivity of sentences but implementing a similar concept in order to detect *important sentences*. To do so, the following assumption is considered:

- **Sentences influence**: sentences sentiment can be influenced by the sentiment conveyed by other nearby sentences

The above assumptions derive from the simple observations that, generally speaking, concepts and reasoning are linked and follow a specific flow in text (this is especially true in *news articles*). In particular, a sentence referring to a subject is very likely to be followed by further enquire about the same topic if no other entity is detected. This is even more true if *indirect* or *direct* references to the target entity are successfully detected. Based on the above assumption, the following definition can be derived:

Def 3.2.1.2.c. (sentence peers influence): a sentence containing a reference to the target entity $\overline{e}_i$ can either be followed or preceded by an additional sentence conveying opinions on the target

- Sentence successor influence: a sentence containing a *direct* reference to the target entity $\overline{e}$ can be followed, by a sentence conveying opinions on the target as well. This kind of sentence is indicated by symbol $\beta_{\overline{e}}$
- Sentence predecessor influence: a sentence containing an *indirect* reference to the target entity $\overline{e}$ can be preceded, by a sentence conveying opinions on the target as well. This kind of sentence is indicated by symbol $\alpha_{\overline{e}}$

Def 3.2.1.1.c is built as a simple variation of a concept modelled in [1] in *5.2 Basic Rules of Opinions and Compositional Semantics*. In particular, in the paragraph, the author states that **opinion rules** express a concept that implies a positive or negative sentiment. Opinion rules are nothing but decisional steps in determining the polarity of an aspect (in aspect-level document analysis) but can also be used as heuristics rules to infer the actual polarity of a sentence. This definitions are instrumental in introducing what this work proposes as a simple parameter to measure the possible influence of an opinion document d with respect to a target entity $\overline{e}$. This parameter, is called the *influence score* and aims at measuring the actual impact a document can have on the subject the search is currently interested in. It is computed by combining the following factors:

- Directed / undirected sentence polarity
- Sentence peers influence
- Overall polarity of the document

This parameters are combined into a weighted formula that models the concept of **influence score**. The process of computing the influence score of a document is simply called **influence evaluation**.

Def 3.2.1.2.d. (influence score): Influence score represents a numeric factor indicating how much an opinion document d is estimated to *influence* the "online reputation" of the target entity $\overline{e}$. The value is computed by the following formula $\forall s \in d$ where s is a sentence:

$$\varphi(d) = s_i \times 0.75 + [\ \varphi(\alpha_{\overline{e}}) + \varphi(\beta_{\overline{e}})\ ] \times 0.20 + \varphi(d) \times 0.05$$

Where s_i represents the sentiment for a sentence referring to - directly or indirectly - the target entity $\overline{e}$, $\varphi(\alpha_{\overline{e}})$ and $\varphi(\beta_{\overline{e}})$ are the sentence predecessor and successor factors scores respectively and $\varphi(d)$ is the computed Sentiment Analysis value of the whole document.

Weights for determining the influence score of a document are chosen empirically through observation and common sense. This can be easily explained by examining the factors used in the equation and their meaning in the larger context of a document. In fact, a sentence

containing a direct or indirect opinion ought to have a bigger impact on calculations with respect to, for example, the overall polarity of the document or its predecessor / successor score. Influence scores are unique for each document and their value is normalized to range between $[0, 1]$ as the score is used as a weighing factor in combination with the sentiment analysis value computed for the whole document. Directed Sentiment Analysis can then be defined as the following activity:

- **Direct Sentiment Analysis**: is the activity of combining a document's overall polarity with the computed influence score for the target entity

More importantly, it is defined that, different values of the *influence score* contribute to determine the importance of the polarity towards one direction or the other. In a very similar fashion to what was introduced with the concept of *Fuzzy Logic*, influence score can be seen as defining a membership function to either polarity values (sets). In fuzzy logic, the membership of an element to a set is a single value ranging from 0 to 1, indicating a degree of membership to the class. Similarly to this concept, the values indicated in the following lists, provide an indication over the strength with which it is possible to express the polarity of the document.

In particular, these values defines the *strength* of the membership to a polarity value:

- Influence score $\in [0.40, 0.60]$ is considered to be associated with a normal level of membership to either the *positive* or *negative* class of values
- Influence score $\in [0.20, 0.39]$ or $\in [0.61,\ 0.80]$ is considered a strong indication of membership to either class
- Influence score $\in [0, 0.19]$ or $\in [0.81,\ 1]$ is considered a definitive indication of membership to the respective class

By combining the influence evaluation procedure with the actual overall sentiment detected by the system is then possible to approximate *aspect-level document analysis* as described by Liu[1]. The introduction of the influence score determines a slight modification of definition *Def 3.1.1.b* of an opinion. By simply taking into considerations all of the adjustments made to the original framework, it is plain to see that:

- A single entity (i.e. target entity) is now considered, in comparison with the multiple possible entities that the original frameworks definition could account for
- Aspects hold no relevance in the context of future applications of this simplified version
- The introduction of the influence score paired with the overall polarity of the document is used as the main discerning factor over the effective importance of the document's content for the target entity

These considerations leads to the adjusted opinion definition of *Def 3.2.1.2.e* that, in comparison with *Def 3.1.1.b* drops the aspects support, introducing, instead, the influence score as well as the support for the document space jaccard similarity defined in *Def 3.2.1.1.*. :

Def 3.2.1.2.e. (opinion): an opinion is a sestuple s.t.

$$(\overline{e}, j, p, s, h, t)$$

Where $\overline{e}$ is the target entity, j is the **document space jaccard similarity** value, $p = \varphi(d)\ s.t.\ d \in D$ is the influence score, $s = \phi(d)$ is the sentiment class, h is the opinion holder while t represents the time at which the document was analysed.

Overall, this simplified framework reduces the tasks presented in *3.1.1.1. Tasks of sentiment analysis* to a bare minimum while still compensating the missing tasks with the less demanding features introduced during the course of the chapter:

- **Task 1** (target entity extraction): extraction of all entity expressions in D directly referring to target entity $\overline{e}$
- **Task 2** (sentiment classification at document level): the document polarity is inferred by the system to gain a sense of the overall text orientation
- **Task 3** (opinion holder extraction): the opinion holder is simply matched to the url at which the document is indexed
- **Task 4** (time extraction and standardization): account for extracting and standardizing the times at which the opinions were collected
- **Task 5** (influence evaluation): is key to Directed Sentiment Analysis and determines the influence score of the document in relation to target entity $\overline{e}$
- **Task 6** (document space jaccard similarity generation): the document space jaccard similarity value is computed for the top ten elements of the document space
- **Task 7** (opinion quintuple generation): is the final task of producing an opinion quintuple summarizing the general feelings expressed in document d after completion of the above tasks

4.1. Meaningful results: web reputation

In this brief chapter, the focus will shift focus towards an important - but often underestimated - consequence of the World Wide Web powered world: the reputation deriving from one's presence on the Internet, more commonly called **Web Reputation** or online reputation. Online reputation is an essential tool for individuals and businesses alike as positive contents and reviews influence the reputation and - in the case of business - revenues.

Ever since the World Wide Web escaped its infancy and turn to version 2.0 the landscape of its featured content naturally evolved to host more opinionated content, often being generated by its users by form of blogs, wikis and often shared by means of social networks. In many cases, the content's nature holds an opinion to something that can influence other people and that has been published on the Internet, freely accessible to anyone. This kind of information can have an impact on a person's life. Data per se have always been extremely valuable in strategy pianification and understanding of the environment and context, but this kind of information holds so much relevance in today's society that public figures, as well as private individuals interests, may for once intersect when dealing with matters regarding their public image on the Internet. As a matter of fact, the **web persona** is now starting to hold as much relevance as the real world persona and has also gained its own *neologism* in the form of the blend of web and reputation into the world *webutation*. The main issue of having a "web persona" (also known as a *digital footprint*) to take care of is the impossibility of successfully managing what the whole set of information published on the web. Contents on the web are often characterized by an almost uncontrollable nature, coming from different "angles":

- **authoritative sources**: magazines, newspapers, government agency websites
- **independent ones**: independent social network channels, independent blogs and – to some extent – wikipedia

Differences in the type of online presence can also be further stated, as they can be either **owned** or **unowned**.

- **Owned content**: is more easily controllable by the persona or entity it represents as it is linked to accounts, websites and information under direct control of the interested parties
- **Unowned content**: can be any type of media not under direct control of the subject (i.e. the author differs from the owner) and it is usually referred to as "earned" content (as is, for example, a Wikipedia page). This type of content cannot be bought or owned but can simply gained organically

As previously stated, the web's nature is that of a collection of intersected nodes, with no particular logical connection between them, if not for the numerous links that allow to gain some context out of the interconnected world of the web. Search engines often tries to make sense of the web by analyzing the link structure underlying the websites connections and, by

doing so, obtain a list of ordered result. As stated, this is just one amongst the different factors influencing the way search engines ranks pages and returns their results in a orderly fashion.

Other factors might include, depending on the specific search engine implementation:

- the trustworthiness of the source including the result
- the popularity of the content
- the number of votes (i.e. links) it receives from other web nodes
- internal (unkown) evaluation mechanism that might favour a page instead of the next one
- and more, depending on the search engine

Given all of the above, it is trivial to understand the difficulties in controlling what is being said about a specific person or subject. As the information can often drive people's thoughts on any given topic, creating unexpected shifts in not just closed environments or specific contexts but at bigger and bigger scales, hence it becomes of vital importance to the entity that is being linked to the articles.

4.1.2. Web Reputation Management

As famous businessman Warren Buffett once said "It takes twenty years to build a reputation and five minutes to ruin it. If you think about that, you'll do things differently". In this age of growing sources of information, it could even take less to ruin one's public image. This, and many other examples explicitate the needs for Web reputation management. Web reputation management is more precisely defined as the manipulation, analysis and creation of information regarding the reputation of a person, a firm or a topic on the web. The uttermost objective of Web Reputation Management is to establish and maintain a digital footprint on the web. Management of web reputation can either be:

- a continuous activity (i.e. the management is continuously performed during the year)
- a periodic activity performed at need or at specific times of the year

Choosing the perfect fit for the kind of management to enact is usually dependent on the client's requirements. Continuous monitoring is mostly useful to be able to trace and manage reputation over time, while a more sporadic activity is useful when particular interests are to be met. This being said, the most astonishing example of the potential of such activity has perhaps occurred this year and can be traced to the recent american presidential campaign. The winner of the campaign, the now american president Donald Trump, has been given an immense amount of visibility during the campaign by the somewhat unprecedented use of the social networks he implemented as he was running for president. Behind his success, however, there's more than just his own cunning, as he was guided throughout its campaign by its advisors, actively working and managing his web reputation and, by doing so, effectively winning his presidential campaign in his stead. In particular, as reported in [1], it can be found that deep within the layers of Trump's 1699-page disclosure report were items that added another layer of mystery to his campaign. A total of $35,000 was paid to "Draper

Sterling" for web advertising services. The three $10,000 payments and single $5,000 payment were disbursed from the campaign's American Express account to the entity in a single day. The FEC report (i.e. the report monitoring campaign finance data) also discloses an additional $79,500 was spent on Facebook web advertising[29]. This is of course a precedently undocumented case, that however proves the importance of having a strong web persona in the era of digital information as Donald Trump's presidential run of the election campaign might just be the best example of successful marketing, both by conventional and online media.

