Tiago Filipe Roque Pereira

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Improving the Reliability of Web Search Results

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: Doutor Tiago Oliveira Machado de Figueiredo Cardoso, Professor Auxiliar, FCT-UNL

Júri

Presidente:	Professor Doutor José António Barata de Oliveira
Arguente:	Professor Doutor João Paulo Branquinho Pimentão
Vogal:	Professor Doutor Tiago Oliveira Machado de Figueiredo Cardoso

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2016

Improving the Reliability of Web Search Results

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "unlthesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2]. [1] https://github.com/joaomlourenco/unlthesis [2] http://www.di.fct.unl.pt

"Em todos os manicómios
há doidos malucos com tantas certezas!
Eu, que não tenho nenhuma certeza,
sou mais certo ou menos certo?"
Fernando Pessoa

Acknowledgements

First, I would like to thank *Faculdade de Ciências e Tecnologias* from *Universidade Nova de Lisboa*. For all the knowledge acquired and for the great working environment throughout this master thesis, this prestigious university was my second home. And I am proud to be a part of this home.

A special thank you to my adviser professor Tiago Cardoso, for the opportunity to work with him in a so fascinating and ambitious project. For all the ideas, all the discussions and all the challenges. For making me think "outside the box" which also led to my personal development. My sincere thank you.

Thank you to my university colleagues and friends, for all the help and all the joy over these years. Bruno Dias, Paulo Rodrigues, Fábio Lourenço and André Pardal, the first ones to be with me. Fábio Nogueira and Ana Rita Salvado for being with me in most of this journey and by the tremendous help provided, your help was priceless. José Reis, for all the hip hop. António Bernardino for all the jokes. Margarida Egídio Reis, Mikail Ribeiro, Patrícia André, Pedro Maricato, Ricardo Pombeiro, Rubina Iquebal, Rui Cardoso, Sérgio Carrola, Tiago Antunes, Tiago Bento, Tiago Lopes, Tiago Robim, Tiago Rodrigues, for teaching me something. Celso Almeida, for all the support, patience and jokes along this last stage. No doubt your support was a strength that helped me finish this project.

Thank you to Jose, Cisolina and Firu Bartolomé Prieto for being the kindness in person. What you have done for me is something I will never forget. Also to Lore Ramos and Margot Mestdagh for showing me how the world is outside Portugal.

Thank you to Tânia Silva for all the books that allowed me to travel without leaving home, Cátia Silva for being the special friend who will always be with me no matter what, Ivo Cardoso for all your wise words which helped me a lot of times, Patrícia Santos for all the conversations in the coffee shop, Mariana Campbell for showing me part of the brazilian culture, Cátia Aguiar for all the laughs, Diana Amorim for being the person who speaks what is on her mind, Tiago João Cardoso for all this years being himself in his own way and of course a special thanks to João André Lopes, for being more than a friend, for being a brother. Someone that showed me family does not mean same blood.

And last but not least, thank you to my parents and sisters, for all the support and patience. Without them, even half would not be possible.

Thank you all. If I am the person I am today, is because everyone taught me something and that is invaluable.

Abstract

Over the last years, it has been possible to observe the exponential growth of the internet. Everyday new websites are created. Everyday new technologies are developed. Everyday new data is added into the web. The search for available online data on the web has become an increasingly common practice to any person because, the regular user wants to know more. For any existing question or doubt, the user wants the answer the fastest way possible. It is in this field where the search engines are an exceptional tool in helping their users.

In order to aid the users reach for what they were seeking for, search engines have become a fantastic tool. Either it is searched for a certain website, some specific information or even for the seek of knowledge, search engines help the user reach his goal. Without their existence, it would be much more difficult and frustrating to find the needed information, which would lead to a tremendous loss of time and resources, and most of the cases, the user would probably not reach the results it was looking for. Thus, the development of web search engines provided a better comfort for the user.

However, despite the fact there is a really effective tool, sometimes it can lead to unintended results. Towards a search, the search engine can lead to a suggestion of a website that does not correspond to the expectation of the user. This is due to the fact that search engines only show part of the content related with each correspondent hyperlink, which for several times, users think the answer for what they are looking for is in some website and when they start analysing it, the intended information is not there. Entering and leaving different websites, can be a big inconvenience, even more if the internet connection is slow (as it can happen outside the big cities or in least developed areas), which makes the user lose more time and patience.

This dissertation intends to explore the possibility and prove the concept that, with the help and junction of different technologies such as parsing, web crawling, web mining and semantic web in a machine, it is possible to improve the reliability from the search engines, in order for the user lose the minimal time or resources possible.

Keywords: Search Engine, Parser, Web Crawler, Web Mining, Semantic Web

Resumo

Ao longo dos últimos anos, tem sido possível assistir ao crescimento exponencial da internet. Todos os dias novos websites são criados, novas tecnologias são desenvolvidas, novos dados são inseridos na web. A busca por informação online que se encontre disponível na web tornou-se numa prática cada vez mais comum por parte do utilizador, isto porque para qualquer dúvida ou questão que exista, o utilizador quer a sua resposta o mais rápido possível. E é aqui que os motores de busca como o *Google*, *Yahoo!* ou *Bing* são ferramentas excepcionais na ajuda aos utilizadores.

A utilização de um motor de busca tornou-se numa prática diária por qualquer pessoa isto porque é das maneiras mais fáceis e rápidas de encontrar a informação que se procura. Quer seja um determinado website, uma informação específica ou ainda pela busca geral de conhecimento, sem a existência de motores de busca, seria muito mais complicado e frustrante efetuar essa tarefa, pois levaria a uma tremenda perca de tempo e recursos e, provavelmente, na maioria das vezes, o utilizador não iria obter os melhores resultados. Desta forma, o desenvolvimento de motores de busca, deram um maior conforto aos utilizadores.

No entanto, apesar de sem dúvida ser uma ferramenta bastante eficaz, por vezes pode conduzir a resultados não pretendidos. Perante uma pesquisa, o motor de busca pode indicar como sugestões de resultados um website que não corresponde àquilo que o utilizador anda realmente à procura. Isto deve-se ao facto de que os motores de busca, quando apresentam resultados, mostram apenas um pouco do conteúdo relacionado com a hiperligação correspondente, o que faz com que por diversas vezes, os utilizadores pensem que a resposta que procuram está nesse website e quando o vão a analisar, a informação pretendida não se encontra aí. O entrar e sair de diferentes websites, pode ser um grande inconveniente, ainda mais se a ligação à internet for lenta (como poderá acontecer fora dos grandes centros urbanos ou em locais menos desenvolvidos), pois faz com que o utilizador perca ainda mais tempo e paciência do que o necessário.

Esta dissertação pretende explorar a possibilidade e provar o conceito de que, com o auxílio e junção de certas tecnologias como *parsing*, *web crawling*, *web mining* e *web semantic*, é possível melhorar o resultado das buscas de um motor de busca, de maneira a melhorar a experiência de um utilizador pela internet, fazendo com que se perca menos tempo ou recursos possíveis.

Palavras-chave: Motor de Busca, *Parser*, *Web Crawler*, *Web Mining*, Web Semântica

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface.

ASCII American Standard Code for Information Interchange.

CLE Command Line Executor.

CSS Cascading Style Sheet.

CSV Comma Separated Values.

DOM Document Object Model.

GUI Graphical User Interface.

HTML HyperText Markup Language.

KB KiloByte.

OWL Web Ontology Language.

PDF Portable Document Format.

RAM Random Access Memory.

RBSE Repository-Based Software Engineering.

RDF Resource Description Framework.

SPARQL Simple Protocol and RDF Query Language.

TSV Tab Separated Values.

URL Uniform Resource Locator.

W3C World Wide Web Consortium.

WWW World Wide Web.

XML Extensible Markup Language.

CHAPTER 1

Introduction

"You can have data without information, but you cannot have information without data." – Daniel Keys Moran, an American computer programmer and science fiction writer

1.1 Context and Motivation

For several years now, the World Wide Web (WWW) has been growing up. And by saying growing up it means the number of websites and associated content has been increasing exponentially. Yesterday began to appear with a "few" websites, today has innumerous websites and tomorrow probably will continue to have many more websites and consequently more content and more information agregated to them.

It is possible to compare the internet with a spider web. By checking a spider web it is possible to verify the web is all connected between itself and there are multiple ways between the strings to reach to a certain point. There are shorter paths and others much longer, but the web is all connected with itself. The same happens if we compare websites. Of course there are some exceptions such as the deep web (websites that a search engine can not find) and the dark web (portion of the deep web where a user can not access through conventional ways) [2], but focusing on the "surface" of the internet, starting from a website there are numerous paths to turn back to the origin. It is like a spider web, some paths are shorter and others longer but it is possible to return to the starting point.

The content of a website can be, among others, defined by text, images, videos or sounds. And that content can be analyzed in a way to discover relevant information that might interest the user. However there is a relationship between the information available all over the internet and the information that can be interesting to a user. Usually, when a user needs to know something, a usual act is to search in a web search engine in way to

have the user's doubt answered. And it is there where the relationship begins. To have a search engine, it is necessary to have a technique which searches information through the internet and also a technique which is capable of filtering interesting information from non interesting information.

1.2 Problem

Taking a look at *Google* (globally the most used search engine), one of its goal it is to work as a search engine. Of course over time there were developed other technologies related to *Google*, however the major known reason is for working as search engine, or in other words "to organize the world's information and make it universally accessible and useful".

There is no doubt that it works with very few (or practically nonexistent) flaws to the regular user, although it is interesting in which way it could be improved.

And this brings up the question, in which way could *Google* (or other usual search engine) be improved? Is there any detail that could be upgraded or fixed? Something that users think could be a gain compared to what currently exists?

Probably one of the most things that happens to a lot (or even to all) people is, after writing what they want to know, the extensive list of results is not so detailed in way to know if the current website has (or not) the relevant information. This could lead to a waste of time, which can be considerable if for the same question search, the user enters and leaves several websites because the result description provided by the search engine was not properly detailed.

This project intends to look into search engine's techniques such as web crawling and parsing but also to add other techniques such as web mining and semantic web in order to create a software composed by those techniques to prove the concept about if this junction could result or not in an improve of the reliability of web search results and improve the user's experience on the web as well.

1.3 Dissertation Structure

This dissertation is organized as follows:

- Chapter 1: Introduction: This is where an introduction to the theme of the dissertation is made. The context is presented and the main problem is identified.
- Chapter 2: State of the Art: Describes what has been done up to the present related with the technology and the theoretical context of how it works.
- Chapter 3: Conceptual Proposal: An explanation of the proposed architecture, which are the used techniques and ideas to solve the problem presented at the Introduction.

- Chapter 4: Results Validation: It is detailed two application scenarios and steps of the implementation of the proposed architecture and gets into the application results to validate the architecture.
- Chapter 5: Conclusion and Further Work: Summarizes the study and the conclusions about its achievements. A suggestion is made by describing the next steps to do with the proposed architecture.

CHAPTER 2

State of the Art

"The next time you see a spider web, please, pause and look a little closer. You'll be seeing one of the most high-performance materials known to man" – Cheryl Hayashi

The present chapter has the aim to do the state of the art that is concerned to the development of the techniques of searching through the Internet. In Section 2.1 an introduction is made related to the Internet and how it is composed. The following sections describe the main techniques in study (parser, web crawler, web mining and semantic web). It is made the state of the art from those techniques and the analysis of three different software related with web crawling and web mining, correspondingly.

2.1 World Wide Web

In October of 2014 it was announced by *NetCraft* an historic milestone. It was the first time that 1 billion websites were reached after several years since the first ever made website, back in 1991. This event was as well mentioned by Tim Berners-Lee (the inventor of WWW).

Since 2000 to 2014, the number of existing websites has been growing up, but starting from 2010, it was registered an expressive increase of the growth, as can be seen on Figure 2.1.

Analyzing the Figure 2.1, it is also possible to find that, despite the websites growth, from 2009 to 2010 and from 2012 and 2013, the total number of websites suffer a decrease. This happens because of fluctuations in the count of inactive websites. That is why in 2014 it is represented less than a billion websites. That milestone was reached, however until the end of that year, the number of websites declined (the last results lasts of nearly

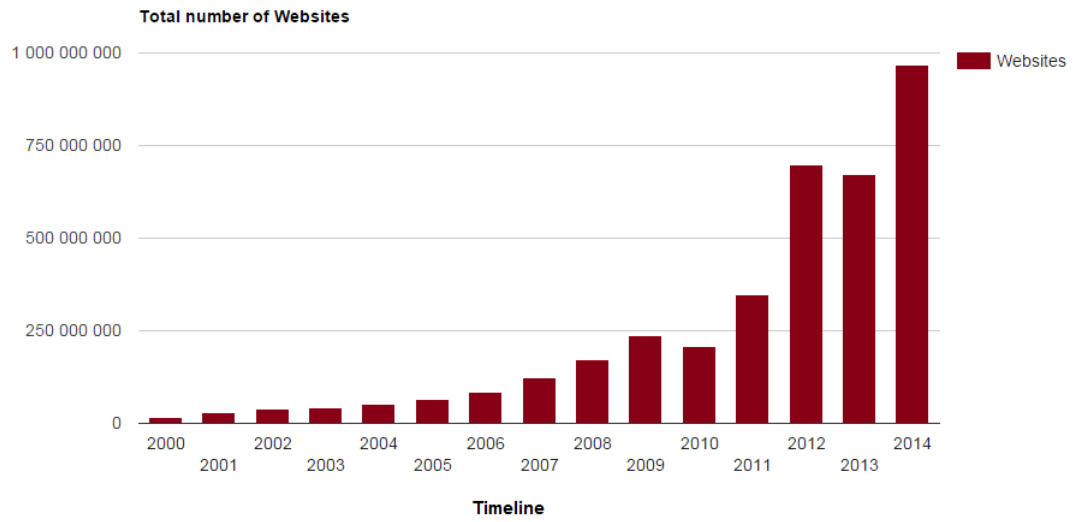

Figure 2.1: Website's growth from 2000 to 2014. Adapted from [32]

969 million websites). [32]

What if it was possible to have a visual representation of the websites, like in a map? Likewise it exists a world map, showing the different continents and oceans, there also exists a representation of the different websites over the Internet. Developed by Ruslan Enikeev, the Internet Map represents the dimension of a website on the Internet. There exists several representation of the internet websites, however, this one was developed for the user to have a better perception of the Internet universe. Figure 2.2 represents the last update of the global network, at the end of 2011. The representation is quite simple. The algorithm was developed in order to any website be represented by a circle, and for more amount of traffic all over a certain website, the bigger the website's circle will be.

The circles tend to be closer to each other if users tend to go from a website to another. For a general behavior from the users visiting a website to another, the closer the websites will be represented in The Internet map. [9]

The Internet map is composed by over 350 000 websites from 196 countries. In order to put the map all clear, a color was defined to represent the websites of each country. Figure 2.3 shows that for example United States is defined by blue, India by green and China by yellow. It is as well possible to see that United States have a larger number of websites represented in this map.

Like it is possible to understand, it would not be totally wrong saying that the Internet is like a world itself. A very large and complex world with a lot of data to explore. Everyday more data is introduced into the Internet and with that, it is interesting to have tools to help people to extract portions of that data which can be relevant to them.

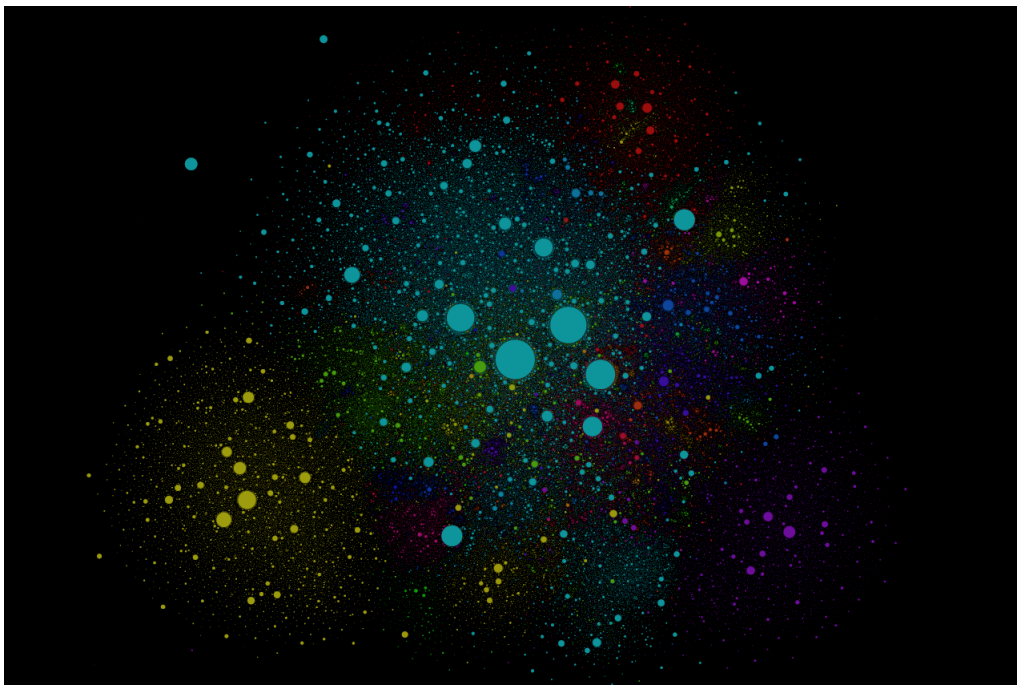

Figure 2.2: The Internet Map. Adapted from [9]

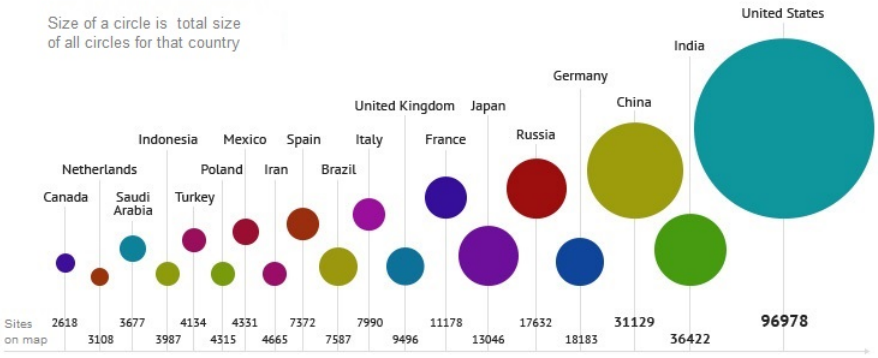

Figure 2.3: Number of websites represented on The Internet Map. Adapted from [9]

2.2 Parser

To analyze the information of a website, the first step to achieve is to get a parser. Basically, like the name suggests, the main function of a parser is to parse something, to analyze information, it separates the content into certain parts. If we take the next phrase for example:

"Tiago reads a book."

It is possible to parse the previous sentence into the subject (Tiago), predicate (reads) and object (a book), this is an example of parsing the English language, and in computing

language the idea is nearly the same. It is possible to analyze files and from those files parse and identify relevant information. In this case, a parser is a program, a piece of code or an Application Programming Interface (API) which can be adjusted to do a specific action. [5]

After a brief search about parsing tools (where the main condition was a tool programed in HyperText Markup Language (HTML)/PHP), three parsers were selected to proceed to their analysis:

- htmLawed;
- HTML Purifier;
- Simple HTML DOM.

These three parsing tools are written in PHP with the functionality to process HTML markup text. The initial difference detected between them was the fact that both Simple HTML DOM and htmLawed do not have more than 50KiloByte (KB) and are composed by a single file, where HTML Purifier is around 15 times bigger, which according to [1] leads to a slower data processing and also to a bigger consumption of Random Access Memory (RAM).

On the following Table, it can be found a brief comparison between them with factors that were thought to be important. Those factors were chosen according to the required needs.

	Friendly User	**Easy to Learn**	**AvailableExamples/ Demos**	**Size**
htmLawed	+/-	-	+/-	+
HTML Purifier	+/-	-	+/-	-
Simple HTML DOM	+	+	+	+

Table 2.1: Comparison of Parsing Tools

In this table it is possible to see some differences between these parsing tools, and with these results, easily it is seen that Simple HTML DOM is the most friendly user and easy to learn.

However, several tools have a better performance depending on the scenario where they are applied. That is why HTML Purifier and htmLawed are probably not the best tools for web crawling. It would probabily be better in a scenario where it was necessary to have cleaned HTML, where it is requested a clean and pretty output.

2.3 Web Crawler

The main function of a web crawler is to follow the most hyperlinks available on the web. Starting from one or more seed URL (defined as the first website that will be analyzed), it will look for the available hyperlinks on the current webpage and it will continue to search in the following webpages. A good seed URL is defined as a URL with the most available hyperlinks to different websites.

Web crawlers have an important role on web search engines because they are used to extract the content from a webpage indexed by the search engine. For a web crawler to be able to go as deep as possible into the web, it is necessary to start with a good seed URL. That is because the hyperlinks of a website may be directed to a different webpage or to another page of the same website and to have a good web crawler it is important to start with a good seed to get as many different websites as possible. Therefore a web crawler can have innumerous tasks such as to serve as web data mining, as a comparator between shopping engines and much more. [23]

2.3.1 Web Crawler/Search Engine: Timeline

Web crawlers and search engines are closely related. As said before, one of the main functions of a web crawler is to follow and to visit the most webpages available as possible. On the other hand, it is not possible to build a search engine without a web crawler. That is why these two technologies are so commonly confused.

It is possible to say that web crawlers are practically as old as the Web. In 1993, the *World Wide Web Wanderer*, mainly described simply as *Wanderer*, was written in Perl and it was developed by Matthew Gray. One of its particularities was the fact that it ran only in one machine and the main function was to measure the growth of the Internet. Later it was upgraded to work by capturing URLs and so it was declared as being the first search engine available on the web. Until the end of the year, a few more crawler-based internet search engines were released such as *Jump-Station*, the *World Wide Web Worm* and the *Repository-Based Software Engineering (RBSE) spider*. *WebCrawler* and *MOMspider* were launched on the following year, which were made part from the first generation of web crawlers. However, as it was the first generation, there were identified some problems in their design, namely, at the level of scalability.

In 1994 *Yahoo!* was founded by David Filo and Jerry Yang. When it was launched, the websites were listed by human editors like a collection of favorable websites. Over the time *Yahoo!* became a searchable directory because of the increasing data.

Altavista, founded in 1995, was initially developed to crawl, store and rapidly index every word of all webpages and it was the first search engine allowing natural language queries. In August of 1995 it was composed with about 10 millions pages. It became one of the most popular search engines at the time.

Ask Jeeves (actually known as *Ask*), was developed in 1996 and launched in 1997, also

with a natural language search engine. It was the characteristic that human editors tried to match search queries, which means that they listed the most important and relevant websites and the paid ones (similarly as *Yahoo!* at the beginning) and the websites were organized by specific popularity.

In 1998, *Google* was introduced by Brin and Page as an attempt to solve the problem of scalability. One particularity was the fact that *Google* was designed to reduce disk access time. Another factor developed by them it was *PageRank* algorithm, which was used to calculate the probability of a user to visit a certain page. The first implementation of *Google* provided 100 pages per second at its peak with an average size of 6 KB per page. Nowadays it is the most used search engine in the world and offers a wide range of different services. [24]

In 1999, Allan Heydon and Marc Najork presented the *Mercator web crawler*, which was introduced as a web crawlers' guide. Written in Java, *Mercator* was highly scalable and easily extensible. Highly scalable because it was designed to scale up all the internet and to fetch millions of web documents. Extensible because it was developed in order to, in the future, new features could be added by other developers.[13]

Polybot, developed by Shkapenyuk and Suel, is another example of a "blueprint design" webcrawler. As well as a scalable web crawler, with the capacity of downloading up to hundred web pages per second, it has the characteristic of avoiding crawling the same URL several times, which led to an improve to its performance. Over 18 days using four machines, *Polybot* was able to download 120 million pages. [24, 30]

According to Seyed M. Mirtaheri et all, the three most influential cases for the use of crawling are the fact that a search engine needs the support of a web crawler to find existing data on the web, for automated testing and model checking of the web application and for the discovery of vulnerabilities. As it is known, there are innumerous web applications using delicate data and services. Thus, it was developed security scanners capable to recognizing potential issues in an automated and effective way, where it is essential to have a web crawler to discover the state of each scanned application. [20]

2.3.2 Web Crawling Software

This subsection describes some open-source web crawling tools, their features and utilities.

2.3.2.1 Sphider PHP search engine

Sphider is an open-source web crawler and search engine written in *PHP* and *MySQL*. It works in a similar way like a webcrawler (in regards to the fact that it can follow the existing links of a website) and like a search engine since the user can enter a keyword (or a set of words) to focus more in a concept and to discover information about it.

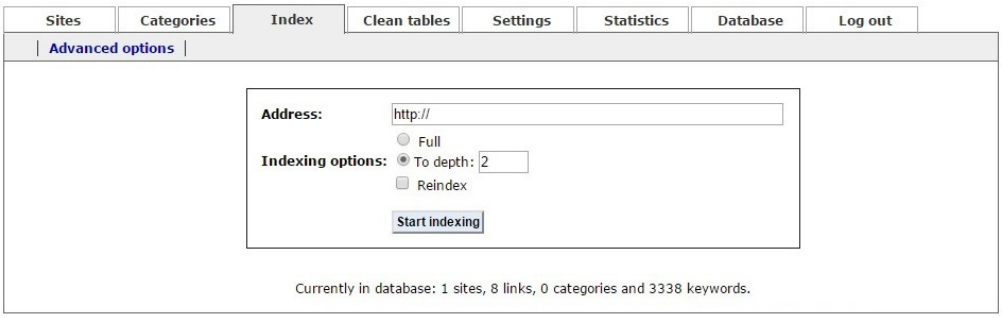

Figure 2.4: Sphider's initial interface.

This software has an indexer which builds an index of the search words that were found in the covered web pages and has the particularity of being implemented with the Porter stemming algorithm (which is based in reducing a certain word into his root word. Analyzing the words "spiders", "spiderlings" and "spiderman", this algorithm returns these words into their root word, which is "spider"). [17, 27]

Figure 2.5: Sphider results.

Observing Figure 2.5, after the web crawler do its task of visiting and indexing the hyperlinks from where it went, it is possible to search for something. As logical, the

deeper the web crawler searches, more results can be available to consult.

Despite the fact that the web crawler works correctly, it is necessary to update a few definitions because the main scripts are deprecated. Those scripts are not updated with the new versions of *MySQL* which leads to have some initial errors. However, there are some updated versions of this software (such as *Sphider-plus* and *Sphider Pro*) where the most vulnerable parts of *Sphider* software are more robust but it has the disadvantage of not being open-source.

2.3.2.2 PHPCrawl

PHPCrawl is a highly configurable machine of web crawling written in *PHP*. It offers to the developer the possibility to specify its behaviour according to his needs. In an easy mode, the user starts the processing introducing a website. At the end, it will receive all hyperlinks found by this web crawler. However the developer can extract data in different ways, such as getting the images that *PHPCrawl* finds or else a different type of content.

This tool is composed by two parts. One of them works in a simple way, it does not need a lot of initial settings and the project script is not very elaborated. In response, as can be seen on Figure 2.6, it leads to an ordinary and simple representation of the obtained results.

```
Page requested: http://www.dmoz.org/Health/ (200)
Referer-page: http://www.dmoz.org
Content received: 41617 bytes

Page requested: http://www.dmoz.org/Health/Fitness/ (200)
Referer-page: http://www.dmoz.org
Content received: 22878 bytes

Page requested: http://www.dmoz.org/Health/Medicine/ (200)
Referer-page: http://www.dmoz.org
Content received: 25571 bytes

Page requested: http://www.dmoz.org/Health/Alternative/ (200)
Referer-page: http://www.dmoz.org
Content received: 31720 bytes

Page requested: http://www.dmoz.org/News/ (200)
Referer-page: http://www.dmoz.org
Content received: 75206 bytes

Summary:
Links followed: 36
Documents received: 35
Bytes received: 1045674 bytes
Process runtime: 26.038589954376 sec
```

Figure 2.6: Phpcrawl's ordinary results.

The other one is more complex and elaborated, also with more settings and with more detailed results, as can be seen in Figure 2.8.

PHPCrawl is easier and more intuitive to use compared to *Sphider*. This is due the fact that this software only deals with web crawling unlike *Sphider* which works as a web crawler and as a search engine. It is as well simpler to use because its only focus is to take the links of a webpage (or other different content). However, despite the fact that its main goal is to get the hyperlinks of a webpage, it is possible to add a set of options to show to the user some more features and options. [14, 15]

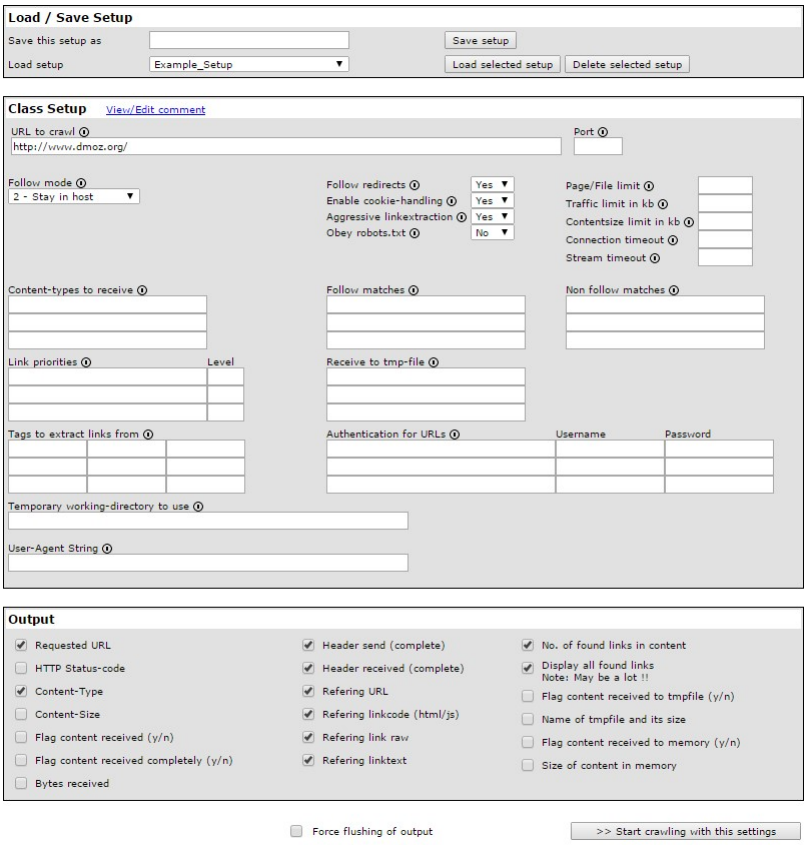

Figure 2.7: Phpcrawl initial interface.