Web Reputation Management is a service often offered by many web firms as a comprehensive package alongside web marketing, management of contents, web design and many others. Such firms often employ at least a web marketing expert to take care of content creation and management as well as to account for managing of social media accounts tied to the client. The are currently a number of firms operating in the Online Reputation industry, the most famous examples being: **Online Reputation Services**, **Internet Reputation**, **Reputation.com** and **BrandYourself**.

Of course, online reputation management is a big enough topic to give over to an unknown management company. Thus, before considering any solution, clients should weight which factors should account when selecting an organization for effective reputation management solutions, as they usually offers many services. Anyway, without accounting for differences in pricing and approaches, all online reputation management firms usually employ a similar set of activities and techniques.

4.1.2.1. Web Reputation Activities

When considering a web reputation management activity of any sort, three factors are generally examined before starting:

- **Where** does the person or business has web presence?
- **How** is each web presence representing the subject?
- **What** is being published?

These questions hold a fundamental importance in determining the next steps and are usually examined with the use of specific software. Depending on the type of media to monitor, software employed in managing and monitoring one's online presence are either called **Web Presence Management Systems** or **Reputation Management Systems**. Both kind of tools have been developed to manage one's online presence; however, they differ in the kind of media they can manage:

- **Web Presence Management Systems**: are systems able to monitor and manage owned media. More specifically, Web Presence Management knows about *where* an individual or business is represented

[29] Trump Web Advertising Campaigns, https://vilvas.com/digital-marketing/trump-web-advertising-campaigns

- **Reputation Management Systems**: are systems able to monitor and manage earned media. More specifically, Reputation Management is knowing *what* is being said about an individual or business[30]

Common to both kind of systems is the ability to monitor the web presence of a subject, be it through *owned* or *earned* media. This ability is instrumental in strategically planning the next marketing campaign or online move. Before applying any strategies or techniques to manage the online presence of a subject, it is therefore important to be able to monitor its online fingerprints by answering the three questions above.

Generally speaking, both management systems can be included under the umbrella term of **Online Reputation Management System** and both include several modules working in order to perform tasks related to:

- **Discovery of web presence**: handles the process of discovering new unprecedented points of presence in order to appropriately handle it in case of unauthorized use or malicious intent on part of non authorized authors
- **Brand protection and management**: is performed in order to avoid damaging acts against one's brand or reputation by malicious individuals
- **Security**: is usually enforced by monitoring the web presence of an individual of business to allow malicious users to exploit the web presence to convey a social engineering attack of sorts
- **Monitoring**: is the basic process of all activities related to web presence / reputation management. Has previously stated, in can either be performed continuously or sporadically and allows the system to uncover new web presence points or threats to the individual or business

Without going into further details about explaining how each of the following activities ties to the specific module, it is simply possible to state that each management module is comprehensive of a series of activities that can be described as the set of actions aimed at:

1. Creating a web persona from scratch if none previously existed
2. Engaging in social media content creation and monitoring
3. Creation of periodic reports to inform the client
4. Identification of areas of improvements
5. Strategic planning
6. Review Generation
7. Pro-actively engaging positive and negative reviews by posting a reply
8. Online presence boost
9. Acting on the content in order to obtain some general and specific improvements
10. Specification of parameters to evaluate the quality of the action
11. Optimize content for indexing
12. Press release monitoring and management

[30] Web presence, https://en.wikipedia.org/wiki/Web_presence

It is worth noting that many of the above points might benefit from the use of automated tools. In particular, tools to obtain useful information from the web, evaluate its content and provide some sort of automated reports could somewhat provide additional help to those in the field of Web Reputation management and be of value to their clients as well.

4.1.2.1. Techniques

Techniques used to improve the reputation of clients are usually comprised of a mixture of marketing, public relations, legal and search engine optimization (SEO) techniques. The following techniques are usually employed by online reputation management firms:

1. Social media setup and management
2. Creation and maintenance of a personal website
3. Be always up to date with the topic of the day (or week, or month), creating content based on it
4. Employing SEO (Search Engine Optimization) techniques to qualify for better search ranking results by adhering to the search engine policies
5. Regularly submitting a new sitemap to search engines based on the new content
6. Removal of unwanted content whenever that is a feasible strategy to employ
7. Using gray hat techniques to act on the search engine page ranking by providing intentionally fooling data to search engine, leading it to believe that the page has somewhat reached a higher ranking that it actually possesses.

Generally speaking, commonly used management techniques can either be grouped into activities for managing *existing content* or to create *new* content. The techniques mentioned above easily fit into either one or both categories, as displayed in the table below.

Existing Content	New Content
Social media management	Social media setup
Maintenance of personal website	Creation of a personal website
Use of SEO techniques	Create content based on currently trending topics
Removal of unwanted content	Gray hat techniques
Gray hat techniques	

Table: web reputation management common techniques

Online reputation management firms often engages in both areas, by managing existing content and creating new ones. All mentioned techniques have an impact on assets reputation management however, the most used techniques are Search Engine Optimization techniques, Social Media Related activities and Gray hat techniques when the need and the opportunity arise.

4.1.2.2. Page indexing altering techniques

Amongst the main tasks of an online reputation management system is that of taking care of content displayed as results of search engine indexing algorithms. As briefly seen in Chapter 1 of this document, indexing algorithms greatly differ between different engines and most of the influencing factor behind their ability to properly index pages is regarded as industrial trade secret.

4.1.2.2.1. SEO - Search Engine Optimization

Search Engine Optimization is regarded as the process of affecting SERPs pages to either:

- Pulling up a website
- Pushing down negative or unwanted results / websites
- Pulling up positive results / websites

SEO is an instrumental part in making a business gain more Internet visibility and traction, as being more up in the list of results pages or being indexed more often translates in the possibility to gather more visitors to the website and, in turn, to turn them into consumers. SEO can aldo influence how people perceive an individual or business by pulling up positive results and pushing down harming results. SEO considers how search engines work and the keywords entered by potential customers, performing an optimization of the indexing capability of the website. There are several available tactics to perform optimization but SEO experts need to select the proper ones in light of the more recent search engine changes. As search engines indexing factors often changes, SEO has to change alongside and techniques that were valid earlier could become deprecated in a matter of months. Just by considering Google's indexing algorithm is possibile to point out how SEO need to evolve to match changes to the algorithm. A first basic tactic used by early SEO experts in the early days was based on enhancing the number of backlinks (i.e. the number of links pointing to a page) a page had by employing the so called *backlink building*:

- **Backlink building**: is a tactic directly inferred from what it is publicly known to be a part of the *pagerank* algorithm. The algorithm itself positively evaluates pages with a relevant number of links to them (as this kind of pages are usually good ones). The tactic was fairly simple: post a link to the to-be-optimized website on any website that could be even remotely related to the topic of the site in the first place. This had to be done in a way that it would not be considered spam, thus backlinks were usually posted on website that needed content, in exchange for the visibility they were giving for free

As SEO practices evolved, Google evolved with them in order to increase difficulty in altering SERPs pages (this is very important: Google is not at all ok with people artificially trying to influence its algorithms). In the latest years, Google also regarded keywords as the main discerning factor by which to index a page while, in the more recent past, semantics of content has started to rule the way pages are ordered in SERPs. Page content has come to hold a much more fundamental importance in light of so called *content relevance* paradigm, that is the correlation between keywords and page content. The growing relevance of the

semantics of content has of course diminished the importance of backlinks and pure keywords in the indexing algorithms rendering much less effective techniques such as *backlink building*. As of this year of writing, the number of factors influencing the indexing algorithm have also become more user centered, as Google began valuing so called *user signals* amongst other factors. The interaction between users and the content has become a very important factor in ranking a website by use of a few effective indicators such as:

- **Click through rate**: which simply is the percentual number of clicks measuring the number of people actually visiting a website listed in the SERPs
- **Bounce rate**: is a factor measuring how many people return to Google after visiting a SERPs result but without visiting any additional URLs of the the domain
- **Retention time**: is an important indicator measuring the *time on site* per user (i.e. how much time the user spend on the website)

More recently, factors influencing the positioning of a page in the SERPs has come to be usability of the websites (i.e. its performances, security and design) as well as the adoption of the HTTPS protocol for security.

4.1.2.2.2. When SEO is not enough

As pointed out by this brief excursus on common SEO practices, the increasingly difficulty in promoting a website often creates the need for more under the hood practices. Depending on the morality of the web firm, there are a number of off SEO practices that can help bumping results in SERPs pages. These techniques are known to be a variation of a simple concept, that is the emulation of organic traffic directed to the website. Traffic to websites can be emulated by either:

- Paying a web traffic generator firm to have their staff actively searching google and clicking on your results in order to move them higher up in the list
- Using automated tools to emulate organic traffic

Questioning the web firm morality even more, the more extreme methods of **Negative SEO** attacks could also be employed to, for example, damage competition thus indirectly promoting the target content. These practices usually twist and make use of search engines guidelines to fool the indexing algorithm into giving an incorrect ranking to the competition websites. Google guidelines, are, for example, very strict on defining what is expected of webmaster in terms of their behaviour. Google is unforgiving of webmasters trying to artificially improve ranking of their websites while also valuing content of unique nature. In the case of Google, negative attacks could feature the following techniques:

- Artificially direct a large number of low quality links to the competitors, thus fooling the Google's algorithms into thinking that the author of the website is artificially trying to improve its ranking, resulting instead in a penalty that will need to be countered by link audit and cleanup techniques in order to be lifted[31]

[31] Legality Of Negative SEO, https://www.rebootonline.com/blog/is-negative-seo-illegal/

- Duplicating content of the competitors website in order to fool Google's algorithms can lead to Google considering the duplicated content as being the original one, thus pushing down the competitors website

4.1.2.3. Ethics

Reputation management has raised some concern over the years for the possibility that it might employ non legal means to hit its objective or that, on the other hand, it might employ fully legal techniques to cloak bad opinions under an amass of positive looking data. For example, companies have been known to have hired people to publicly their business without publicly disclosing that they were being paid to do so. Other companies have hired staff to push positive reviews of their products on famous e-commerce websites. The most concern, however, is raised when web firms employ under the hood practices in order to emulate, for example, organic visits to their websites or automatically generated reviews or to attack competition by negative use of SEO and search engines guidelines.

Even though no official regulations exists as of today, the line between legal and illegal seems to be often crossed in these type of activities and future regulations might be needed in order to keep morally questionable web firms and webmaster to employ unethical techniques.

5. The Companion Tool

Following the discussion of the theoretical framework proposed by this work in *Chapter 3 Explorative Algorithmic Techniques* this final inquiry will focus on the implementation of the concepts, first exploring and discussing the architectural choices and finally moving towards an analysis of the results.

5.1. An analysis of the work

The tool is developed with the purpose of being a simple, yet effective monitoring asset for the web marketing force working towards a better online reputation for their clients. Not only that, it also aims at providing the casual user with informative results on what opinion the Internet might hold in his or hers regards. With that in mind, the development of the software followed a simple approach: its users should feel empowered when using the "product", receiving supporting information and data in the form of charts and reports. Also, they should know what to expect when using it and what their following course of actions should be to better their online reputation, if they are not yet actively pursuing such activity. Following these guidelines, the tool development was also oriented towards simplicity of use, online availability and a clean on-request service approach that pushed the development towards a clean web implementation of the framework that could be available to anyone by use of a web browser.