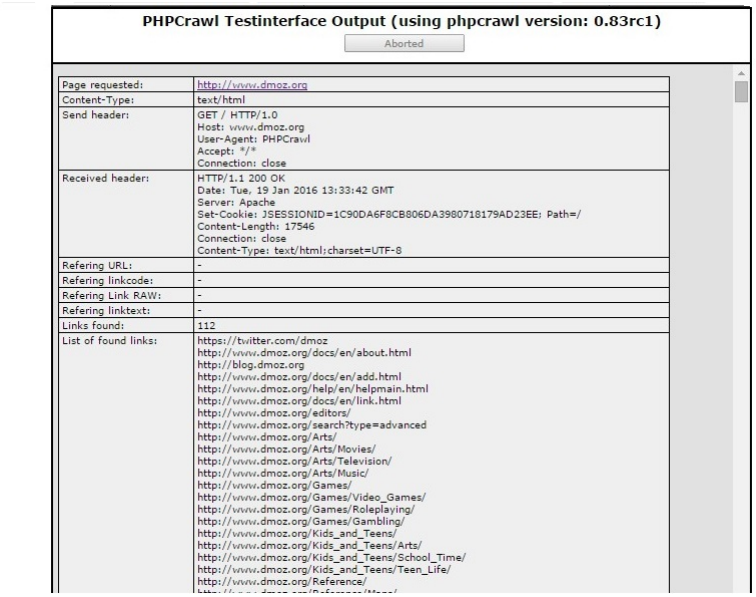

Figure 2.8: Phpcrawl final results.

2.3.2.3 OpenWebSpider

OpenWebSpider is a multi-threaded web crawler and search engine written in *node.js*. Like other web crawlers, *OpenWebSpider* has as its main purpose to visit websites, however it also creates an index of entries to work like a search engine. With this option, after crawling a website, it is possible to search words or expressions that exist in the visited websites. This is a good feature because if the user is searching for anything in specific, this eases the work since the user does not need to search link by link for the information, which would be an expensive and laborious job. When testing this software, *MySQL* was used for its database. The version tested was v0.2.3. [6] [34]

Figure 2.9 shows the main menu of this software.

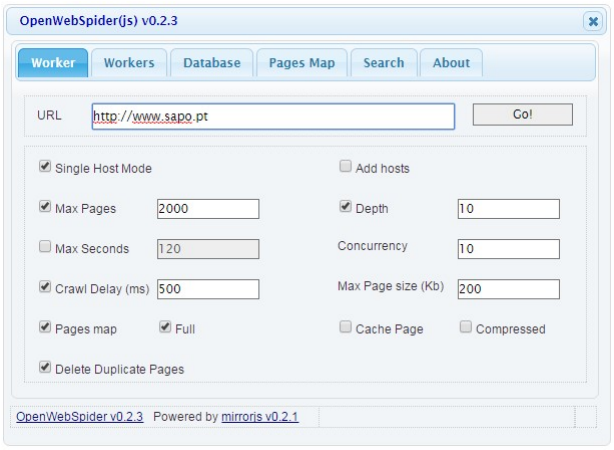

Figure 2.9: OpenWebSpider initial interface.

Working with *OpenWebSpider* is considerably simple since it has a friendly interface and the provided documentation is clear and easy to understand, regarding the required initial configuration and the essential steps needed to start to work with this web crawler. The main menu has some options to the user to configure, such as the maximum number of pages or the depth that the web crawler will go by. Another appealing factor this software has, is the fact that it creates a map of the pages. Basically it shows where the anchor text links to and where the anchor text is linked by, as it can be seen in Figure 2.10. It also has the appealing feature of having an option to avoid duplicate pages, which can be really useful since it could create unnecessary links.

Since it works as well as search engine, *OpenWebSpider* has a field where it is possible to search for a certain topic, returning the relevant links related to the topic. However a detail that was noticed was the fact that it does not parse all the hyperlinks of a website. It was tested in several websites and in some of them no result was found by this software, which shows that this software can be relatively limited of what it parses. [25]

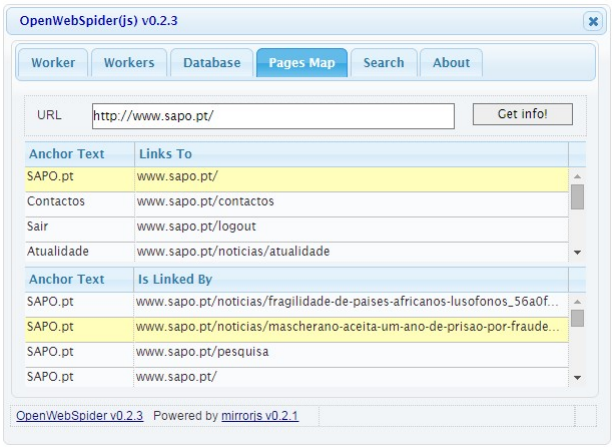

Figure 2.10: OpenWebSpider results.

2.4 Web Mining

According to U.M. Fayyad et all [10], web mining can be defined as "the nontrivial process of identifying valid, previously unknown, and potentially useful patterns". As it is known, there is a lot of available data on the web that can be classified as structured, semi-structured or unstructured where it is possible to apply web mining techniques, which is a huge help to translate data from human-understandable to machine-understandable semantics. Web mining can be divided into three main areas: web content mining, web structure mining and web usage mining. Those three areas are related to each other although they differ in some points as it is possible to see in Figure 2.11:

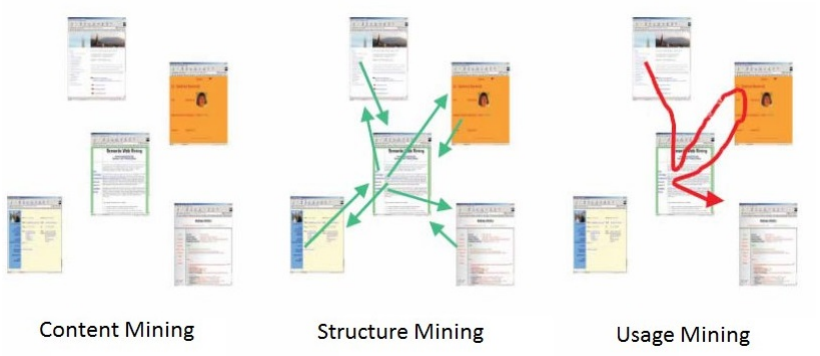

Figure 2.11: Representation of different web mining techniques. Adapted from [19]

This Figure will be described in more detail later but in a brief description it is showing content mining, which is mining the data as an individual page, structure mining, which acts on the hyperlink structure of a webpage, and usage mining, which mines the requests of the users in a website to see their behavior, usually gathering the information in a web server log. [19]

2.4.1 Web Content Mining

The purpose of web content mining is to examine methodically and in detail the content of web resources such as individual web pages. In order to establish facts and reach new conclusions, information retrieval is one of the areas that supplies a range of popular and effective, mainly statistical methods for web content mining.

To complement standard text mining techniques, web content mining is in a good position to use the semi-structured nature of webpage text. Taking a look at HTML tags and Extensible Markup Language (XML) markup carry information, it is noticeable that it interests not only regarding the layout but also at the logical structure.

Web content mining's main goal is based on the discovery of patterns in large documents and in documents collections which are commonly changing. There are several options of use for this technique such as collection a set of news from online newspaper or in a little bit more complex usage and analysis, used to detect trends or declines of topics who might have some interest. [33]

2.4.2 Web Structure Mining

When referring to web structure mining, it means the mining is concentrated on sets of pages, starting from a single website to the web as a whole, which commonly acts on the hyperlink structure of those Web pages. There is some information available in the structure of the hypertext, which can be exploited by this type of mining. That's why, according to Gerd Stummea et all, "an important application area is the identification of the relative relevance of different pages that appear equally pertinent when analyzed with respect to their content in isolation".

For instance, hyperlink-induced topic search analyzes hyperlink topology by finding authoritative information sources for a wide search topic. Authority pages (webpage with high relevance to appear on search engines [31]) find this information, which is determined in connection to hubs (pages that redirect as many as possible to associated authorities).

Analyzing the *PageRank* algorithm, it shows that the more hyperlinks a page has from other pages the more relevant the page becomes (especially if these other pages are relevant as well). This algorithm is used by the search engine *Google*.

Some researchers include web structure mining and web content mining just as web content mining because they usually perform together which leads to the algorithm discovering both the content and the structure of hypertext. [33]

2.4.3 Web Usage Mining

Web usage mining pays particular attention to records of the requests from a user on a website, which generally are collected from a web server log. The behavior that a user has while is navigating through a website can reveal additional structure.

Certain connections between webpages may be introduced by usage mining where at first no specific structure was designed. The meaning of the previous sentence is, taking as an example, in an online list of products, there is no structure between them, no relationships between the products. However, if the behavior of the users in a site is analyzed, probably it will be possible to notice that someone who was interested in a certain product X, was as well interested in product Y. This "interest" can be defined, for example, by placing a particular product in the shopping cart or by making a request which matches the description of that product. Those correspondences among the interest of the user in several products can be used for personalization (for instance, by mentioning to the user about the existence of product X at the moment that he was viewing product Y). [33]

2.4.4 Web Mining Software

This subsection describes web mining software based on web content mining. There were chosen three different software (*Mozenda*, *Automation Anywhere* and *DeiXTo*) which have the same goal, the knowledge discovery and extraction from a website. It is made a description of each software and finally a comparison about the similarities and differences between those software.

2.4.4.1 Mozenda

Mozenda is a tool that can transform information found in the Internet into structured data. The main goal of *Mozenda* is to obtain the information in a usable format and it works in three main steps. The first one is to identify and collect the required information, where the user selects the intendent content (which can be text, files, images or Portable Document Format (PDF)). Then, second step is responsible to structure the selected information. This information can be previously exported into common formats such as XML, Comma Separated Values (CSV) or Tab Separated Values (TSV) and then, the final step, analyzes what it was obtained and see if it meets the intended requirements. [22]

Mozenda simulates the human browsing experience reproducing several commands to extract information from a particular webpage. Thanks to this, it is possible to transform unstructured data into systematized and structured information, which leads to a better performance when the user analyzes the data. This software can have several purposes, depending on what the user has in mind, however the most common uses are for price comparison, data monitoring, list building, among others. It can be as well applied in implementations of large amount of data as a way to enhance their optimization, because it usually has to handle a lot of data at once, but this way, all that data can be processed and examined on scheduled intervals, otherwise this would have to be done "by hand" which would take a lot of time and resources. [21]

It is a user friendly software, the user does not need to have any programming skills, only needs to know how to use a browser. Then, selecting the data products that are

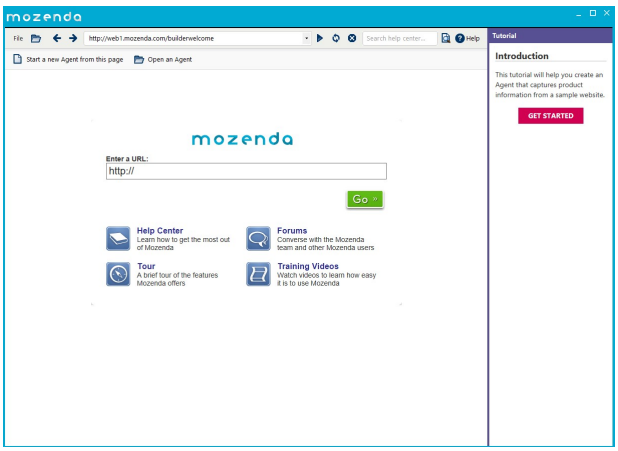

Figure 2.12: Mozenda's interface.

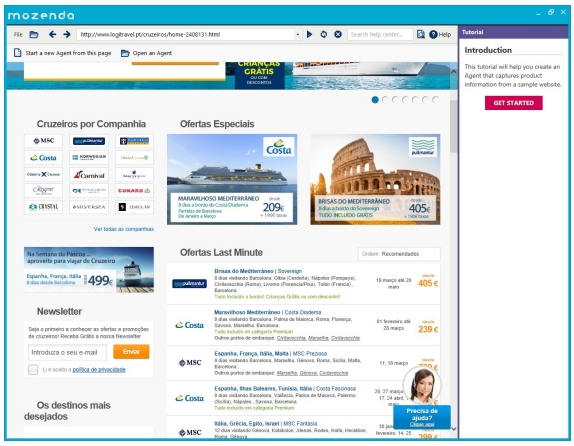

Figure 2.13: Mozenda's results.

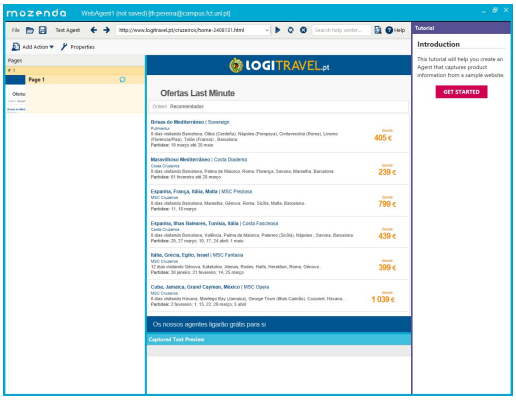

Figure 2.14: Mozenda's results.

Testing Results

	test	name	BriefDescription	Date	price
1	Ver to...				
2		Brisas do Mediterrâneo	8 dias visitando Barc...	2016-03-19 00...	405 €
3		Maravilhoso Mediterr...	8 dias visitando Barc...	2016-03-28 00...	239 €
4		Espanha, França, Itália,...	8 dias visitando Barc...	2016-03-18 00...	799 €
5		Espanha, Ilhas Baleare...	8 dias visitando Barc...	2016-03-27 00...	439 €
6		Itália, Grécia, Egito, Isr...	12 dias visitando Gén...	2016-01-30 00...	399 €
7		Cuba, Jamaica, Grand ...	8 dias visitando Hava...	2016-03-29 00...	1 039 €
8		Antilhas e Sul das Cara...	8 dias visitando Coló...	2016-01-29 00...	649 €
9		Caraíbas Tropical	8 dias visitando Sant...	2016-03-19 00...	899 €
10		França, Itália, Malta, Es...	8 dias visitando Mars...	Partidas: 20 fe...	479 €
11		Jamaica, Grand Cayma...	8 dias visitando Mia...	Partidas: 30 a...	359 €
12		Bahamas, St. Maarten, ...	8 dias visitando Mia...	Partidas: 23 a...	309 €
13		Caraíbas e Antilhas	8 dias visitando Mia...	2016-03-05 00...	1 669 €
14		Mediterrâneo Ocidental	12 dias visitando Bar...	Partidas: 3 fev...	655 €
15		Mediterrâneo Ocidental	13 dias visitando Bar...	2016-03-16 00...	598 €
16		Itália, França	4 dias visitando Barc...	Partidas: 2, 10...	155 €
17		Ilhas Baleares, França, ...	6 dias visitando Barc...	Partidas: 5 maio	439 €
18		Ilhas Baleares, França, ...	5 dias visitando Barc...	Partidas: 28 a...	344 €
19		O vale do Douro	6 dias visitando Opo...	2016-03-24 00...	669 €
20		Ilhas Baleares, Itália	8 dias visitando Valê...	2016-03-07 00...	309 €
21		Portugal, Espanha, Fra...	7 dias visitando Lisb...	Partidas: 30 a...	259 €
22		Ilhas Caimão, México ...	8 dias visitando Mia...	2016-03-20 00...	780 €

Figure 2.15: Mozenda's final output.

relevant to analyze, it starts the agent and begins to harvest all related data. Finally the knowledge is transferred into a spreadsheet to be easier for the user to analyze the results.