5.1.1. A structural problem analysis

As will be later illustrated by this chapter, the development cycle followed two important stages. Whereas the tool started as a management utility and a fairly simple project, it was later followed by another iteration whose increase in complexity and need for manageability corresponded to the need of finding a way to analyse and structure the scenario in manageable units, modules and sub-problems. This need prompted the renewal of the project's foundations, its core functionalities and libraries but, more importantly, introduced **Data Science** techniques as a way of structuring the context in which the tool would operate.

By use of analysis techniques and tools provided by modern disciplines such as **Data Science** the problem of implementing a working solution was first analysed in order to better the understanding of the domain of data as well as to better define the problem's settings. As already introduced in *Chapter 2. Data science: a brief exploration of the principles behind data*, Data Science is a scientific approach to problem analysis and solving, characterized by following a step-by-step approach that guides the user of said principles from the initial definition of the problem to the presentation of results. The real strength of using this approach resides with its scientific foundation and the possibility to micromanage the difficult stages of the process preceding the implementations of a solution.

5.1.1.1. Getting the Data

The premises behind the Framework implementation have already been made clear by everything that has been stated in the document, thus the first step in Data Science in this scenario is to define a way to gather the data. The outcome of this step is pretty straightforward when taking into considerations the proposed usage of the tool. As the user wants to gather opinionated information about a certain subject, the most feasible way is by means of exploring content indexed by search engines.

It is worth recalling the considerations made in *1.3.2. What does the "query" say*. As depicted in the paragraph, there's some information to be considered already at SERPs (Search Engine Result Pages) level. This is easily portrayed by this example image of a Google search for the "Simpons", showing many types of information already on its own.

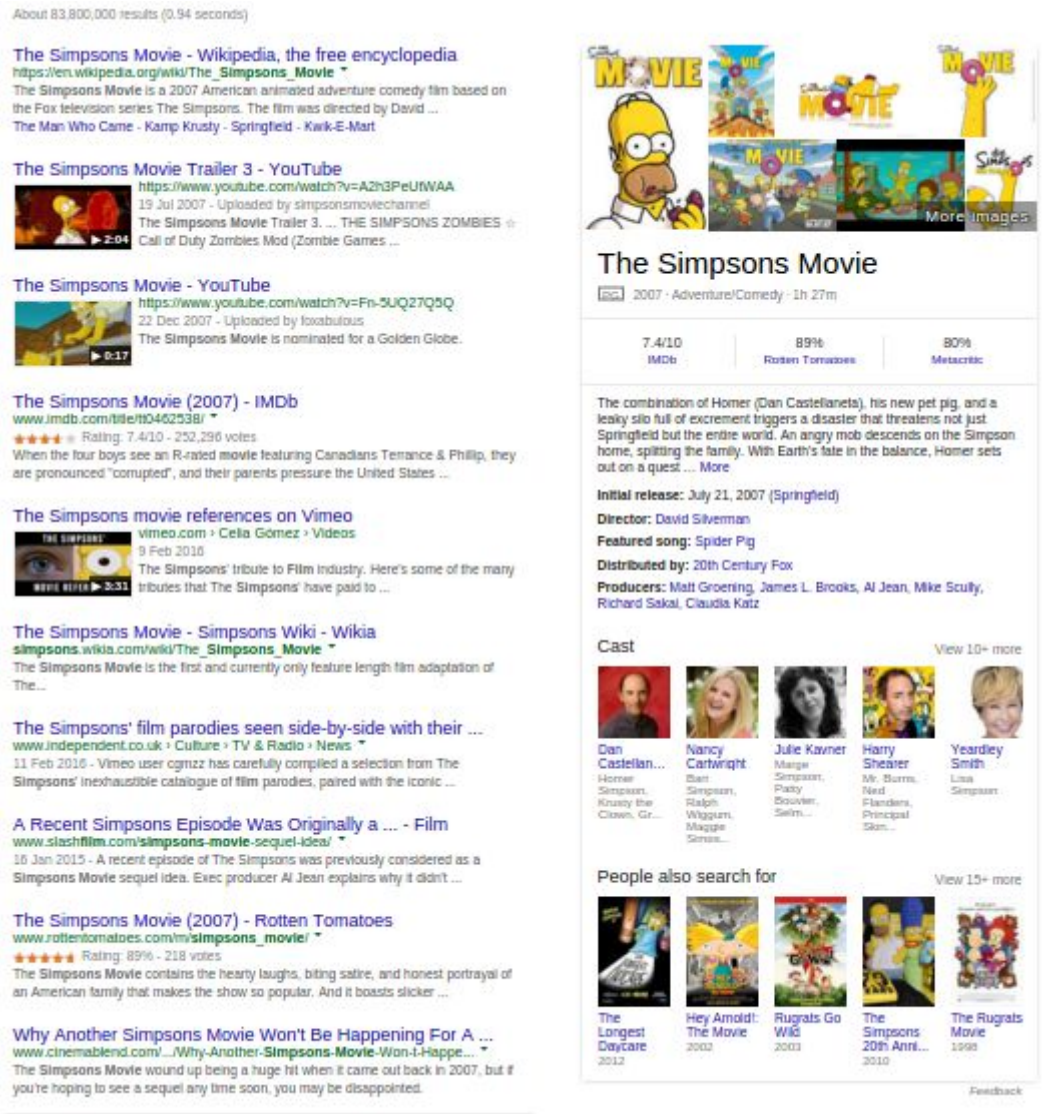


Image: a closer look at a Google's SERP

Specifically, the brief description which accompany each “classic” results (i.e. non multimedia results such as gallery of images or videos) can already be considered a possible source for information gathering as it often provides a glimpse of the real content hosted by the corresponding webpage. However, performing any form of natural language processing becomes increasingly more meaningful when taking into account all the content. This essentially means finding out more information about what is being indexed by search engines, thus resolving into exploring the actual content of this pages by following the links to the effective content. This consideration lead this step to considering *Web Scraping* (over Database searching) as a primary tool to gather information for the system to compute for Web Scraping is a more convenient mean of interacting with the great quantity of information the World Wide Web holds at everyone's disposal.

5.1.1.2. Data Exploration

The second step (Data science *third* step) is data exploration. This step is useful in understanding what operations are allowed with the data in either their raw or transformed form. Web documents are characterized by a freely organized structure that is reflected by the difficulty in analysing a web page by simple algorithmic means (as it will be later shown). This categorizes them into the **nominal** level of data in their raw form, making very few operations available while working on them, as web documents are written by use of several markup languages (e.g. HTML, CSS) as well as client-side programming languages (e.g. javascript), thus making it difficult for analysis and scraping tools to perform well. It is worth recalling that, at this level, data can be described by “name” or “category” only, which leads to some consequences. In particular, these categories:

- cannot be nor ranked or compared quantitatively
- feature a mutually exclusive characteristic as one item cannot be a member of two categories at the time
- they are also exhaustive, meaning that there is a category for any possible case

Chapter 2. Data Science: a brief explorations of the principles behind data however, provides the means to move a little further in the data level hierarchy (in the hope of being able to do much more with said data): pre-processing techniques. These difficulties can in fact be dealt with by use of analysis algorithm that can uncover the main document content. Furthermore, modelling of a document’s content can be achieved by, for example, employing the bag-of-words model, which makes the information **organized** in a tabular like form, thus moving data towards the **ordinal** level, allowing it to be ordered and analysed with a finer granularity. This transformation relies on several algorithms that need to be able to:

- extract the main document content from the cluttered original HTML structure
- tokenize the extracted information into a form suitable for further organization and analysis

These algorithms - of course - exists and will be explored in further details while examining the implementations of the framework functionalities.

5.1.1.3. Modelling Data

The model used in order to transform data for this task needed to focus on the ability to classify information based on its features. In order to perform this task, a number of algorithms have been taken into examination, however, the **Naive Bayes Model** displayed interesting features that made shine even in light of more precise alternative. In particular, the Naive Bayes Model is characterized for[32]:

1. Easy and fast to predict class of test data set. Also, performs well in **multiclass** prediction
2. When assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression and you need less training data
3. It perform well in case of **categorical input variables** compared to numerical variable(s)

Thus, the use of this model has been chosen for its simplicity and acceptable error levels when dealing with predicting the right class for a document, while also accounting for the fallacies related to the the probabilistic theory foundation behind it by providing additional "smoothing" techniques as well as integrating the model with the **influence score**.

5.1.1.4. Presenting Findings

Presentations of findings is a defining step in the context of the application as all of the previous iterations have little to no meaning if results are not clearly explained to the users. This can be accomplished in many ways, by use of both textual and graphical information. In order to achieve the best possible result, the system can return a report featuring both charts and general information about the results. Since most results featured in the application will span across several time spans, the chart to be used when presenting data is the line chart, which is particularly well suited for this kind of tasks. For example, a possible chart depicting the variation of page ranking results could be as follows:

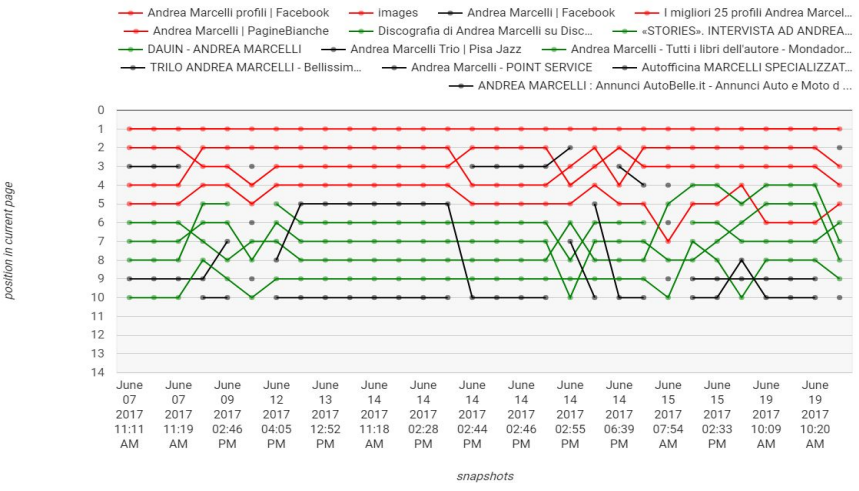


Image: a line chart illustrating pages SERPs position variations over time

[32] Naive Bayes. Unfolded, https://medium.com/data-science-group-iitr/naive-bayes-unfolded-b2ab036b42b1

The implementations of the considerations made while analysing the problem's settings will be analysed in more depth later in the chapter.

5.1.2. Tool's architecture

Working by the requirements the most simplistic approach would have been that of implementing all business logic alongside the presentation layer, in a single application, creating a simple self-containing web application that could be run in every server environment. This approach had the positive feature of being very simple and clean while delegating some of the computational demanding tasks to client side processing. On the bad side of things, however, the nature of such an application would then limit the flexibility of the project as a whole, resulting in issues in changing the algorithms running behind the scenes, being them coupled with the presentation layer.

Following these considerations, development moved towards a different (proper) architectural pattern by taking into account the development of a RESTful Server and consumable REST APIs for requesting services. This approach is able to decouple business logic more effectively from the rest of the application while also providing a centralized - yet flexible - way of accessing the service advanced features. With this approach, the project flexibility increases in the way its services can be used by the client to create even more functionalities on top of the provided ones. On the other hand, however, building a RESTful server delegates most of the computationally expensive tasks to the server itself.

Both models features both the good and the bad of their respective implementations and have been used in different stages of the project's lifecycle. The first approach, that was used during the project's infancy and first experimental phase will be later expanded upon in the section dedicated to the file-based implementation of the tool. While the second approach marked the shift towards a more "professional" phase of development that also involved moving the system towards a database-based implementation.

5.1.2.1. Early development

At early stages of development the tool was conceived as a way of monitoring web pages positions concerning certain results. These web pages would then be ranked for either good, neutral or negative content concerning the search subject and, as the data was collected overtime, the information would then be used to draw an opinionated web profile for the entity being analysed. The initial idea was that of providing marketing professional with a monitoring tool, allowing them to search the **Google** search engine for results about the person or business they needed to take care about while also being able to evaluate the results in the SERPs (i.e. search engines resulting pages) for content polarity, being it either positive, negative or neutral, thus allowing them to draw an online reputation profile based on their own manual evaluation, print it and come back at it at different times of the weeks in order to see any changes as consequences of their work. The software was useful in collecting data but the manual process involved in its evaluation and tracking of information certainly was a major drawback. This being a big concern, the next iterations in the development cycles implemented a file based storage of information in order to maintain

historical data to compare with when searching Google for the same query as previously done. Information about the set of pages and evaluations were thus stored in JSON files and later recovered for allowing the user of the tool to draw charts based on the information, visualizing the actual ranking tendencies of pages concerning its client's interests. Still, the usage of a file based storage system held development and efficiency of the tool back as most of the computations were devoted to extracting the information from file, properly formatting it for the routines to handle them back to the user in a nice and understandable format.

5.1.2.2. Later stages of development

Following the initial version of the web application the approach moved towards a more serious take on the development and analysis of the activities to be performed by the software. After making considerations on the nature of the software and researching the fields of **Sentiment Analysis** and **Data Science** the software was rewritten from the ground up in order to implement the Sentiment Analysis framework that was presented earlier in this work. By approaching development with the analysis tools provided by Data Science, the application was built in order to provide useful information condensed in easy to understand charts and reports. The natural consequence of involving Sentiment Analysis techniques was that of providing an automated way of analysing text in order for the algorithm to uncover the polarity of an online document. This automation and the Framework functionalities implemented by the application improved on the initial concept by providing more advanced features as well as more flexibility by implementing most of the business logic on the application server side. The RESTful server has, in fact, been developed in order to be easily queryable by either *javascript* or *curl* implementations, exposing a simple public API to be used by previously authenticating with the system. In order to use the functions exposed by the public API a user is expected to implement its own client GUI (Graphic User Interface) or to use the publicly available one developed for use with the tool. From a technical standpoint, the design of the application followed the Model View Controller, which allowed for great modularity and reuse of components. The next paragraph is devoted at illustrating the innerworking of the different system's components as well as their characterization.

5.1.2.3. Component Analysis

The system - as is - is characterized by several components and algorithms whose interactions provide the basic functionalities exposed by the server. Conceptually, the tool can be seen as shown in the following image:

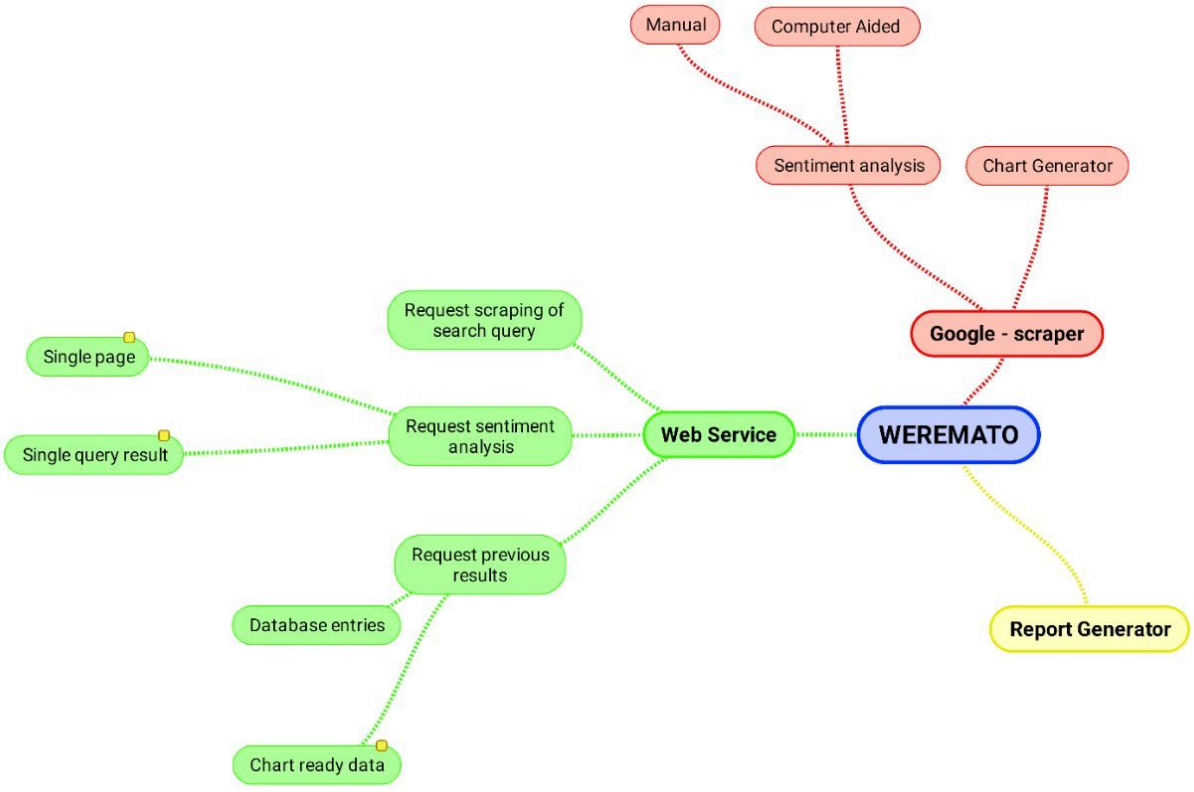


Image: the web application components illustrated by a mind map graph

As it can be seen illustrated in the graph in the above picture, the main components of the project can be identified in the **Web Service**, **Google Scraper** and **Reporting** services. The first two services are used in order to gather data from the web (specifically through use of the Google search engine) while the latter is a presentation utility used to provide the user with portable results without having to rely on the web browser.

Web Service

The Web Service designed for the tool has been implemented following the REST principles. The idea behind a *RESTful* web service is based on the on the concepts introduced by Roy Fielding with his proposal of a **REST** pattern described in his paper “Architectural Styles and the Design of Network-based Software Architectures”. REST stands for Representational State Transfer and can be better described as a series of principles primarily used in

designing highly scalable, and efficient web services[33]. The driving idea behind REST principles are *resources*. Resources can represent anything accessible and exchangeable between a client and a server. While REST is not specific to the World Wide Web, its implementation over the HTTP protocol has become the most concrete example of the architectural pattern. The definition is extended during the interaction between server and client where Fielding defines the exchange as the act of transferring the internal state of the resource (i.e. its representation)[34] .

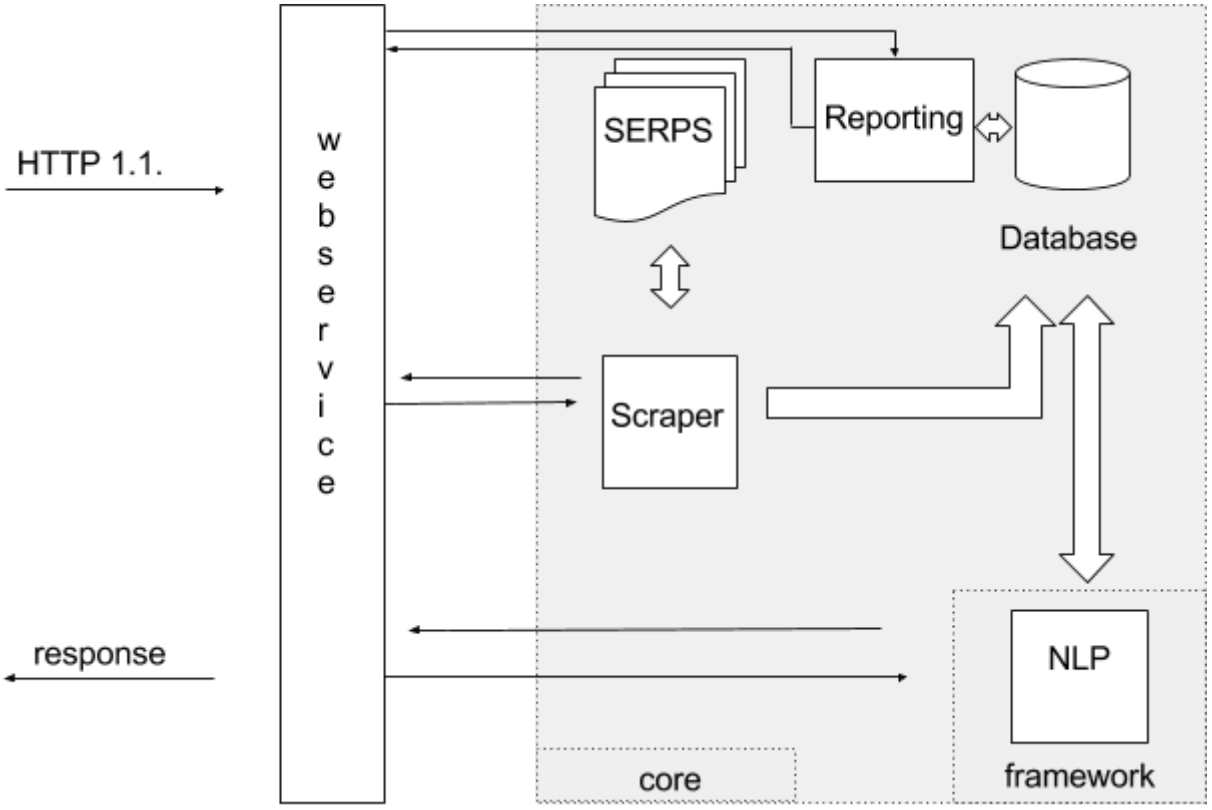


Image: tool's functional architecture

The role of the web service is quite simply that of a gateway to the application's core functionalities. Each functionality can be reached by combining the appropriate HTTP verb[35], which is mapped to a so-called CRUD (Creation Reading Update Deletion) action:

[33] RESTful Web Services: A Tutorial, http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069
[34] Roy Fielding, Architectural Styles and the Design of Network-based Software Architectures
[35] Using HTTP Methods for RESTful Services, http://www.restapitutorial.com/lessons/httpmethods.html

HTTP Method	CRUD Action	Description
POST	Create	Create a new resource
GET	Read	Obtain an existing resource
PUT	Update	Update an existing resource
DELETE	Delete	Delete a resource