2.4.4.2 Automation Anywhere

Automation Anywhere is a powerfull software to extract data at real time. Each business has its own way of showing available information online. From finance to retail industry, each one has its own relevant content and information. In the case of finance, the main content could be to extract stock information and in retail it is possible to find product catalog with different products and prices which the user can compare. Like *Mozenda*, *Automation Anywhere* is a tool that can extract any type of data independent of the industry or formats.

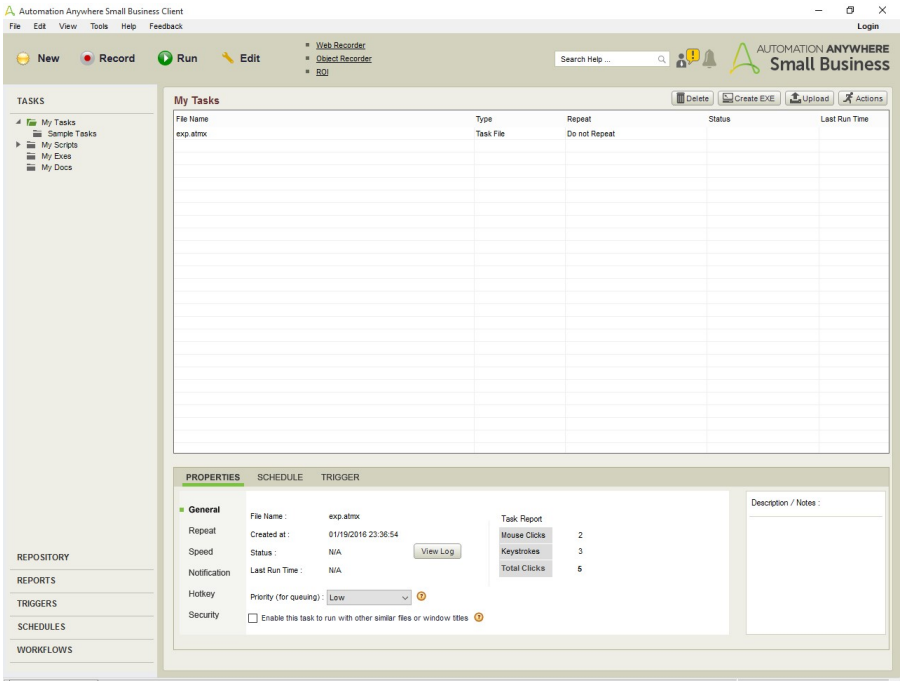

Figure 2.16: Automation Anywhere's interface.

This software has developed a smart automation technology to help the user to extract what is relevant to him. When it comes to web mining, there is the option to extract data based on a pattern in the web. This software recognizes the pattern of relevant data running through several webpages and captures what the client intends.

It is as well a user friendly software since there are a lot of demos and support teaching how to use it and does not require any type of programming. When it comes to extract data, it has the option to extract regular data or pattern based data, whichever will have a better performance, according to the type of data the user intends to analyze and it has the capability to extract data in a quick way (no more than 2/3 minutes).

Another feature that *Automation Anywhere* has is that it shows an "Action List" where it previews the steps of what the user wants to collect. At the end, similarly to *Mozenda*, it generates a file containing the collected data.

2.4.4.3 DEiXTo

DEiXTo is a software based on the World Wide Web Consortium (W3C) Document Object Model (DOM). This tool allows the possibility to extract content based on accurate rules described by the user. It is composed by three main components: *Graphical User Interface (GUI) DEiXTo*, *DEiXToBot* and *DEiXTo Command Line Executor (CLE)*. *GUI DEiXTo* is used to create a user friendly interface that is used to handle the extraction rules, the *DEiXToBot* is an open source module written in Perl and its main function is to extract interesting data which is based on patterns built using *GUI DEiXTo*. Finally *DEiXTo CLE* is a cross-platform application that can apply the extraction rules over several target pages and produces a structured output. [18]

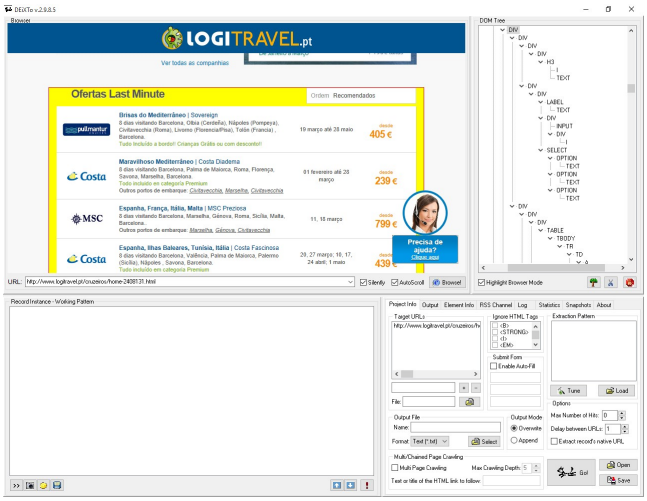

Figure 2.17: *DeiXTo* analyzing http://www.logitravel.pt/cruzeiros.

DEiXTo is an interesting tool due to the fact that it is so interactive and it allows to view in real time where the user is clicking. Probably is not the best tool for people who do not have any programming skill, but for those who have some basic knowledge in HTML, it is an interesting software because it is possible to see the components belonging of a website and see where the user is clicking and what it corresponds to.

However there is a difference between this software and the previous two. The application used is more focused on web scraping, however it can be applied to web mining techniques as well.

The version used it was v.2.9.8.5.

2.4.4.4 Software Comparison

Table 2.2 shows a brief comparison between the previous software.

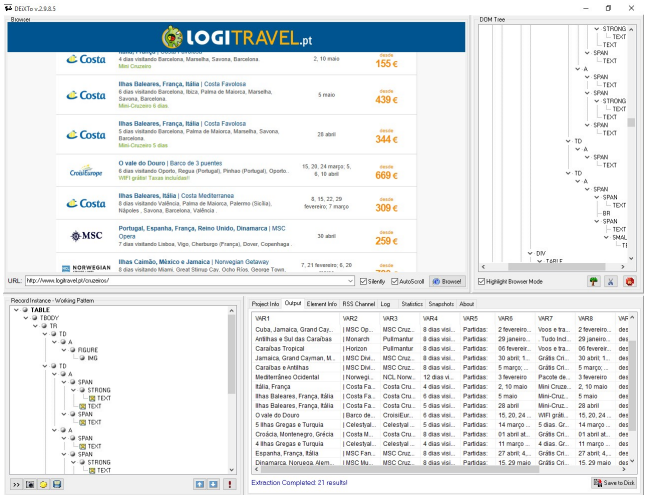

Figure 2.18: *DeiXTo* results.

It is easy to notice that between these three software, *Mozenda* is probably the most efficient. It is the most robust and has the best results regarding to what the user was searching for. *Mozenda* proved to be great in terms of support. The software's support pay a lot of attention to their clients by sending emails asking if there is any questions and if any help is required. However *DEiXTo* proved that it can be a good tool to work with for those who have some programming skills, however probably it is not suitable for those that do not understand it. Although it can be appealing to use it does not have a lot of demos or documentation like the other tools, therefore it can be a challenge to define the rule to extract information.

However it should be noted that, as previously referred, *DeiXTo* is not necessarily a complete software but more a tool where web mining can be applied.

Table 2.2: Comparision between *Mozenda*, *Automation Anywhere* and *DEiXTo*

	Friendly User	Easy to Learn	Extract Structured Data	Extract Unstructured Data	Defining Rules	Performance
Mozenda	+	+	+	+	+	+
Automation Anywhere	+/-	+/-	+	+	+	+/-
DEiXTo	-	+/-	+	+	+/-	+

2.5 Semantic Web

Semantic web's [28, 29] main goal to give a new meaning to the way scientific data is collected, stored and deposited. It means that with this technique, it is possible to give a new meaning to the collected data (more related to the meaning than to the structure

of information). This way, the machine can try to understand what, for example the real meaning of an english sentence in order to try to respond the several requests of a user.

Semantic web is supported by three pillars, Resource Description Framework (RDF) which is responsible for the data modeling language, it is an effective method to represent any type of existent data defined on the web, the Simple Protocol and RDF Query Language (SPARQL) a protocol used to manipulate data between different systems and Web Ontology Language (OWL) which is a language that is better interpreted by the machine. The concepts are defined in a way that can be used in different possible situations combined with other concepts and applications.

CHAPTER 3

Conceptual Proposal

"There's a saying among prospectors: 'Go out looking for one thing, and that's all you'll ever find.'"– Robert J. Flaherty

This chapter describes the processes required to provide the support to understand how the logical architecture worked. In order to specify how the proposal intends to work, the transfer and sharing of information from one block to another will be explained.

Revisiting the initial research question, this chapter will describe how the proposed tool works and if it is possible to have interesting and better results when searching for something. Two different areas of Use Cases will be described and where they can work to provide good results to the user combining web crawling, web mining and web semantic all together in one software.

3.1 Revisiting the Problem

Nowadays, when a person has some kind of question or doubt, the most common act is a quick search on a web based search engine. No doubt *Google* is the most visited and used search engine globally. According to [8], *Google* has a unique estimate of around 1.1 billion visitors per month, where *Bing* is the second most used search engine with around 350 million users, followed by *Yahoo!* with around 300 million monthly users.

There could be a lot of reasons why people prefer *Google* instead of other search engines, like being able to offer more services while being the most efficient and user friendly in comparison to the other tools. However, the main reason why these search engines are used is because people want to know or search for something in a fast and accurate way.

Having in mind these three main search engines (*Google*, *Bing* and *Yahoo!*), despite the fact that nowadays there are many more services and applications, their main functionality continues to be working as a search engine. The final result, apart from using *Google*, *Bing* or *Yahoo!*, is the same: showing different relevant URLs that could be the answer for the user's search. Of course, for the same typed question, the three search engines' results can change. This happens because each search engine has its own search algorithm. However the final result is the same, it is shown a hyperlink with 2 or 3 lines of content of that same hyperlink where in those lines, it is shown a little description with information related to the search. In some cases, only by reading those lines, the user finds what he was looking for, in other cases, he follows the hyperlink and searches there for the information he was looking for. There is also a misleading case in which the user thinks that a certain hyperlink will take him to the information that he is expecting, but at the end he won't find anything useful making him feel frustrated for losing some time.

And there is the problem. In which way could it be possible to improve the reliability of web search results? What could be done to make users lose less time in getting what they are looking for? How to improve the user's experience on the internet?

3.1.1 Proposed Solution

The proposed solution for this problem is based on a set of certain technologies which are:

- Parser;
- Web Crawler;
- Web Mining;
- Semantic Web;
- URL Classification.

The main characteristic of the proposed tool is the combination of these techniques all in one. The use of a parser and web crawler is necessary to build a simply search engine and the use of web mining, semantic web and interesting classification is used to obtain the intelligence of this architecture.

3.2 Use Cases

This section describes a few use cases where the proposed tool can be applied. This is a generic tool in the way that it is disconnected of any specific business area, which means that is suitable to any type of web search. The next subsections will to detail some use cases where the tool can be applied in a description of possible environments.

3.2.1 Tourism

It is a fact that most people enjoy traveling, to visit new places and have new experiences. Either in their own country or in a different one, people generally appreciate new experiences to escape from their daily basis routine. There are a lot and different reasons why a person would like to travel. Either to learn a new language or see new things, people can have several reasons to do it. Depending on someone's preference, when a person is trying to decide if he wants to travel, he looks for information available on the internet.

According to the typed expression, the proposed tool will travel through several URL, starting from a seed URL, and blocks with pieces of relevant information will be shown as result of the algorithm. This pieces of relevant information could be the answer the user was looking for, and if so, does not need to lose some time to enter and exit differents websites.

3.2.2 Retail

Whether the user is searching for a laptop or a mobile phone, nowadays it is usual for a person to search for a determined product and look for prices to see the relation price/quality or even to compare characteristics between products. Inserting an initial search query, the proposed tool will search for relevant content to return to the user.

3.3 Software Requirements

According to Wilson Filho [26], what makes the value of a product is its characteristics. In the case of a software, usually the characteristics are divided into functional characteristics (which represent the program behavior) and non functional characteristics (which quantify certain aspects of the program behavior such as if it is user friendly or the mean time between errors).

The requirements are characteristics which define the acceptance standards of a product. For the proposed architecture, the following points were defined as functional requirements:

- The user's interface should be accessed by mouse and keyboard to be possible to use all the features available;
- In order to improve the software's performance, there is the possibility to change several input fields;
- The user should have the capability to cancel the software processing earlier than forecast if any problem shows up or when enough results have been found;
- The software should work in a generic way, which means that should work and have interesting results no matter the input fields introduced by the user;

- The software should show the results in a simple way, which means that it should be easily noticeable for any kind of user even if does not have any programming skills. Additional results like statistics should have the same characteristic.

As for non functional requirements:

- The software should work in any computer, no matter it is a new one or an older one;
- The software should be user friendly and work in an intuitive way, so it does not take too much time to any kind of user;
- If any failure happens, the software should be capable of recovering the processed data until the actual moment, saving it;
- If for any reason the system goes down, the user should have a solution to resuming the data processing at any time;
- The programming structure should be schematized and have a description of the most relevant parts of the architecture, in order to ease the work development, if for any reason it is justified.

3.3.1 Default Inputs

For the purpose of putting the software to work, it will be necessary to have some initial requirements. Those inputs are changeable to better please what the user wants and what he thinks it could adapt better to what he wants, in order to have different results when those values are modified. The main input values the software receives are:

- **Seed URL:** initial URL, the starting point where the web crawler will begin to get results;
- **Depth:** how deep will the web crawler go, in way to get more or less results;
- **Search query:** what the user is looking for, the results will depend on this;
- **Use Google or/and Yahoo to help the search:** possibility to avoid (or not) any link related to Google or Yahoo. This proposed software will be tested to see if it can get better results than those entities, however it does not mean necessarily that the software needs to reject the help of those search engines.
- **Value of weights:** to understand if a webpage has interesting information, it is necessary to classify it. The classification is made based on initials values from each weight.

After those inputs are filled, the software will be ready to begin its search, where the web crawler will begin to get results starting in the seed URL and finishing when the deep is reached. Those values can be changed by the user, however, in some fields there will be some default values and advised values.

3.4 Software Structure

The developed software is divided into five different webpages where each webpage has a main function. Those functions will be described in detail on the next chapter. On Figure 3.1 it is possible to see the structure of the proposed software and how the interaction is made between them.

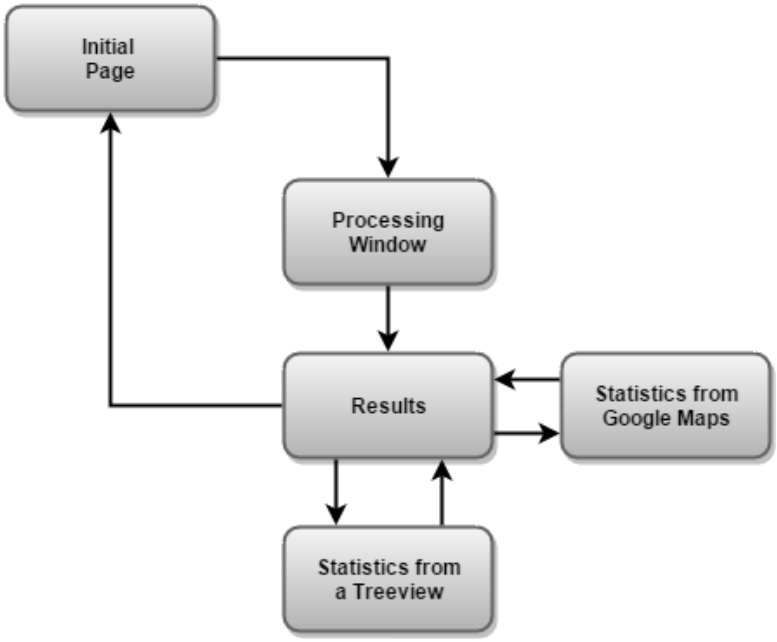

Figure 3.1: Structure of the proposed software.

The structure of each webpage is the following:

- **Initial Page:** Main menu, it is in here where the user introduces the initial inputs to get the software working;
- **Processing Window:** Place where the software processing can be visualized. It is in here where if there is any problem, the user can notice that, and find a solution;
- **Results:** Window showing the results obtained from all the processing;
- **Statistics from Google Maps:** Interactive window where it is possible to observe the location of each website using a Google Maps API;
- **Statistics from a treeview:** Interactive window, where it is possible to expande or to reduce the relation father-son of URLs;

3.4.1 Processing Cycle

The prototype's processing cycle related to the relevant information can be seen in the Figure 3.2.

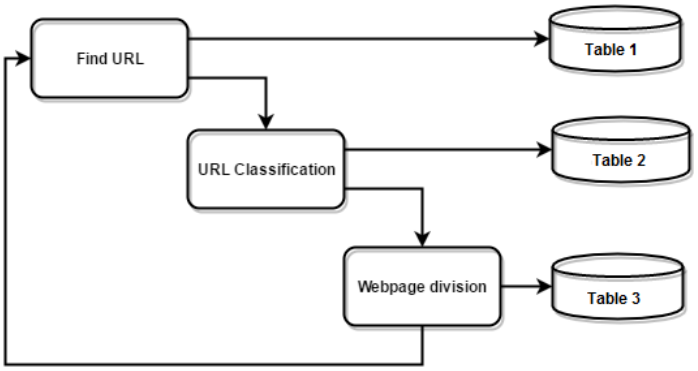

Figure 3.2: Processing cycle.

As mentiones before, there exists three different tables into a database, each one defined with a specific goal. For any URL found by the web crawler, it will be stored in the first table. Proceding to its classification, if a URL has a relevant classification, it will be stored in the second table, otherwise it will not be stored at all. For each webpage with an interesting classification, the next step is to divide the page into different sections in order to identify which section could contain the relevant information which can be interesting to the user. Later on, each section with relevant data will be stored in the third table.

An interesting concept to understand is what relevant information really means in this context. It can be described as the relevant data, the relevant content that a user was searching for. In way to obtain it, it is necessary to follow some main stages. Searching and finding the information are the first step and can be related with the "Find URL" and the "URL Classification" from the previous Figure. It is necessary to locate the most promising webpages instead of search in all webpages in order to improve the performance. Consequently Extract Information and Mining are related with "Webpage division". Applying these stages, it is possible to obtain the relevant information as final result. Figure 3.3 shows the several steps in order to get a website's relevant information.

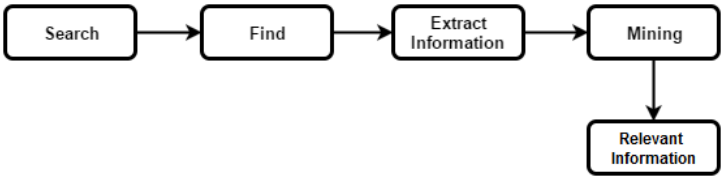

Figure 3.3: Steps to obtain a website's relevant information.

3.5 UML - Behavioral Diagram

3.5.1 ICE Diagram

ICE Diagram is for Interface, Control and Entity. In this subsection it will be represented a short description of how the system is built. Going into detail, ICE Diagram is divided in the following components:

- **Interface:** represents the inputs and outputs. The main goal of this field is to prepare relevant information for control classes to the users or other systems. It is shown in a "fancy" way;
- **Control:** it is the intelligence of the software. All the intelligence of the system it is found in this field. It is where the system receives the requests of any Use Case. At some point, the control components use or update the information found in the database which corresponds to the Entity field, where generates a result from the Use Case that was being in use;
- **Entity:** usually represents the database which contains the data of the system.

In Figure 3.4 it is possible to observe an ICE diagram relative to the relevant information acquiring system from the proposed architecture with the key blocks of the system.

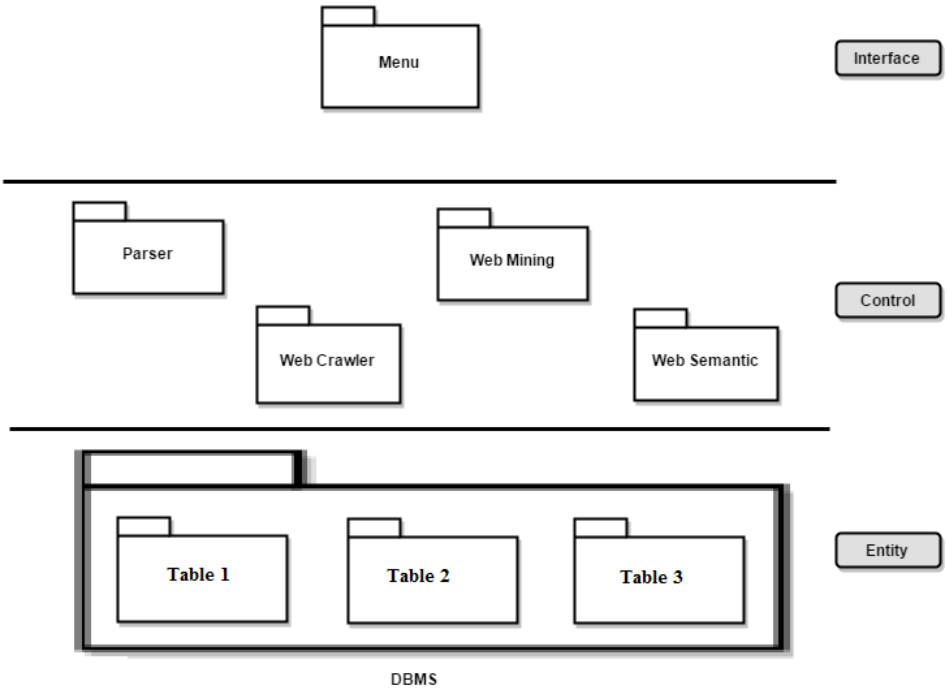

Figure 3.4: ICE diagram.

With the help of the diagram, it is possible to observe that the system will be composed by four main classes of intelligence and one database with three tables, each one with the main function of storing specific data.

3.5.2 Prototype Description

The following Figures have the purpose of helping to describe the prototype of an Use Case scenario. Figure 3.5 is related with the sequence that the software will have, and the Figure 3.6 represents the states transitions. However it is important to keep in mind that these diagrams are only a global illustration of the prototype and not a detailed scheme of the same.

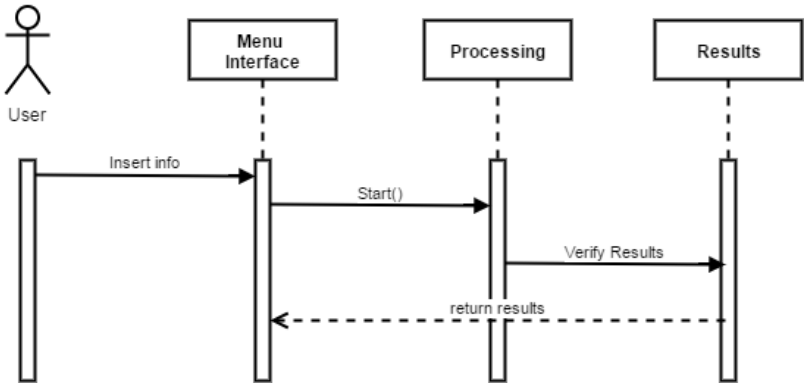

Figure 3.5: Sequence Diagram

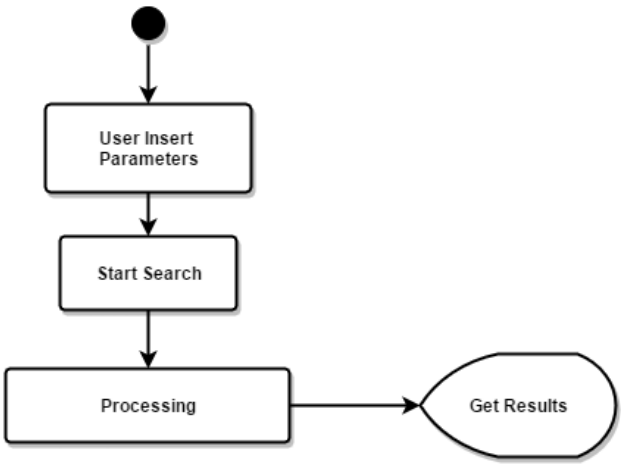

Figure 3.6: State Transition Diagram

The process represented is described by the following main steps:

- **Insert Parameters:** this field has the responsibility of receiving starting parameters and putting the software to work while making it adaptive to the user preferences.

Some starting parameters have a default value which represents a way of having some interesting results, but those parameters can differ according to user preferences;

- **Search:** starting the search based on the web crawler. It will go from one URL to another, starting on a seed URL;
- **Processing:** it will follow a set of rules that will help to define if an URL is interesting or not. If it is interesting, it will be saved in a database, otherwise it will be discarded;
- **Results:** here the obtained results will be represented. The user can choose how many results wants to see and according to that value, it will show the most relevant obtained results.

CHAPTER 4

Software Description

"In God we trust. All others must bring data" – W. Edwards Deming, statistician, professor, author, lecturer and consultant

This chapter specifies in detail the methods used in the development of the proposed prototype and the results obtained from its application. It is possible to say that the mechanics of the developed prototype is divided into three main blocks (parsing, web crawling and web mining) and so, the chapter starts with a brief study of three available parsers and the reason which led to choose the one that fits better in this project. After that it is described with detail the mechanics of web crawling and how it works followed by a description of web mining, which also contains semantic web.

4.1 Parser

As described earlier, a parser is responsible for the data analysis in a superficial way. Defining some initial conditions, it is possible to break the data into smaller pieces. This way, a parser can be very adaptable by the developer's will.

An example of a use case of a parser is the possibility of getting all the images available in a website. Or else to get a certain content from a website (the most viewed news from a news website). As the pillars are necessary to build a house, a parser has also an important role in the developed prototype because it will be the base of all this implementation.

Although the main idea of a parser can be simple, the development can be more complex because it is necessary to have in mind what should be taken from the website to build an efficient web crawler.

One of the first steps was to search for several parsing tools, in order to satisfy the

requirements for the goal in mind.

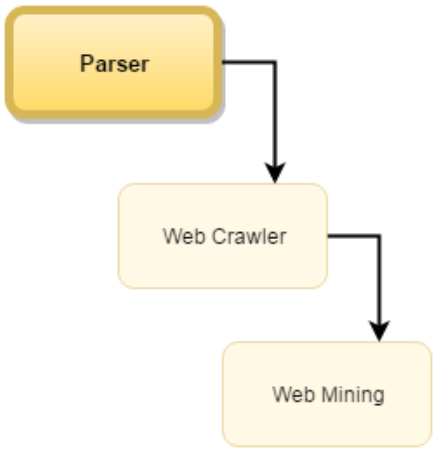

Figure 4.1: Parser Description.

The chosen parser to implement the proposed prototype was simple HTML DOM. The main reason for the choice of this software is because it is a tool easy to understand with a good amount of examples to follow and comprehend (which gives the upper hand in comparison to the others).

4.2 Web Crawler

This subsection describes the behavior of the implemented web crawler and the technical details.

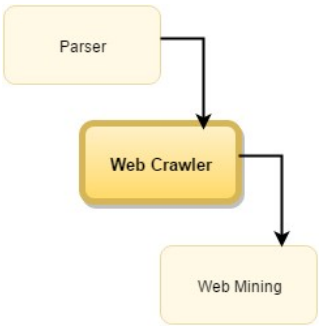

Figure 4.2: Web Crawler Description.

After introducing the input parameters, the web crawler is triggered and starts its task of visiting all the websites that are identified. As it is possible to analyze in Figure 4.3, the initial parameters are composed by:

- **Seed URL:** Initial website from where the web crawler will begin the search;
- **Depth Value:** According to this value, the web crawler will search more or less data. As logical, the higher the value, the wider the search will be over the internet;

- **Search Query:** The software will search for results related with this search query, in order to satisfy the user's information needs;
- **Number of links per page:** It will return the most promising URLs from a webpage. This field was added in order to increase the webcrawler's performance, otherwise it would be processed too many unnecessary webpages, leading to a waste of time;
- **Without Google, Yahoo?:** Activating this label, the web crawler will avoid any URL related to Google or Yahoo while searching;
- **Weights:** In a 0 to 1 scale, these parameters will be responsible to give more (or less) relevance to any appearence of any word from the search query in the following fields of a website:

  In the middle of site: Relevance given in the website's body;

  In title: Relevance given in the website's title;

  In URL: Relevance given in the website's URL;

  In metadata: Relevance given in the website's metadata.

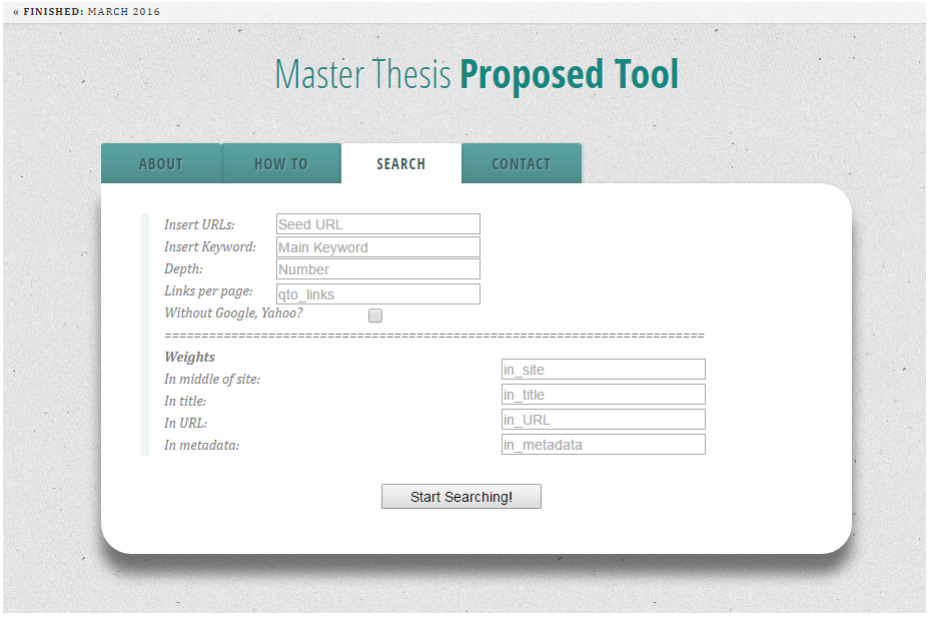

Figure 4.3: Main page of the proposed tool.