Table: HTTP mapping to CRUD actions

This mapping is typical to RESTful web services but is however not always strictly adhered to as some implementations prefer to always use GET methods to implement interactions between users and the server. The mapping itself is often associated to an *htaccess* file (in case of an **Apache** web server) whose task is that of re-mapping the URLs to more specific internal routes. As a matter of fact it is common practice to expose elegantly phrased URLs in the web service APIs while retaining a more verbose and practical syntax on the server side. Consider the following two examples:

```
# Return the result set for a certain query (GET) or add new content
(POST)

RewriteRule  ^data/content/([a-z\s+]+)/([0-9]+)/([0-9]+)/$requestsHandle
             r.php?target=record&action=content&query=$1&search_page=$2&
             amount=$3&encode_string=false [nc,qsa]

# Same as above but with explicit support for latin set characters

RewriteRule  ^data/content/([a-z\s+]+[áâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿÀÁÃ
             ÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞß]+)/([0-9]+)/([0-9]+)/$requests
             Handler.php?target=record&action=content&query=$1&search_pa
             ge=$2&amount=$3&encode_string=true [nc,qsa]
```

The first and the second rule perform the same basic task: they map one incoming URL with certain features to another URL for internal use. They do these by applying a `RewriteRule` that instructs the server to perform the mapping. Thus, by using the REST principles alongside the expressiveness of URL syntax it is fairly easy to develop a service matching simple APIs with functionalities in a straightforward manner. The principle behind this approach is simple: whenever a certain service is needed, the user calls upon the web service to execute the appropriate internal module by simply using the right URL that basically behaves as an API. The public APIs to the web service are summarized in the table below:

HTTP Method	API	Modules	Input	Output
POST	data/**content**/([a-z\s+]+)/([0-9]+)/([0-9]+)	Database	query_id, page number, amount	Add new content for a certain query, run on a certain page of the SERP holding a certain amount of data
GET	data/**list**/	Database	-	A list of all previous results for the user
	data/**content**/([a-z\s+]+)/([0-9]+)/([0-9]+)/		query_id, page number, amount	A list of previous user's results for a certain query, run on a certain page of the SERP holding a certain amount of data
	data/**content**/([a-z\s+]+)/([0-9]+)/([0-9]+)/date/([0-9-]+)/([0-9-]+)/		query_id, page number, amount, start date [, end date]	A list of previous user's results for a certain query, run on a certain page of the SERP holding a certain amount of data either from a starting date or between two dates
	data/**content**/([a-z\s+]+)/([0-9]+)/([0-9]+)/last/		query_id, page number, amount	The last of previous user's results for a certain query, run on a certain page of the SERP holding a certain

				amount of data
	data/**content**/([a-z\s+]+)/([0-9]+)/([0-9]+)/last/([0-9]+)/		query_id, page number, amount, n	The last n of previous user's results for a certain query, run on a certain page of the SERP holding a certain amount of data
	data/**content**/([a-z\s+]+)/([0-9]+)/([0-9]+)/first/		query_id, page number, amount	The first of previous user's results for a certain query, run on a certain page of the SERP holding a certain amount of data
	data/**content**/([a-z\s+]+)/([0-9]+)/([0-9]+)/first/([0-9]+)/		query_id, page number, amount, n	The first n of previous user's results for a certain query, run on a certain page of the SERP holding a certain amount of data
	data/**lookup**/([a-z\s+]+)/([0-9]+)/([0-9]+)/	Database, Scraper	query_id, page number, amount	At least n SERP's items in a list with default values for sentiment and influence score. Results are stored in the database
	data/**nlp**/([a-z\s+]+)/([0-9]+)/([0-9]+)/	Database, NLP (Framework)	query_id, page number, amount	Sentiment and influence score for the url associated to a certain query, at a certain SERP page number holding
			url	

				a certain amount of data
	data/**linechart**/([a-z\s+]+)/([0-9]+)/([0-9]+)/	Database, Reporting service	query_id, page number, amount	A .pdf report with chart and general information about user's results for a certain query, run on a certain page of the SERP holding a certain amount of data
	data/**linechart**/([a-z\s+]+)/([0-9]+)/([0-9]+)/date/([0-9-]+)/([0-9-]+)/		query_id, page number, amount, start date [, end date]	A .pdf report with chart and general information about user's results for a certain query, run on a certain page of the SERP holding a certain amount of data in a certain time period
	data/**linechart**/([a-z\s+]+)/([0-9]+)/([0-9]+)/last/([0-9]+)/		query_id, page number, amount, n	A .pdf report with chart and general information about user's last n results for a certain query, run on a certain page of the SERP holding a certain amount of data
	data/**linechart**/([a-z\s+]+)/([0-9]+)/([0-9]+)/first/([0-9]+)/		query_id, page number, amount, n	A .pdf report with chart and general information about user's

				first n results for a certain query, run on a certain page of the SERP holding a certain amount of data
	user/authenticate/	Database	-	Is used to authenticate the user against the system. It should be called before using any other API
			Encoded username and password	
	user/lastoperationtime/		-	Returns the timestamp corresponding to the last time the user registered an activity on the tool
			Encoded username and password	
	user/insync/([0-9-\s*:]+)		date	Returns a boolean indicating whether the user is currently in sync with its last registered activities. Synchronization mismatch are caused by the user accessing the service from different clients which need to maintain a consistent state with the server
			Encoded username and password	
	user/sync/		-	This operation

			Encoded username and password	is called by a client that wants to “pull” data from the server. The call to the API function returns the whole amount of projects created by the authenticated user
	user/sync/([a-z\s+]+)/		query_id	As before, this operation is called by a client that wants to “pull” data from the server. However, in this case, the call to the API function returns data created by the authenticated user for the selected project only that refers to a query
			Encoded username and password	
PUT	-	-	-	-
DELETE	-	-	-	-

Table: Web service APIs

Most HTTP verbs were not used when developing the API layer in order to ensure compatibility with client API implementations in Javascript. In fact, for obvious security reasons, there currently exists no possibility to perform any cross-domain POST, PUT or DELETE client side, while, on the other hand, GET requests are supported via CORS[36] or JSONP[37]. For the use with the application web service, calls by either CURL or JSONP

[36] Cross-Origin Resource Sharing, https://www.w3.org/TR/cors/
[37] JSONP, https://docs.microsoft.com/en-us/dotnet/framework/wcf/samples/jsonp

clients are currently supported. Furthermore, most results can be returned in either of two formats:

- **JSON**: a json object is returned when including the "`Accept: application/json`" header in the request
- **HTML**: an HTML table is returned when including the "`Accept: text/html`" header in the request

Additionally, when requesting the services of the Reporting Service, the output format will either be:

- **Encoded PDF**: when including the "`Accept: application/json`" header the pdf will be encoded in base64 and included in a JSON response
- **In browser PDF**: when including the "`Accept: application/octet-stream`" header, the pdf will attempt to render in a browser window

Examples of the output of each services called via the web service API can be found later in the document, when examining the different usage scenario for which the application can be featured.

Google Scraper

The Google Scraper module is a fundamental part of the tool. Its function is, quite simply, that of acting as a in-code browser for querying the Google search engine. The scraper allows to select different options to query the search engine, starting from defining what headers to sent to the web page and, ultimately, the language and localized version of the engine to be used. The scraper can be customized to return a certain type of results only, like for example, only media results, organic results or all of them together. In the project, it is simply used to perform a search on google by personalizing it with three different parameters:

- **Query id**: the query itself. A collection of terms that should be used as search keys by the search engine
- **Search Page**: the SERP page to be searched. Google can be asked to search in a specific SERP page. The default value for this is, of course, the front page of results
- **Amount**: the amount of results per SERP page to be included

By using these parameters, the Google Scraper module initialize a CURL request to the google search engine url, returning the values and several other informations that could be computed, exploited and stored for future reference.

Reporting Service

This service takes care of presenting data to the user in the most exhaustive way possible by combining both descriptive text, numerically relevant data and charts. By considering the type of data at disposal, the focus of the reports should be on summarizing pagerank variations overtime by also visually combine it with the sentiment and influence scores for each different page. In order to better do so, a system of colors - reminiscent of the DEFCON color schemes - was devised to convey information in a simple manner:

Polarity	Influence Score	Color Scheme
Negative	> 0.7	red
	0.5 < AND < 0.7	orange
	0.25 < AND 0.5	yellow
	< 0.25	beige
Positive	> 0.7	green
	0.5 < AND < 0.7	white
	0.25 < AND 0.5	purple
	< 0.25	blue

Table: The DEFCON table

In this way, the additional information conveyed by use of the influence score can be successfully visualized without hindering the overall structure of the chart, keeping it simple. The reporting service has both a client side and a server side implementation, both relying on using the Google Chart Visualization APIs for rendering charts on the fly via Javascript. The workflow for creating charts is the same for both implementations:

1. Data is collected and prepared
2. Colors are matched using the DEFCON table schema
3. The wrapper function for Google Charts is called and the chart rendered on the page

Things of course differ whether working client or server side. In order to achieve a rendered result when working in the server environment the web service makes use of a headless browser (PhantomJS) to visit a web page that will render the chart with all the information provided to the page. Fortunately enough, the browser also includes functionalities to

download the displayed page as a PDF. This feature is used when needing to provide a full report to avoid making use of any additional third party library for creating PDFs. The report itself is created on a A4 template model (300 dpi) which is essentially comprised of two main sections:

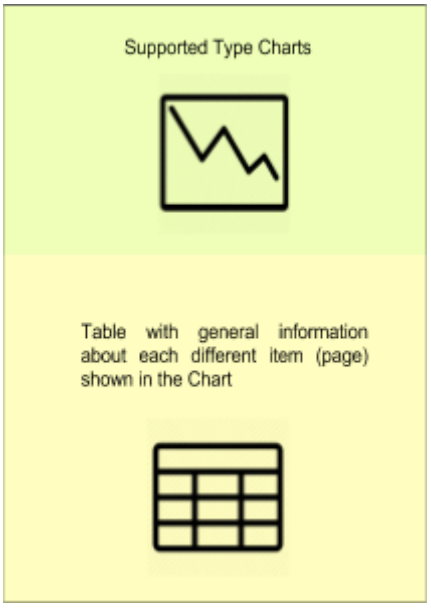


Image: report template

The content of the report is dependent on the template. The default template includes two charts, one describing the over time variations for page positions and an additional chart for describing the reputation level of the single page. Furthermore, the report contains a table which further describe the result set by adding information on the average, maximum, minimum values obtained by the page (over time) as well as the standard deviation as a measure for describing how a single entry can be expected to variate.

NLP

This service is at the core for providing an automated evaluation for the content associated with a unique url and, thus, an entity (subject). The implementation strictly relates to what was introduced with the framework adjustments of *Chapter 3. Explorative algorithmic techniques*. It introduces several methods for attributing a reputation score to the entity related to the document as well as, more generally, to define the sentiment polarity for the whole document content. This module also features the ability to distinguish between directed and undirected sentences, by employing a “label” or “synonym” list, as per the framework definitions. Optionally, the NLP module can employ “*Named Entity Recognition”* (NER) for distinguishing between multiple entities in a sentence. This is useful for documents characterized by a succession of entities in the same sentence, which could be the target for an opinion or, more generally, for the generic polarity of the statement. Benchmark for the execution of the NLP module with the optional NER activated shown, however, that gains in accuracy are risible while performances take a deep slope so it is turned off by default in the default tool implementation.

Dashboard Interface

Even though the web service can be accessed by use of any HTTP capable entity (such as, for example, software and browser add-ons for mocking HTTP requests) sending GET and POST requests to the API, there's a definite need for a more visual approach, tailored to the user and capable of simplifying the interaction between the latter and the web service. It is often customary, for web services based applications to leave the client's implementations up to the users of the system. In the context of this work, however, a first client based interface is provided in the form of a **Private Dashboard**. The explanation behind this concept is simple: the user, that registered to the system, is given access to a personal dashboard area that grants him access to an implementation based on the following principles:

1. Client side functions are a preferential method over relying on the web service to perform additional computation in the following cases:

    a. Delivering reporting material
    b. Requesting existing data without any computation required

2. Caching of information on the user's space (more specifically: an application folder)

When the client application it's first started it requires the user to access with its personal service credentials and, on successful authentication, it performs the following task:

- **New Dashboard Initialization**: the dashboard is initialized and synchronized with the server by the following procedure
    - the client application requests all existing user data searches to the web service
    - data from the web service is stored in separate time - stamped files on the application directory

Additionally, to maintain consistency between the client application and the web service application the following tasks are carried out:

- **Lookup memorization**: whenever the user performs a search, he / she uses the web service application that stores his / hers data in the database as well as returning them in the custom user defined format. So, in order to maintain consistency across the system, the returned data must also be stored in the application folder
- **Sentiment analysis requests**: when requesting sentiment analysis on a content identified by a url the client needs to interact (via web service) with the NLP modules of the service, therefore needing to perform the following:
    - sending the urls to be analysed to the web service, one at the time
    - updating the corresponding file with the new information received by the server
    - eventually re-initialize the private dashboard area data related to the query

The tool comes with a specific client implementation called WEREMATO. More on the application will be discussed in *5.1.3. WEREMATO Client*.

5.1.2.4. Tool and Web Reputation

Web reputation, as introduced in *Chapter 4. Web Reputation*, is "the reputation deriving from one's presence on the Internet". This tool was specifically developed to be of aid when managing a person's web reputation, and falls under the family of **Online Reputation Management Systems**. Web agencies are often in need of managing the web reputations of their clients by actively controlling what kind of results are being displayed when searching for their client's name on a search engine. This process is often done in a tedious and long winded manner that could greatly benefit from having a semi - automated tool to perform most of the tasks on its own. Enter the "Companion Tool" to the framework: users of the tool can expect to be able to search and evaluate their client's reputation by using the tool, while also being able to visually see the impact of their work on the SERPs.

5.1.2.5. Tool libraries stack

So, what exactly lies under the hood? What makes the different parts of the applications move and work together? The application's library stack is basically composed by the following scripts:

- Serps Spider
- NLP Tools
- Naive Bayes Model for Sentiment Analysis
- PhantomJS

Library	Module	Author	Description	Reference
Serps Spider	Scraper	Soufiane Ghzal	Serps is a set of tools that ease the parsing of popular search engines (such as google, yahoo or bing). It helps to parse SERP (Search Engine Result Page) and gives you a standard output of what is parsed[38]	https://serp-spider.github.io/
NLP Tools	**Nlp** (Framework)	Angelos Katharopoulos	Nlp Tools is a set of php 5.3+	http://php-nlp-tools.com/

[38] SERPS The PHP Search Engine Result Page Spider, https://serp-spider.github.io/documentation/#licensing

			classes for beginner to semi advanced natural language processing work[39]	
Naive Bayes Model for Sentiment Analysis	Nlp (**Framework**)	Nick Duncan	Strings are broken into tokenized arrays of single words. These words are analysed against TXT files that contain emotion words with ratings, emoticons with ratings, booster words with ratings and possible polarity changers[40]	https://github.com/NickDuncanZA/php-sentiment-analyser
Stanford Named Entity Recognizer (NER)	Nlp (**Framework**)	Stanford University	Stanford NER is a Java implementation of a Named Entity Recognizer. Named Entity Recognition (NER) labels sequences of words in a text which are the names of things, such as person and company names.[41]	https://nlp.stanford.edu/software/CRF-NER.shtml
Language Detection Library	Nlp (**Framework**)	Patrick Schur	This library can detect the language of a given text string. It can parse given training text in many different idioms into a sequence of	https://github.com/patrickschur/language-detection#api

[39] PHP NlpTools, https://github.com/angeloskath/php-nlp-tools/
[40] Sentiment analysis and adaptive phrase match learning with PHP, https://nickduncan.co.za/tag/naive-boyes/
[41] Stanford Named Entity Recognizer (NER), https://nlp.stanford.edu/software/CRF-NER.shtml

			N-grams and builds a database file in JSON format to be used in the detection phase.	
PhantomJS	Reporting	Ariya Hidayat	PhantomJS (phantomjs.org) is a headless WebKit scriptable with JavaScript[42]	http://phantomjs.org/

Table: Application's library stack

For the proper development stack, the web based tool has been developed on a LAMP (i.e. Linux, Apache, MySQL, Php) stack for its simplicity of development and also reliability. Although this components were not originally written to work together they have, over the year, developed a big affinity towards each other and extensions have been developed to improve their exchanges as together they provide the base for many web applications:

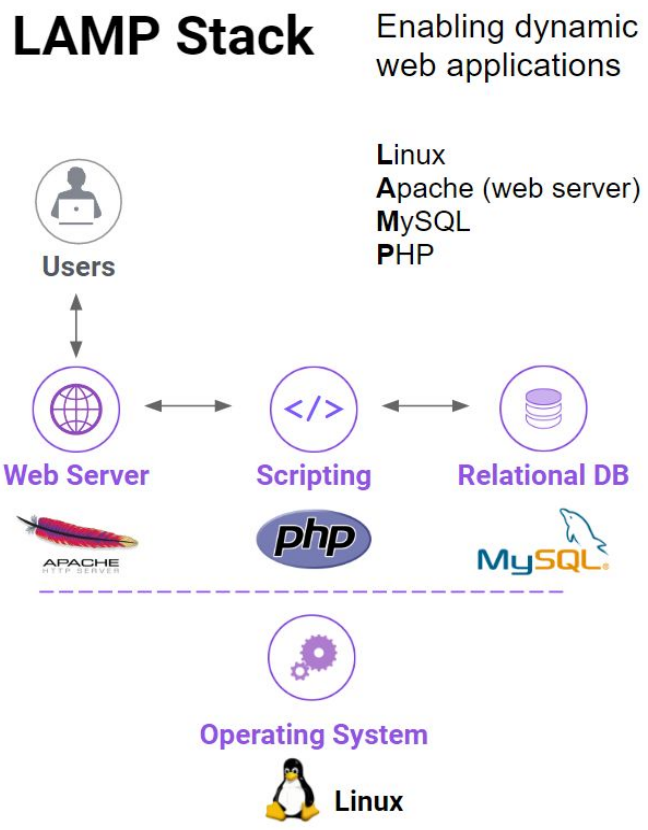


Image: the LAMP stack visualized[43]

[42] PhantomJS - Scriptable Headless WebKit, https://github.com/ariya/phantomjs
[43] The SMACK Stack is the New LAMP Stack, https://mesosphere.com/wp-content/uploads/2017/06/smack-stack-is-the-new-lamp-stack-comparison.png

In short, a scripting language such as Php was preferred over a compiled language because of its “web readiness” when compared to most other languages and is reliability given its long history serving the web developer community. The same considerations (over reliability and popularity of use) can be made also for the other components of the stack which is - specifically - comprised of:

- Manjaro Linux v17.0.5
- Apache v2.4.27
- MariaDB v10.1.26
- PHP v7.1.9

5.1.3. WEREMATO

Image: WEREMATO logo

The client implementation of the whole application stack was called *WEREMATO* as an acronym from **We**b **Re**putation **Ma**nagement **To**ol. Generally speaking, the tool is, essentially, a non client service that can be used simply by referring to its web interface (REST Api). WEREMATO is a basic interface supporting all possible requests and operations performed via web service while implementing the concept of “private dashboard” as previously mentioned. This allows *WEREMATO* to operate locally when performing operations such as:

- drawing informative charts for the user
- computing the general “health” of the projects
- computing how well the management process is going

The application connects to the web service for authenticating purposes and to be synched with the latest searches performed by the user. This is done in order to maintain consistency between the user local environment and the remote database, as *WEREMATO* account for the user logging in on other computers and / or using other clients to perform the same operations. The applications is composed of the following areas:

1. Dashboard
2. Projects
3. Reports

Each section is used to give informative feedback to the user and is discussed in the next section.

5.1.3.1. Dashboard

The Dashboard area contains information about the general projects health and simply informs the user of the application on how well things are going for its clients (i.e. the subject of its queries). These values are computed by examining and computing all reputation values for all projects, which give a precise summary of the situation. This way, the user can document the way in which its activities can affect its clients by taking a simple look a the charts displayed in the page

5.1.3.2. Projects

This area displays in a table all past projects created by the user. The strength of this tool section, however, is the so-called “Time View” function, which describes the by time variations of a search result pages in their position with respect to the query. This, combined with charts gives a unique view on how the pages have varied in position and what this means for the subject of the query. This can be made clearer with just a simple look at this example charts:

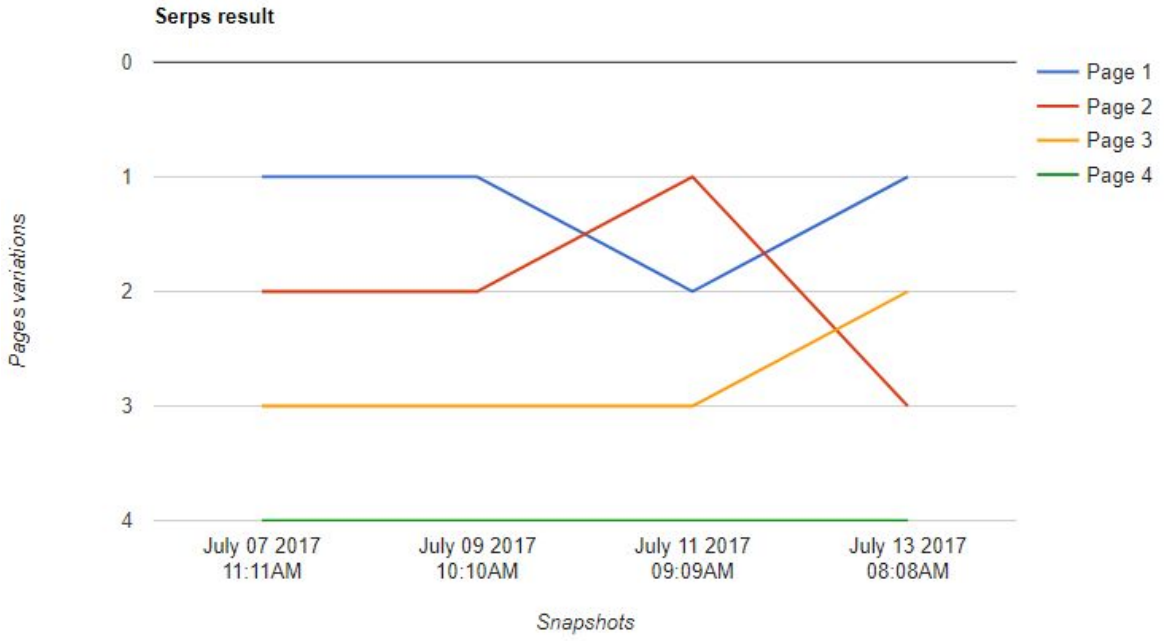


Image: page’s SERP position variation over time

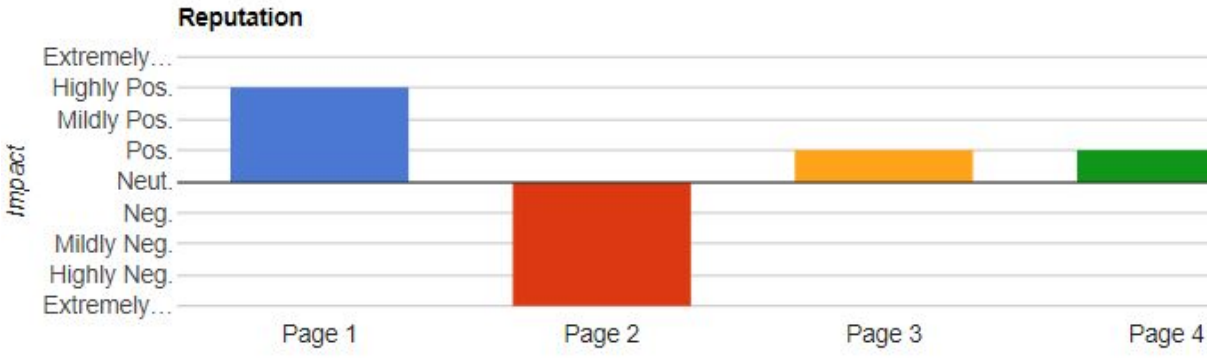


Image: page's reputation score

This images describe a query where the variation describes a positive trend, as pages with negative results are slowly turning away from the first positions.