In order to get the best output from the proposed software, the sum of all weights should be equal to 1. The default values are:

- **In the middle of site =** 0.50;
- **In title =** 0.10;
- **In URL =** 0.20;

- **In metadata =** 0.20.

After clicking the start button, the web crawler is triggered for the first time and creates three databases, each one with different purposes. The first one, is responsible for storing all valid URLs from each analyzed webpage. It was defined a valid URL one that starts with "http://".

The second database was design to only store relevant URLs in order to improve the web crawler's performance. It is a fact that a webpage can have a lot of hyperlinks, however if the web crawler analyzes all the existents hyperlinks, it will lead to a waste of resources because in several webpages, only a few percentage of hyperlinks could be relevant. This database will be composed only by hyperlinks that will be analyzed. The selection rule of these URL will be described further ahead.

The last database, will store the relevant pieces of knowledge from a webpage. This knowledge will be acquired after applying web mining technique, which will be described in the next section.

At the time when the web crawler was being developed, analysing a website could lead to three main ways to finding a hyperlink. The ideal way would be to find one that starts by "http://". The other two ways are beginning by "www" or by "/". If the hyperlink starts by "www", it is necessary to append "http://" before that, in order to convert it to a valid hyperlink. When it starts by "/", it is necessary to discover the domain of the current website to be possible to add the domain before that piece of hyperlink. It is possible to observe the solution in a general way with Figure 4.4.

Figure 4.4: Process to obtain a website's domain.

Taking this as an example, what sometimes happens is, when analyzing the website "http://www.dmoz.org/Arts/Music", viewing the webpage's source code, it is possible to have a hyperlink like "/Arts/Music/Hiphop". However if we append this to the hyperlink that is being analyzed the result would be "http://www.dmoz.org/Arts/Music/Arts/Music/Hiphop" which is an invalid hyperlink. The valid step is to obtain the webpage's domain and then append it to the piece of the hyperlink.

One available feature when a webpage is being analyzed is the capability of trying to get a brief and correct description of the same webpage. It is possible to obtain a website's description by analyzing the metadata from the source code. However, it is necessary to have in mind that it is not possible to have always a good website's description. This happens because it depends on the way that the developer created the website. As logical, if the developer does not put a good description of the webpage, or

even if he does not put any description, the web crawler will not return any interesting result as it is possible to view in Figure 4.6. On the other hand, observing Figure 4.5 it is possible to see two good examples of a website's description.

Figure 4.5: Good website's descriptions.

Figure 4.6: Bad webpage descriptions. (a) Wrong description. (b) No description.

The following step is to verify if the webpage that is being analyzed already exists or not. If it is the first time that the webpage is being analyzed, it proceeds, otherwise it stops the current website analysis. When the web crawler was being developed, a fact to have in mind was to avoid equal webpages. This was important to avoid undesired loops which could lead to a waste of unnecessary resources and also lead to being forever stuck. This would ruin the idea of web crawling (by flooding the web).

If the webpage is being analyzed for the first time, the following fields will be stored in the database:

- **id:** id of the current entry;
- **website's id:** Current website's id;
- **website's name:** Current website's name;
- **son's id:** Son's id of the current website;
- **son's name:** Son's name of the current website;
- **depth:** Depth from the current website.

After those values being stored, the proposed algorithm will begin to obtain a webpage's relevance. This algorithm will be described in the next subsection but the main task is to get an evaluation of an URL.

According to the value obtained from the classification algorithm, the URL will be stored in one of two different lists. The first one was declared as being interesting and the other one as non interesting. As a matter of convenience, it was defined that if the final value of the proposed algorithm is above 25%, the URL will be stored in the interesting list, otherwise it will belong to the non interesting list. These two lists were created as a matter of convenience, because a determined webpage can have a vast list of hyperlinks. If all these links were to be analyzed, it would lead to a performance problem. This way it is possible to optimize the processing.

Depending on the URL that enter on those lists, it will happen one of the following three cases:

- **number of existing URLs in a webpage bellow the predefined by user's number:** at the beginning of the processing, if a user intends, for instance, to get the 20 most promising URLs from the webpage that is being analyzed and it only identifies 14, it will be stored in ascending order according to their relevance and, as logical, those 14 URLs will all be stored;

- **number of existing URLs in a webpage above the predefined number:** taking the previous example, if the user wants to get the 20 most promising URLs from the webpage and the webpage identifies 40, the 20 best classified URLs will be selected;

- **number of interesting URLs in a webpage bellow the predefined number:** if the user inserts to show the best 20 URLs and the webpage only identifies 12 URLs in the interesting list, it will also get the best 8 URLs from the non interesting list (even if the classification is bellow 25%);

After the analysis, the valid URLs are stored in the second database. This database contains potential URLs that can be relevant/interesting. It will store the same values that the first database stores and also the value of a webpage interest.

4.3 Web Mining

The web mining processing is divided into two main parts. The first one is responsible for the classification of a webpage in order to determine if the webpage has information that could be potentially interesting for the user or not. The second part, is responsible to split a website into different sections, to get the knowledge related to the website.

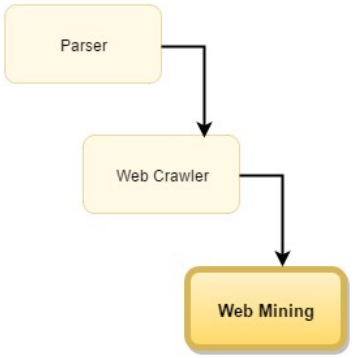

Figure 4.7: Web Mining Description.

4.3.1 URL Classification

When the web crawler starts its search for relevant content, that can be done in one of two ways. Either the web crawler crawls all the hyperlinks that were found on every visited webpages or else selects and follows the most promising hyperlinks from a webpage.

To obtain a webpage's relevance it was implemented an algorithm, which can be seen on Figure 4.8. According to the initial inputs of the weights introduced by the user, it will classify a webpage on a 0-100% scale.

The algorithm runs in a simple but effective way. In the webpage's source code, it will search for the words previously typed on the search query label. As it is possible to observe by the previous Figure, the algorithm splits the search query into individuals words. If the number of characters of one of those words is less than three, it will be discarted (in order to avoid irrelevant words such as "to", "of", "a", etc).

The next steps are similar but applied on different fields. For each word it will be searched if there is any reference in the metadata, title, hyperlinks and content (the website's body). The final value of each variable from each field is determined by the sum of each fraction being that each fraction is determined by the total number of words to be analyzed. At the end, the final values of each fields will be multiplied for each initial weight and then summed. The result will be multiplied by 100 (to have the final value as percentage). The following equation describes the mathematical form of the algorithm.

$$ProposedEquation = (ValMetadata + ValTitle + ValUrl + ValBody) \times 100 \quad (4.1)$$

Where:

$$ValMetadata = VarMeta \times WeightMetadataValue \quad (4.2)$$

$$ValTitle = VarTitle \times WeightTitleValue \quad (4.3)$$

$$ValUrl = VarUrl \times WeightURLValue \quad (4.4)$$

$$ValBody = VarContent \times WeightBodyValue \quad (4.5)$$

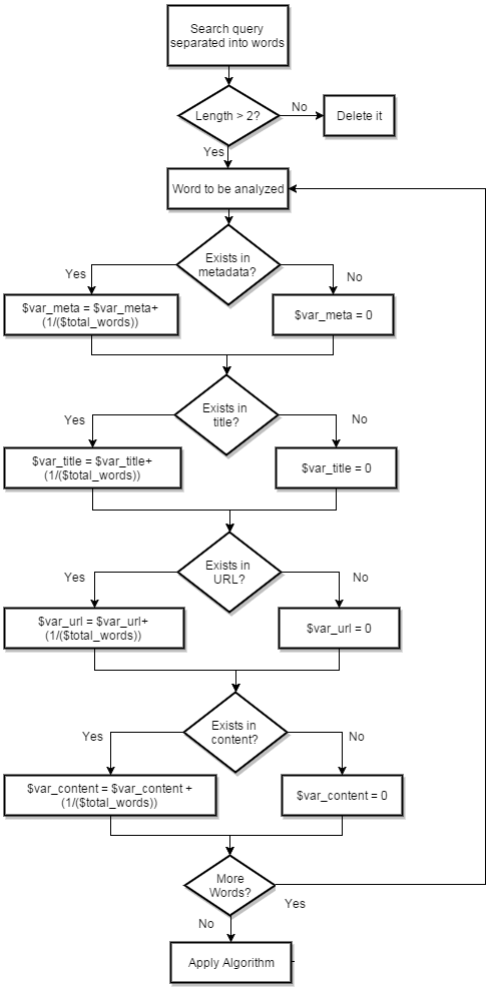

Figure 4.8: Proposed Algorithm

As mentioned before, the value of weights is configurable by the user because, for instance, in the metadata or in the title field, if the developer of a certain website did not put any information there, it does not show anything. So those values will be 0 in those fields. Making the values dynamic, the user can put a field with more relevance than the other. Another valid option is only to give a weight to the body of the website, discarding the other members.

The obtained result from the proposed algorithm, will decide if the analyzed website has interesting information, in order to analyze it later.

4.3.2 Spliting the website into different sections

In order to improve the reliability of web search results, it was decided to split a website into sections of information, according to its design. This way it is possible to get blocks of knowledge and discard those that are not important or do not have relevant information.

To proceed to the website's splitting, the first step is to break (again) the search query

into different words. As in the previous case, words with less than three characters are discarded for the same reason (irrelevant words are removed to improve the software's performance).

After splitting the website into different sections, these sections will be compared to each other in way to understand if there exists repeated information or not (this because sometimes the content from one section can be in another different section). However, before the section's comparison it is necessary to proceed to the cleaning of each section.

The section cleaning is made because, to be possible to compare different sections, we need to compare only the "real" content (what the user usually visually sees on a website). However, when extracting those sections, the content comes with a lot of HTML markup text and also can come with "garbage" American Standard Code for Information Interchange (ASCII) characters (such as new lines, tabs, white spaces, etc) which are not visual to the users but are visible to the software.

To proceed with the cleaning, all non visual characters and white spaces were deleted. This way it was possible to have the section's useful current size. After analyzing a set of around 20 random different websites it was found that sections with similar sizes were identical in content. So, after identifying all sections, only one was left and all the others with similar size were deleted. One other fact that was found was that blocks with less than 100 characters of size, usually do not have related useful content. They are more related with advertising content of repeated content from a bigger section, so those sections were, as well, discarded.

The sections that were not deleted were stored in a list, and according to their size, were sorted in ascending order.

To get the website's knowledge, the final step is to put the search query into lowercase letters as well as the website's content. This is done to be possible to compare all words without the case sensitive problem. If in a section there is no reference to a search query, that means that that section is not important. Otherwise represents the website's knowledge. After doing that to all words of the search query it will save the different sections of knowledge in the final database. The one which is related with the knowledge of the search.

At the end of comparing all words, the program cleans the variables, freeing the alocated memory in order to proceed to the analysis of new different URL without any memory problem.

4.3.3 Semantic Web

Semantic web has an interesting role on the development of this prototype because it helps and can complement the user's search. Figure 4.9 represents one example where semantic web is applied. For instance, for the search query "Ski in europe with good prices", applying semantic web it is obtained:

Figure 4.9: Application example of semantic web.

Analysing the Figure, the component SocialTag is what it matters. According to the initial search query, it will return as suggestion, the words "Ski", "Elan SCX" and "Winter sports in Slovakia". It is possible to say that this three results are related with the initial search query. "Elan SCX" is a skiing brand and "Winter sports in Slovakia" can be a little bit uncertain, but it is easily related with ski. Of course the results obtained from the use of semantic web, sometimes can be inaccurate. However, this technique is used to try to have a wider search zone in order to get quicker results from the initial search query. Since those words can be related with what the user is searching for, any time it is found any occurence of each word, it will give some importance because it means that can be related with what the user is looking for.

To apply this technique, it was used *Open Calais*, a PHP class for extracting entities and tags. This software performs semantic analysis of the text and has the capability of doing it using natural processing language. [12]

4.4 Applying Bootstrap

It is said that usually, people eat with their eyes first, which means that the look of anything is an important factor to have in mind. Figure 4.10 represents the beginning of the prototype's interface.

Using Cascading Style Sheet (CSS), it is possible to have an interface much more appealing and not so "raw". With help from [4] and integrating with my prototype, it was possible to obtain the following interface, as can be seen in Figure 4.11.

4.5 Final Results

Figure 4.12 shows part of the obtained result from the search query "Specifications and price of ASUS Zenbook UX305FA". It is possible to analyze and see that the most revelant webpage found has a classification of 76.666%. It is possible to see the website splited into different sections of knowledge, where in bold and orange color is highlighted the

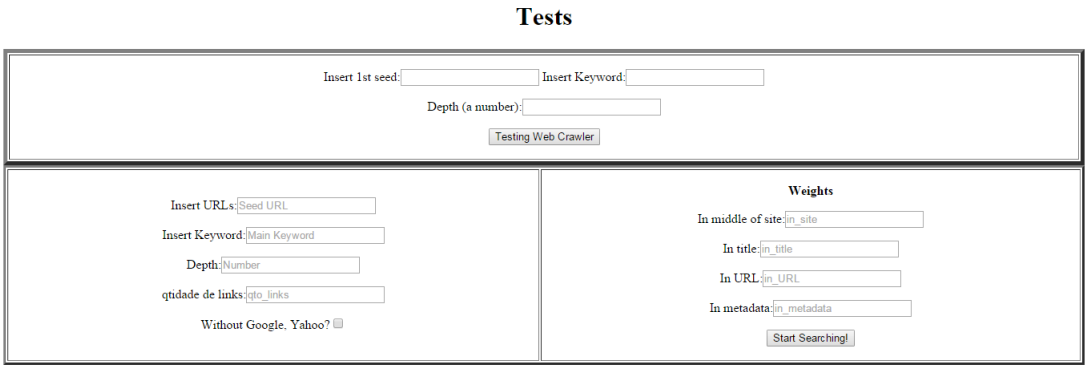

Figure 4.10: Proposed tool without Bootstrap.