5.1.3.3. Reports

This section allow users to request reports describing a certain project. The report, as already described, contains both informative charts as well as a table describing additional information that becomes increasingly significative as the amount of data grows. This page simply is a interface for obtaining and viewing reports.

5.1.3.4. Synchronization and configuration

Another important feature central to the client is its ability to synchronize with the server in order to build and maintain a local dashboard area. This client's feature is vital in several scenarios involving user interaction with server without using the client but by other means, as for example other available clients or by directly using its entry points by using a mocking framework as, for example, **SoapUI**, **Postman** or **AdvancedRESTClient**. Synchronization is, however, not done automatically: the user is notified whenever new updates are available (fetch) and needs to add them to its local dashboard (pull). The only instance in which the synchronization process is automatically instantiated by the client is at user's first login to WEREMATO: at this time, in fact, the client fetches all projects previously created by the user and pulls them into the private dashboard by saving them on disk utilizing its private internal representation. While doing so, the client also manages all info regarding the most important part of the data, being the reputation, sentiment, and influence values by storing them into the private user's settings file in the config folder.

A user's configuration file for the application is basically comprised of the following information:

- **lastmachine**: the last operating system from which the user launched *WEREMATO*
- **lastsync**: the last synchronization time as a UNIX timestamp
- **projects**: an array of information related to each single project ever created by the user
    - influence
    - sentiment
    - reputation
    - latest_search

An example of the file is the following:


```
{
    "config": {
        "lastmachine": "WIN",
        "lastsync": 1507784267,
        "projects": [
            {
                "andrea marcelli": {
                    "influence": {
                        "1": 10
                    },
                    "sentiment": {
                        "neutral": 10
                    },
                    "reputation": {
                        "uninfluent or uninitialized sentiment and influence": 10
                    },
                    "latest_search": 1506930021
                },
                "donald trump": {
                    "influence": {},
                    "sentiment": {
                        "neutral": 9,
                        "negative": 1
                    },
                    "reputation": {
                        "uninfluent or uninitialized sentiment and influence": 10
                    },
                    "latest_search": 1506554004
                }
            }
        ]
    }
}
```

5.1.3.5. Client architecture

The client is structured around the Model View Controller pattern in order to be maintainable and modular. Generally speaking, all of the client's interactions with the web service as well as within itself (when requesting particular module functions) are mediated using a "middleware" class which behaves as a controller for the whole application. The controller makes sure that all client's requests to the web service are made using cURL in order to be processed and returned to the calling view modules. The architecture is described in the following image:

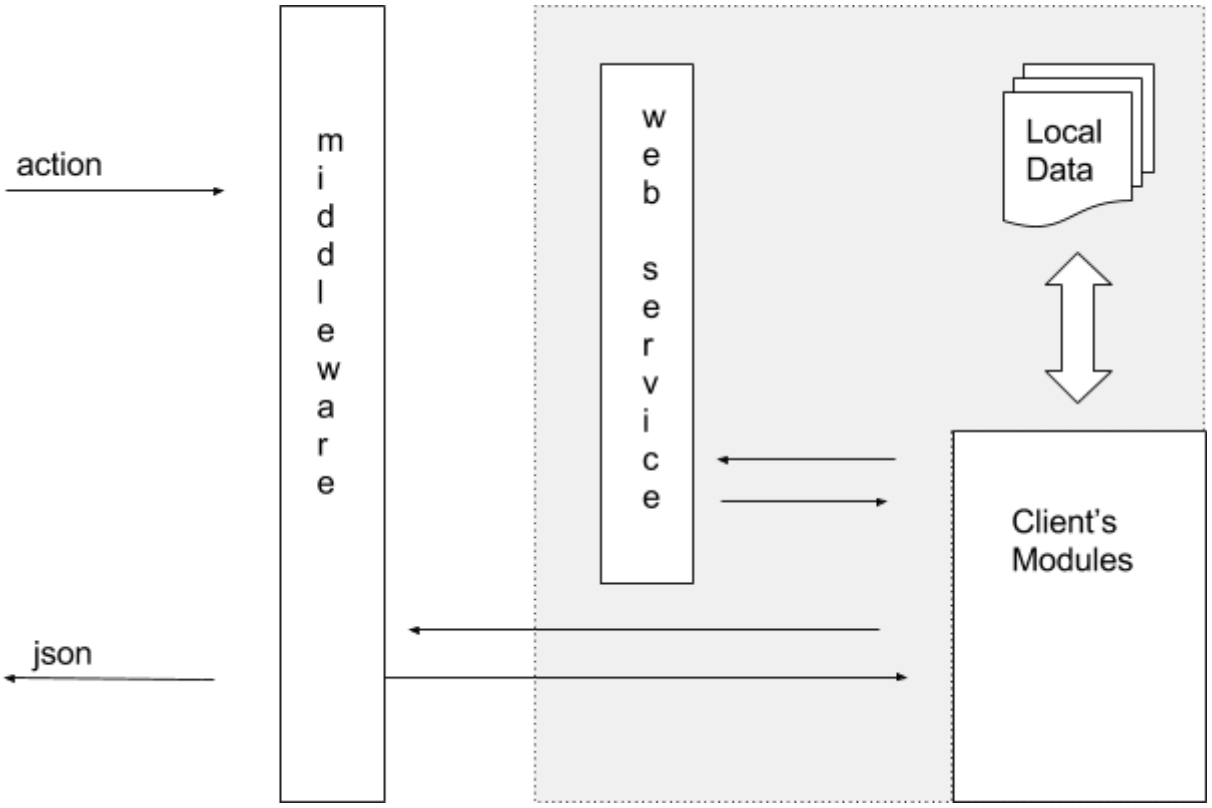


Image: client's functional architecture

The controller takes care of initiating client's call to the web service which are then executed by *WEREMATO* client's modules while all data returned to the views is in json format. This architecture allows to maintain good abstraction and separations levels between the client's business and presentation logic.

6. Conclusions

In this chapter the limitations of the tool - as is - will be discussed, alongside the possible remedies and future features to be implemented.

6.1. Limitations

6.1.1. Framework limitations

The framework limitations are, of course, of theoretical nature. The concepts introduced in *Chapter 3. Explorative Algorithmic Techniques* are a generalization with respect to the intuition presented by experts of the field of Sentiment Analysis such as **Liu**. This generalization comes at the cost of losing accuracy. In fact, the polarity of the document (as a whole), combined with the inferred “influence” (score) with respect to the subject of the inquiry is a good starting point and can be suitable for some general - but limited - settings. Fortunately, this is true in the case of the framework implementation mentioned in the previous chapter, as its utility is a tool is focused towards providing a general indication and not a precise analysis of each subjective opinion towards a specific feature (or characteristic) of an individual. However, as the tool evolves, the framework should too, by steadily incorporating more evolved features, such as those described by **Liu** in its works, and, in “**Sentiment Analysis and Opinion Mining**” in particular.

6.1.2. Technical limitations

The technical limitations to the current software algorithms are due to some particular restrictions that are enforced by search engines in order to protect against web scraping. These Internet giants are - in fact - well aware of the potential hold within their data, and, more importantly, in their public interface (SERPs) that simply are a small windows over the very vast amount of information they collect. Thus, in order to avoid exploitation of even the smallest of information they make available to their users, they enforce a series of strict policies to avoid being “scraped” by intelligent algorithms. In particular, Google enforces a simple policy to protect itself against web scraping: a form of increasing form of punishment to discourage the web scraper to proceed any further. Generally speaking, the punishment mechanism goes through the following steps:

1. Presenting reCAPTCHAs puzzles
2. Temporary banning the guilty ip
3. Suspending all google services provided to that ip

It is unknown how many “scraping” activities cause a script to be detected and how Google determines the “appropriate” punishment to the ips that incurs in said behavioural pattern. They are, however, blocking with regards to any scraping, therefore actively blocking the service provided by *WEREMATO*. Solutions to even the less hindering form of punishments are difficult to say the least but can be addressed in various measures.

1. Addressing the reCAPTCHA(s) is a difficult task as they are purposely made for scripts and tools to be difficult to both detect and solve without any human interaction. The simplest idea, in order to have the script move forward with its analysis would be that of presenting the human operator with the reCAPTCHA challenge in order for it to be solvable. This can be true and effective for many websites, but, alas, the Internet “bigs” always will have a technical vantage point with respect to the majority of lone developers and companies trying to circumvent their well thought “defensive” mechanism:

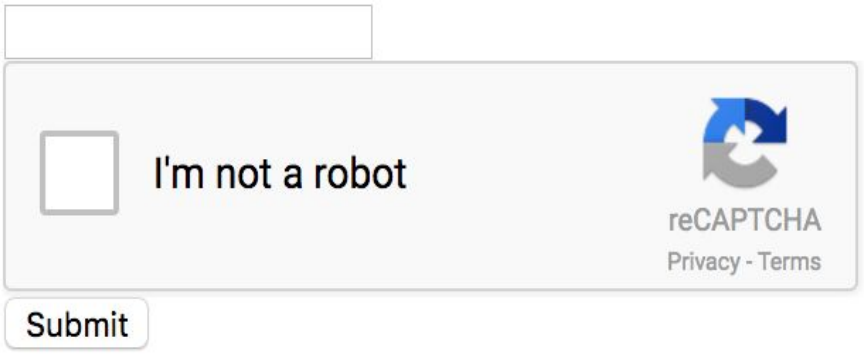


Image: a simple reCAPTCHA simple example

   Google per se, issues a reCAPTCHA at every access to their pages, meaning that, simply passing on the task to the human operator is a non feasible operations as the puzzle will be re-generated, not being valid for the web scraper

2. The temporary blocking of any guilty ip-related activities is simply a matter of waiting a fixed amount of time before attempting another request to the search engine. But, exactly, how long is the right “fixed amount of time” that the script should wait for? This is unknown as it is well hidden into Google’s anti web scraping policies. Preventing, as the saying goes, is better than curing. Thus, the solution is simply to “scrape with gentleness” by not overwhelming the website with huge amount of requests, waiting a feasible “human like” amount of time between requests. However, this is another big problem for what *WEREMATO* is set to do. A possible solution for this could be that of employing a rotating ip proxies schema to execute requests by using different ips, thus being able to continue sending requests to the website. However, this solution presents hurdles of its own:

   a. Ip proxies with the right features are difficult to find. In fact, to be able to pass as a “legitimate” browser, the proxy should at least have the following features:
      i. be a **https** proxy
      ii. be a **Google ready** proxy (i.e. a proxy which is able to correctly render Google pages)
   b. Furthermore, proxies have a big “?” stamped on their owners foreheads. Who actually owns these proxies? And what are they going to do with data flowing in and out of their machines?