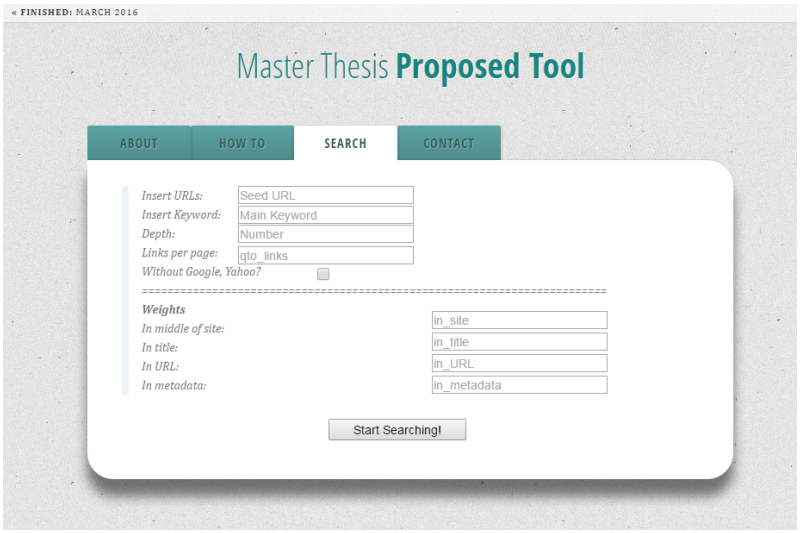

Figure 4.11: Proposed tool with Bootstrap.

words belonging to the search query. However the output is not the ideal which can be a difficult to analyze what was obtained. This happens due the fact that, as described earlier, this blocks of knowledge are stored in MySQL database. Hereupon, to be stored in MySQL it needs to follow some rules and sometimes, those rules, conflict with what is stored. Some HTML tags are not valid to be stored into MySQL, and those tags are responsible for the text formatting. Since those HTML tags are deleted to be possible to store into the database, it is more dificult to format the text. However, thinking in some specific cases such as slow internet connection, the user can find a better utility with this scenario despite the inconvenient of the output is not so appealing instead of trying to find the webpage which can contain (or not) the information he really needs.

Figure 4.13 represents the correspondent webpage. It is possible to observe that the result meet the expectations since the website has the intended information.

However, analysing both Figure 4.13 and Figure 4.12 it is as well possible to see that the gathering of the website's knowledge is not working 100% correctly. It is possible to

see by the colors the correspondent relation, however, in Figure 4.13, the block that is not marked with any color is because it selected all the website's information without doing any distinction. This is not completely wrong, only shows more than the expected, what can be undesirable.

In appendix it is possible to observe more results for different search queries.

4.5.1 Tree Results

One of the available features in the proposed architecture, is the possibility of having the website's treeview. Figure 4.14 shows the beginning of the treeview and Figure 4.15 shows the complete treeview.

This treeview represents all followed paths by the web crawler for the typed search query. It is interactive in the way that is possible to expande and to minimize each root. This feature was done in order to satisfy the possible curiosity from the user in analyzing all websites which were related in the current search.

It was made with help of [16].

4.5.2 Google Maps Results

In order to have a perception from where the web crawler was, it was used a *Google* API [11], so this way it is possible to know the different websites from where it past.

Figure 4.17 shows the location of each website geographically. However it does not represents all the hyperlinks but yes the correspondent website. It was decided to represent this way because since there are several hyperlinks belonging to the same website, it would overloaded the map representation.

However, to know where to represent each website on *Google Maps*, it is necessary to know their location. For this, it was used an API [7] to get a website's location based on its IP address. To get the correction location, the website's hyperlink needed to be treated cutting of the "http://" and the final "/". This way it was possible to obtain the latitude and longitude in order to represent on *Google Maps*. Figure 4.16 shows the API application.

Figure 4.12: Retail Results.

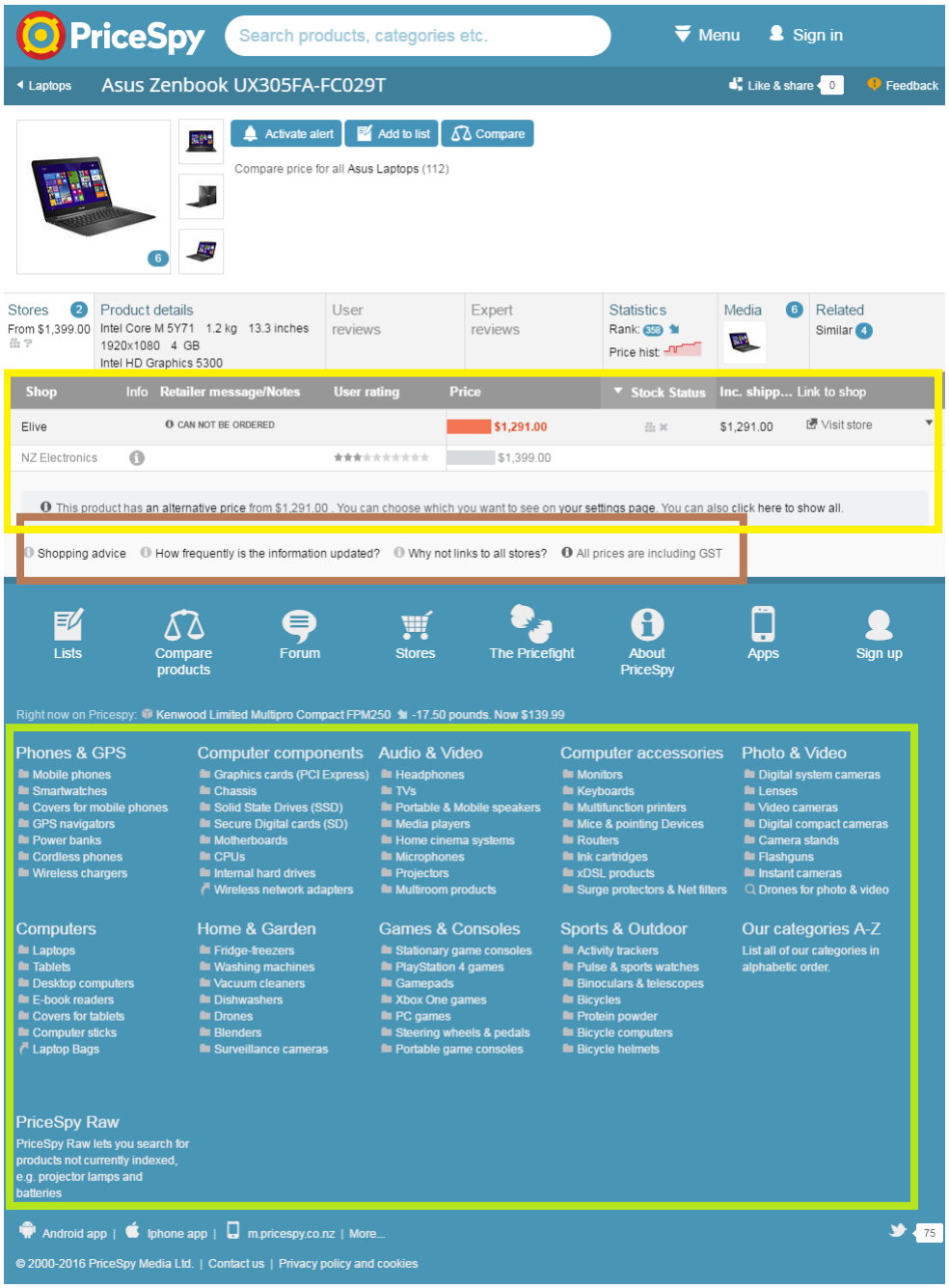

Figure 4.13: Retail first website from results.

Figure 4.14: Initial treeview.

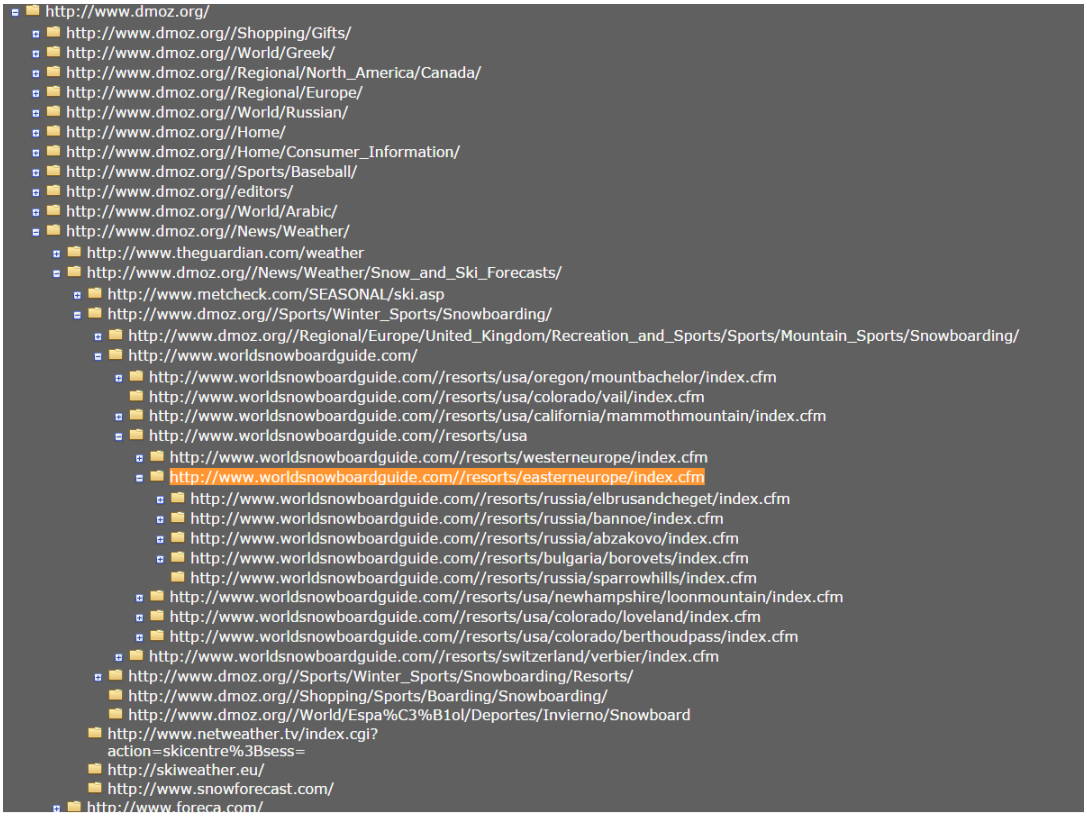

Figure 4.15: Complete treeview.

Figure 4.16: Getting the coordinates on a website's location.

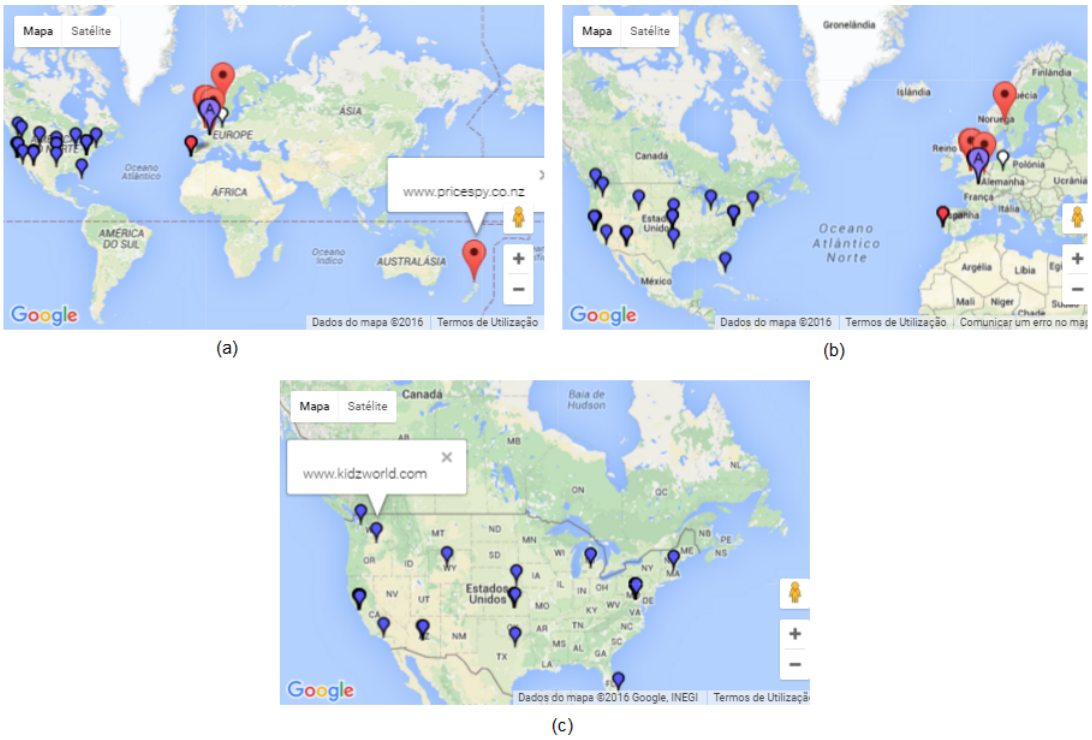

Figure 4.17: Google maps representation.

CHAPTER 5

Conclusion and Further Work

"Spiders are the only web developers that are happy to find bugs." – Unknown

It is possible to say the main purpose of this dissertation is based on the study of web search engines because after the study, comprehension and junction of each technique described over this document, it was possible to obtain a search engine. It was done to understand in which way it is possible to improve the reliability of web search results. It was an interesting challenge because it led to a greater understanding of the web, how it works and how it is possible to take more advantage of its resources.

After this long journey, it is possible to do an insight about all the done work. How it was been said over this project, it is possible to split all the developed prototype into three main techniques: parsing, web crawling and web mining. There is not much to say about parsing because it can be considered a small detail in all this work (however completely fundamental in order to put all techniques connected). About web crawling, it is known there are several available open source web crawlers on the web, but it was thought that could be interesting to do a new web crawler. First of all to understand completely the logic behind the main goal, the problems that happen when creating it, how to solve them and there is no better way to understand a technology than to build it. With this, it's not reinvent the wheel, but yes to understand exactly how the wheel works and adapte it in the best way to fit our goals. Finally web mining. It is an interesting field of technology because can be really wide with a lot of different goals, depending of where it is used. It was as well used semantic web in order to help the mechanic of web mining.

At the end of all this work, there are some relevant considerations to do. One of them, is related with the web crawler. One of the biggest frustrations was the fact the web crawler can only "load" text content related and not images or other types of documents

such as pdf's, .docx, etc. It is common to find in a website, for example, an hyperlink that is a direct link to a pdf, one of the invalid files to read. When the web crawler tries to load the hyperlink, what happens is that it runs out of memory. This is because it was trying to load an invalid hyperlink. It was difficult to find this error and frustrating because sometimes it took to much time until this proposed tool stopped working. When this error was found, it was developed the option to rerun the program instead of run from the beginning. However to rerun it is necessary to go down to the code and change a variable.

There are some points that need to be outlined, such as the results' output. Despite the fact that the output is not so appealing as intended and in sometimes it gets more content than the expected, it is possible to say that the obtained results are satisfactory. Depending on the application scenario it can be useful to the user. However, as it is referred in [3], websites could be divided into three main categories (commercial, service and mixed type), and each category has subcategories. It would be interesting if for each category, a different processing would be done, in order to achieve better results to get the correspondent website's knowledge.

Thinking on all the done work, it is possible to say that corresponds to the expected because it was proved the concept that it is possible to improve the reliability of web search engines. Now it all depends where it is used. Probably in big cities and in areas with good internet connection, the conventional search engines works perfectly and the description to each hyperlink is sufficient to meet the user's expectations. If the user clicks on a website that does not correspond to what he eventually was searching for, easily and quickly he can click on the next hyperlink. However outside of the big cities or even in the big cities but if the internet connection is slow, it is believed that users can take advantage of this method because even the output is not perfect, it is perceptible enough to the user sees if a hyperlink has what he is looking for. Or even the user can get the answer of what he was looking for, only by seeing the output. It all depends on where this tool is applied and the conditions where it is applied. If it is not possible to improve the internet connection, at least it can be possible to search ways to improve the experience that a user can have on the web.