Another possible solution can be that of using a headless browser to perform requests. In fact, some of the *anti-bot detection paradigms* employes some sort of “challenges” against the scripts declaring themselves as fully fledged and javascript ready web browsers by setting up their http referer header to a well known browser build as, for example, Chrome 50, which definitely has javascript support. In this case, any form of javascript challenge to a “fraudulent” script would immediately label it as a bot. What can be done in order to prevent this from happening is simply to use the so called headless browsers. Headless browsers have full webkit support and can thus render pages properly, as a normal browser would, thus bypassing the “javascript challenge”. However, this might simply be not sufficient to overcome bot detection as this is just one of the many challenges employed by Google and others websites. Furthermore, headless browsers are known to be more “power hungry” than, for example, a simple cURL script, thus being difficulty manageable in terms of efficiency

3. The permanent banning from all Google services is the final unavoidable step for anyone not employing any precaution when scraping the Mountain View giant and is only avoidable if steps 2 and 3 of this analysis are carefully managed

As it can be seen, circumventing the *anti scraping policies* is no easy task. Especially so when taking into account the centralized approach used by the tool, which simply performs all of its requests with the same “internet identity” (i.e. a combination of ip, http referer header, and behavioural pattern). The solution is however, simpler that the the ones that were briefly examined in the analysis: moving the scraping logic to the client. The advantages of doing so are pretty easy to identify:

- A client will always have its own private ip
- A client will almost certainly not issue a great amount of requests at the same time

This being said, the client should also employ some sort of preemptive mechanism in order to be *gentler* when performing web scraping. In particular, the client should:

- Employ http referer header string rotation by picking out a new one at each requests
- Have a manageable time out delay between requests
- Being able to send data back to the webservice in a compatible manner with the latter’s specifications

The way this can be accomplished is discussed in the next paragraph *6.2. Future Features*.

6.2. Future features

This work can be regarded as being somewhere in between a *alpha* and a *beta* release and, as such, a number of features and architectural choices could benefit from further developments and conceptual refinements.

6.2.1. Web Scraping logic

In *6.1.2. Limitations* the technical difficulties in providing a centralized web server approach have been discussed, alongside some possible remedies to the issues. The suggestion for improving the situation is that of adopting the more direct approach by taking advantage of the user's system. The requirements for this approach to work would be the following:

1. In order to maintain compatibility with possible future *WEREMATO* clients the scraping process (including the exchange of data with the web service) should happen in a transparent way with respect to the user's machine
2. The scraping procedure should be one that can be easily called via *WEREMATO* clients by using a common API to call upon the procedure
3. The scraping module should be system agnostic, meaning that it should work no matter the system that it is being executed on

These requirements suggests either of the following two options:

- Integrate the solution within each *WEREMATO* client
- Create a service, or daemon process, on the user's machine

Each solution has its implementation costs, advantages as well as disadvantages:

Solution	Implementation	Advantages	Disadvantages
Integrated	Each client can have its own implementations, provided that the information sent to the web server are coherent with the service specifics	The implementation can be tailored to better suit and interact with the client	Each client need to implement it on its own
		Complexity can be handled in the client	No common library (or framework) is available for use with the implementation language of choice

			The complexity of implementing a functional scraping system can discourage developers from contributing to the project
Service (daemon)	The scraping module is developed as a service (or daemon) silently running on the user's machine. It provides common APIs that can be called by any client	A common implementation is available and can be relied upon by the clients, no matter their internal functioning and implementation	The service needs to be system agnostic. This means that a version of the service should be developed to match machines running either Windows, Linux or MacOs operating systems
		A common API interface means that all complexity is hidden within the service	The service needs to be installed in the user's machine. This requires an additional step for the user to take, and possibly discourage him / her from using *WEREMATO* as it requires additional installation procedure
		Usage of a common library can be achieved by relying on the same web scraping module currently employed by *WEREMATO*	

Table: an analysis of the possible alternatives

This analysis suggest that an implementation that relies on a integration of the module with the client would be too costly and less reliable with respect with an internally developed solution that would rely on providing a service (or daemon) to be run independently on the user's machine. This will be the most urgent task concerning the development of *WEREMATO* and should be taken care of before actual release to public use.

6.2.2. Optimizations

Future features of the tool are strictly related to its performances. Currently, the **NLP (Framework)** module is the one in need of most improvements, with respect to its performances and computation costs. The analysis of a single *url* can take up to different seconds at best, while it can result in less acceptable computation times when employing more sophisticated techniques using, for example, the NER (Named Entity Recognition) features provided by the **Stanford NLP Tools**. The proposed optimization is, in this case, an improvement that would benefit almost all modules of the application: a page versioning system to check if the page's contents have changed with respect to a previously encountered version that has been stored in the database and has previously been analysed by the modules. The proposed approach is the following:

- The system would search for the page url's in the database and start either of two procedures:
    - return the previously computed values upon finding the content in the database
    - execute the normal application workflow

How does the tool can detect if any changes occurred in the page's contents? First of all, it is important to point out that any changes to the layout and other elements of the page should not be take into account. This means that the application is only interested in detecting changes to actual content. Of course, storing the actual content of the page inside the database is a non feasible approach that should be avoided, as it consists in duplicating content that is already accessible online and provides no further re-use potential for the application. The best approach is employing a hash function to create a short, concise representation of the page's content to be stored in the database for later comparisons with the most up-to-date page's content retrieved when launching a *lookup* procedure in *WEREMATO*. The hashing function in question should be fast (possibly computing in either linear or logarithmic time) and should be either of two possible hashing functions implementations:

- A cryptographic function, such as **MD5** or **SHA** that would ensure that mapping of even slightly different versions of the content would result in a different hash
- A **locally sensitive hashing function**[44] that would be capable of mapping similar content pages (ones that do not differ in an impactful manner between each other) to the same hash

Since the application needs to be ensured that a page's content is coherent with its previously ran analysis, the best possible solution would be that of employing either MD5 or SHA for creating a hash of the content. The mapping of the content ensures that $hash(x) = hash(y)\ iff\ x = y$ which would allow the application to compute faster, returning results in a less demanding amount of time than what it currently does.

[44] LSH Algorithm and Implementation (E2LSH), http://www.mit.edu/~andoni/LSH/

6.2.3.1. Sentiment analysis algorithm

Similarly to what has been already partially addressed in *6.1.1. Framework Limitations*, the need for a better performing and more accurate sentiment analysis algorithm could be more important in the near future. In order to improve both performances and accuracy, the model's training needs to be refined by incorporating additional elements such as:

Category	Description
Optimization	**similar phrase matching**: whenever a similar statement polarity has been determined, the distance between statements could be computed and the previously computed sentiment returned, if said distance is within an acceptable value
	exact phrase matching: whenever a previously analysed statement is found, its polarity value could be directly returned
Accuracy	a more accurate model (other than the Naive Bayes model) could be considered for providing a more business ready approach and improved accuracy if needed

Table: possibile improvements

Given the current application's settings, however, these possible future optimizations are not considered as either urgent or preventing the application from moving towards a public release stage as requirements for better accuracy are currently not stringent.

6.3. Final Thoughts

The field of application where this work is positioned is a complicated one, with many structural difficulties and caveats that need to be addressed properly in order for everything to blend and work together. This work, as is, provides a first approach towards what can grow into a more robust implementation. Both from a theoretical and technical point of view, this work would benefit for further refinements (which are, of course, planned). However, this first step certainly is a personally satisfying one, as the application is already able to help web reputation professionals in improving and automating their everyday procedures. As such, putting further optimizations aside, I'm personally satisfied with the final outcome of the project.

Bibliography

Bing Liu (2012), *Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers*, Morgan & Claypool Publishers.

Cho J., Roy S. (May 2004), *Impact of Web Search Engines on Page Popularity In Proceedings of the World-Wide Web Conference (WWW)*, Proceedings of the 13th international conference on World Wide Web, pp 20-29.

Das S. R. (2016), *Data Science : theories, models, algorithms, and analytics*.

Devi P., Gupta A., Dixit, A. (2014), *Comparative Study of HITS and PageRank Link based Ranking Algorithms*, International Journal of Advanced Research in Computer and Communication Engineering.

Fielding R. (2000) , *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine.

Jueasky D, Martin J. H. (1999), *Speech and Language Processing*, Prentice Hall Series.

Kirch W. (2008), *Level of measurement*, Encyclopedia of Public Health.

Ozdemir S. (2017), *Data Science. Guida ai principi e alle tecniche base della scienza dei dati*, Apogeo.

Sitography

http://archive.oreilly.com/pub/post/the_worlds_longest_domain_name.html

http://www.wordstream.com/articles/internet-search-engines-history

http://www.makeuseof.com/tag/the-archie-search-engine-the-worlds-first-search

https://tools.ietf.org/search/rfc1436

https://bobbydotseo.wordpress.com/veronica-jughead-1991/

https://www.livinginternet.com/w/wu_sites_yahoo.htm

http://www.internet-guide.co.uk/WorldWideWebWanderer.html

https://searchenginewatch.com/sew/study/2064954/where-are-they-now-search-engines-weve-known-loved

https://digital.com/about/altavista/

http://marketshare.hitslink.com/search-engine-market-share.aspx

http://www.infotoday.com/searcher/may01/liddy.htm

http://www.wikiwand.com/en/Graph_drawing

https://mathsbyagirl.wordpress.com/2016/01/29/the-maths-of-google/

https://www.semanticscholar.org/paper/An-Improved-HITS-Algorithm-Based-on-Page-query-Sim-Liu-Lin/e7e182659614da4f92daca8d8455fd11350f198a

https://moz.com/search-ranking-factors/survey

https://www.theverge.com/2017/6/28/15882272/twitter-riot-predict-faster-than-police-cardiff

https://en.wikipedia.org/wiki/International_Standard_Book_Number

http://www.kdnuggets.com/2015/08/statistics-understanding-levels-measurement.html

http://www.saedsayad.com/k_nearest_neighbors.htm

https://www.kaggle.com/wiki/Metrics

https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

https://vilvas.com/digital-marketing/trump-web-advertising-campaigns

https://en.wikipedia.org/wiki/Web_presence

https://www.rebootonline.com/blog/is-negative-seo-illegal/

https://medium.com/data-science-group-iitr/naive-bayes-unfolded-b2ab036b42b1

http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

http://www.restapitutorial.com/lessons/httpmethods.html

https://www.w3.org/TR/cors/

https://docs.microsoft.com/en-us/dotnet/framework/wcf/samples/jsonp

https://serp-spider.github.io/documentation/#licensing

https://github.com/angeloskath/php-nlp-tools/

https://nickduncan.co.za/tag/naive-boyes/

https://nlp.stanford.edu/software/CRF-NER.shtml

https://github.com/ariya/phantomjs

https://mesosphere.com/wp-content/uploads/2017/06/smack-stack-is-the-new-lamp-stack-comparison.png

http://www.mit.edu/~andoni/LSH/