Other achieved conclusion was related with the applications of this presented prototype. Since at certain point the search is much more centralized, it could be applied in specific business such as a web search engine centralized on business areas. The developed tool could have other particularities and uses.

5.1 Further Work

Like practically every project, there are always some options to correct or some features that would be interesting to have. Either to improve the software's performance or in order to get better results, there are always anything that could be done to be improved.

After an insight of all the work done over this dissertation, some of the most interesting features/options detected that could be done are:

- Improve how the output is shown, the representation of the obtained knowledge of each website. Probably this is one of the main features which would be interesting to be upgraded. Not because it is not working good but for the fact that the output could be more appealing. The aspect is a really important feature because users can (or can not) use determined tool based on its aspect;
- Creation of an option in order to get and store different content (such as images), to be possible to the user the visualization of different files instead text only;
- Identify all invalid hyperlinks in order to improve the performance of the software. Hyperlinks which lead to a certain type of file such as a direct download, a pdf, etc;
- When the web crawler tries to access an invalid hyperlink, it would be interesting to have a feature capable to give to the user the opportunity to add hyperlinks terminologies to avoid and other button to rerun the program (instead of changing manually into the code);
- Web crawler improvement in way to follow different webpages when it is too focused in a certain domain;
- Changing the pinpoints colors of google maps. For example, it would be interesting the greater the depth, the darker the color would be;
- A deeper investigation in order to conclude if the application of the proposed prototype would be interesting to use in a business area.

Of course there are other features that could be done as a future work, however the referred ones are the ones that would be the next step to put this work more robust, efficient and even more attractive.

Bibliography

[1] Bioinformatics. *htmLawed documentation*. Accessed: 2016-02-20. 2016. URL: http://www.bioinformatics.org/phplabware/internal_utilities/htmLawed/htmLawed_README.htm#s4.6.

[2] BrightPlanet. *Clearing up Confusion – Deep Web vs. Dark Web*. Accessed: 2016-03-06. 2016. URL: http://www.brightplanet.com/2014/03/clearing-confusion-deep-web-vs-dark-web/.

[3] S. Cebi. "Determining importance degrees of website design parameters based on interactions and types of websites". In: (2012), p. 1031.

[4] Codrops. *Animated Content Tabs with CSS3*. Accessed: 2016-02-01. 2016. URL: http://tympanus.net/codrops/2012/04/12/animated-content-tabs-with-css3/.

[5] S. Consultants. *The XML FAQ*. 2015. URL: http://xml.silmaril.ie/parsers.html.

[6] S. D. *Crawling with OpenWebSpider*. Accessed: 2016-01-21. 2016. URL: http://scriptsonscripts.blogspot.pt/2015/10/crawling-with-openwebspider.html.

[7] devzone. *Find location from IP address in PHP using Geoplugin API*. Accessed: 2015-12-18. 2015. URL: http://devzone.co.in/find-location-ip-address-php/.

[8] eBizMBA. *Top 15 Most Popular Search Engines, March 2016*. Accessed: 2016-03-02. 2016. URL: http://www.ebizmba.com/articles/search-engines.

[9] R. Enikeev. *The Internet Map*. Accessed: 2016-02-08. 2016. URL: http://internet-map.net/.

[10] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996, pp. 1–34.

[11] Google. *Adding a Google Map to your website*. Accessed: 2015-12-15. 2015. URL: https://developers.google.com/maps/tutorials/fundamentals/adding-a-google-map.

[12] D. Grossman. *Open Calais Tags*. Accessed: 2016-01-18. 2016. URL: http://www.dangrossman.info/open-calais-tags/.

[13] A. Heydon and M. Najork. "Mercator: A Scalable, Extensible Web Crawler". In: *World Wide Web* 2 (1999), p. 219.

[14] U. Hunfeld. *PHPCrawl Testinterface*. Accessed: 2016-01-23. 2016. URL: http://phpcrawl.cuab.de/testinterface.html.

[15] U. Hunfeld. *PHPCrawl webcrawler library/framework*. Accessed: 2016-01-19. 2016. URL: http://phpcrawl.cuab.de/about.html.

[16] JavascriptKit. *Cut and Paste Folding Treeview Menu*. Accessed: 2016-02-03. 2016. URL: http://www.javascriptkit.com/script/treeview/.

[17] K. O. Khorsheed, M. M. Madbouly, and S. K. Guirguis. "Search Engine Optimization Using Data Mining Approach". In: *Computer Engineering and Applications* IX (2015), pp. 188–189.

[18] F. Kokkoras, K. Ntonas, and N. Bassiliades. "DEiXTo: A Web Data Extraction Suite". In: (2013), pp. 9–10. URL: http://lpis.csd.auth.gr/publications/BCI2013-kokkoras.pdf.

[19] A. Madkour. *"Semantic Web Mining: Using Association Rules for Learning an Ontology"*. Accessed: 2016-02-23. URL: https://www.cs.purdue.edu/homes/amadkour/presentations/SemanticWebMining.pdf.

[20] S. M. Mirtaheri, M. E. Dinçktürk, S. Hooshmand, G. V. Bochmann, G.-V. Jourdan, and I. V. Onut. "A brief history of web crawlers". In: *arXiv preprint arXiv:1405.0749* (2014).

[21] MOZENDA. *Data Mining Software*. Accessed: 2016-01-18. 2016. URL: http://www.mozenda.com/data-mining-software/.

[22] MOZENDA. *Transform The Internet Into Strutured Data*. Accessed: 2016-01-16. 2016. URL: http://www.mozenda.com/.

[23] M. Najork. "Web Crawler Architecture". In: Springer Verlag, 2009. URL: https://www.microsoft.com/en-us/research/publication/web-crawler-architecture/.

[24] C. Olston and M. Najork. "Web Crawling". In: *Foundations and Trends in Information Retrieval* 4 (2010), pp. 180–183.

[25] openwebspider. *OpenWebSpider Open Source Web Spider and Search Engine*. Accessed: 2016-01-21. 2016. URL: http://www.openwebspider.org/documentation/openwebspider-js/.

[26] W. de Pádua Paula Filho. *Engenharia de Software: fundamentos, métodos e padrões*. LTC, 2000, p. 13.

[27] M. Porter. *The Porter Stemming Algorithm*. Accessed: 2016-01-19. 2016. URL: http://tartarus.org/martin/PorterStemmer/.

[28] Q. K. Quboa and M. Saraee. "A State-of-the-Art Survey on Semantic Web Mining". In: *Intelligent Information Management* 5 (2013), p. 10.

[29] C. Semantics. *Introduction to the Semantic Web*. Accessed: 2015-12-14. 2015. URL: https://www.cambridgesemantics.com/semantic-university/introduction-semantic-web.

[30] T. H. of SEO. *Short History of Early Search Engines*. Accessed: 2016-03-07. 2016. URL: http://www.thehistoryofseo.com/The-Industry/Short_History_of_Early_Search_Engines.aspx.

[31] SEOmoz. *What is Page Authority?* Accessed: 2015-12-27. 2015. URL: https://moz.com/learn/seo/page-authority.

[32] I. L. Stats. *Total number of Websites*. Accessed: 2016-03-02. 2016. URL: http://www.internetlivestats.com/total-number-of-websites/.

[33] G. Stummea, A. Hotho, and B. Berendt. "Semantic Web Mining State of the art and future directions". In: *Web Semantics: Science, Services and Agents on the World Wide Web* (2006), pp. 128–129.

[34] A. Uebe. *Instalando e utilizando o Web Crawler OpenWebSpider*. Accessed: 2016-01-21. 2016. URL: https://www.vivaolinux.com.br/artigo/Instalando-e-utilizando-o-Web-Crawler-OpenWebSpider?pagina=2.

APPENDIX A

This chapter is responsible to show the different flowcharts used to implement the prototype.

A.1 Web Crawler and Web Mining

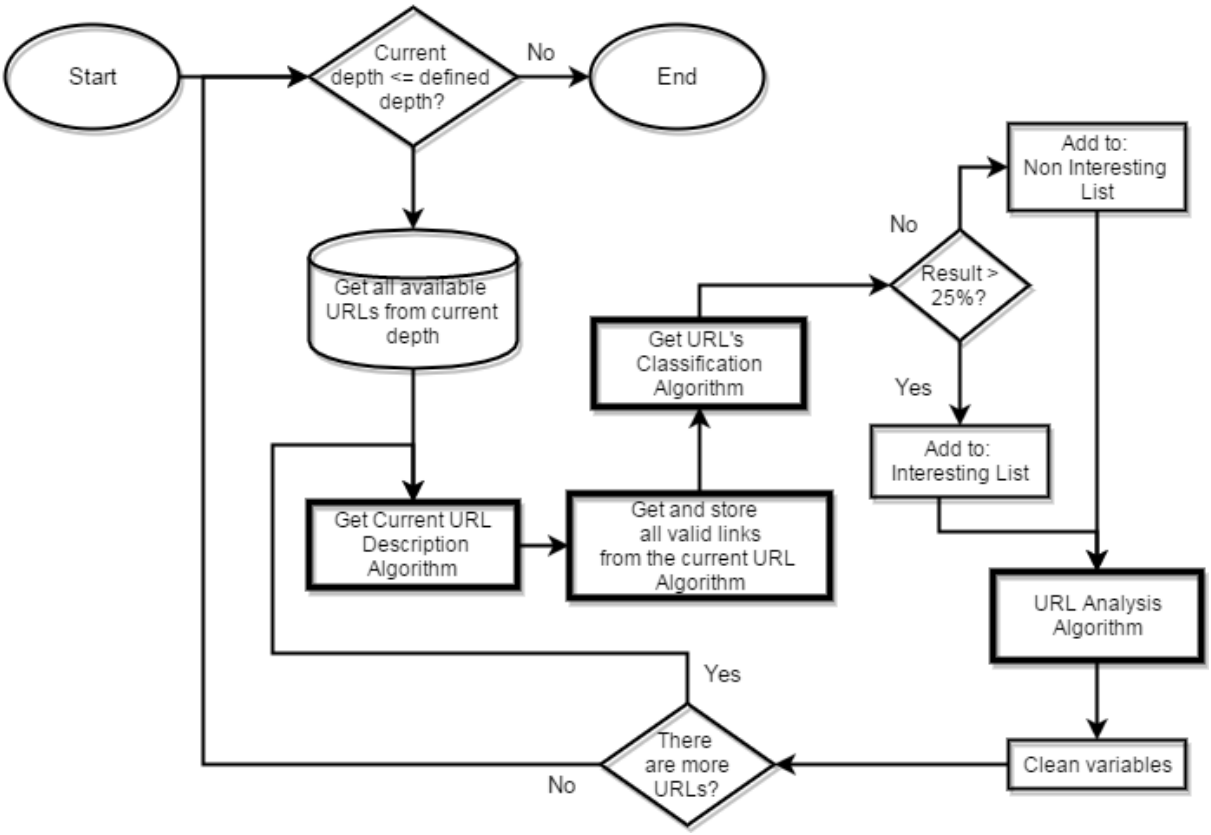

Figure A.1: Webcrawler General View

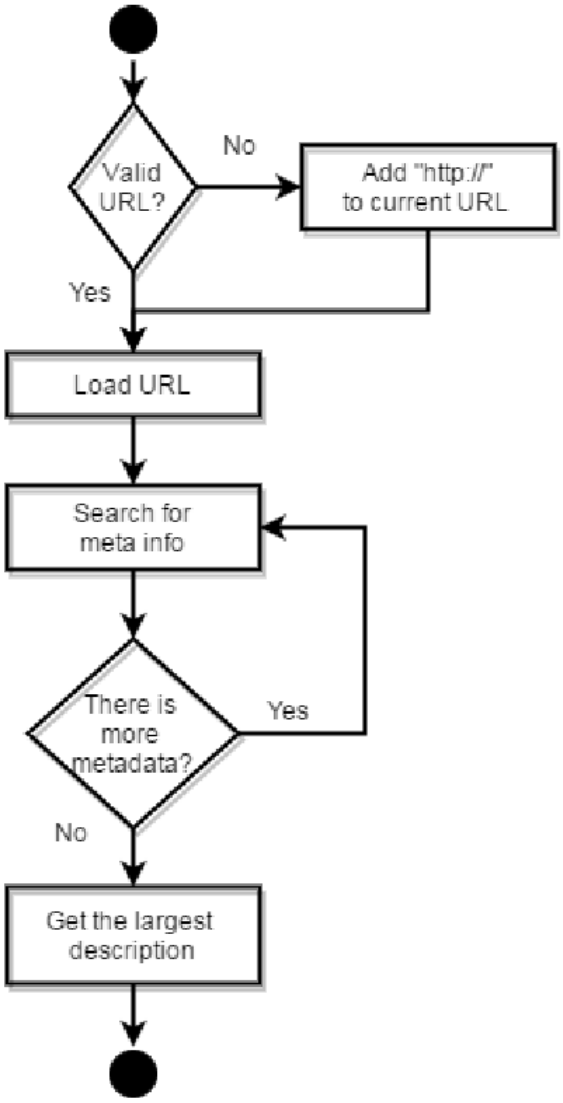

Figure A.2: Web Crawler - Get URL description

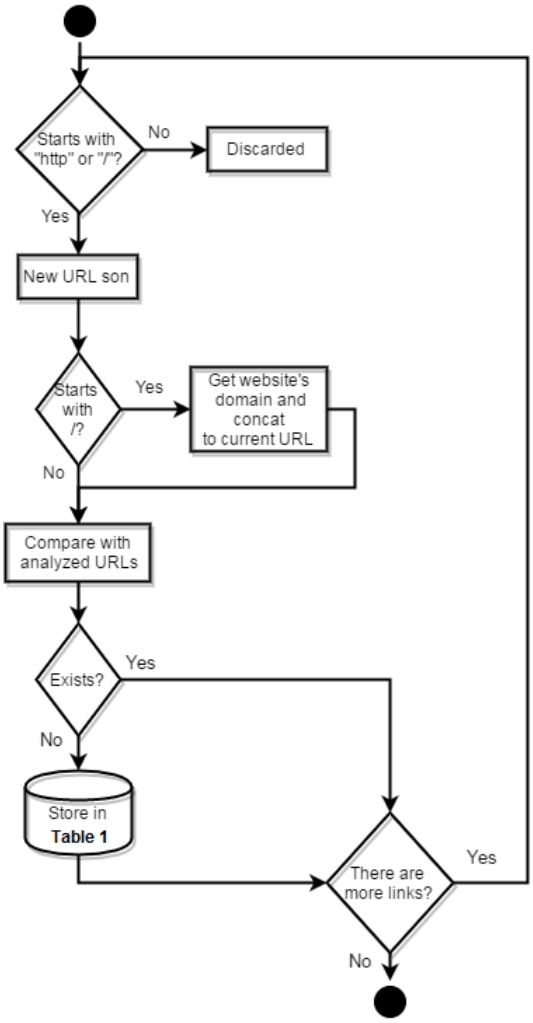

Figure A.3: Web Crawler - Get and Store Valid URLs

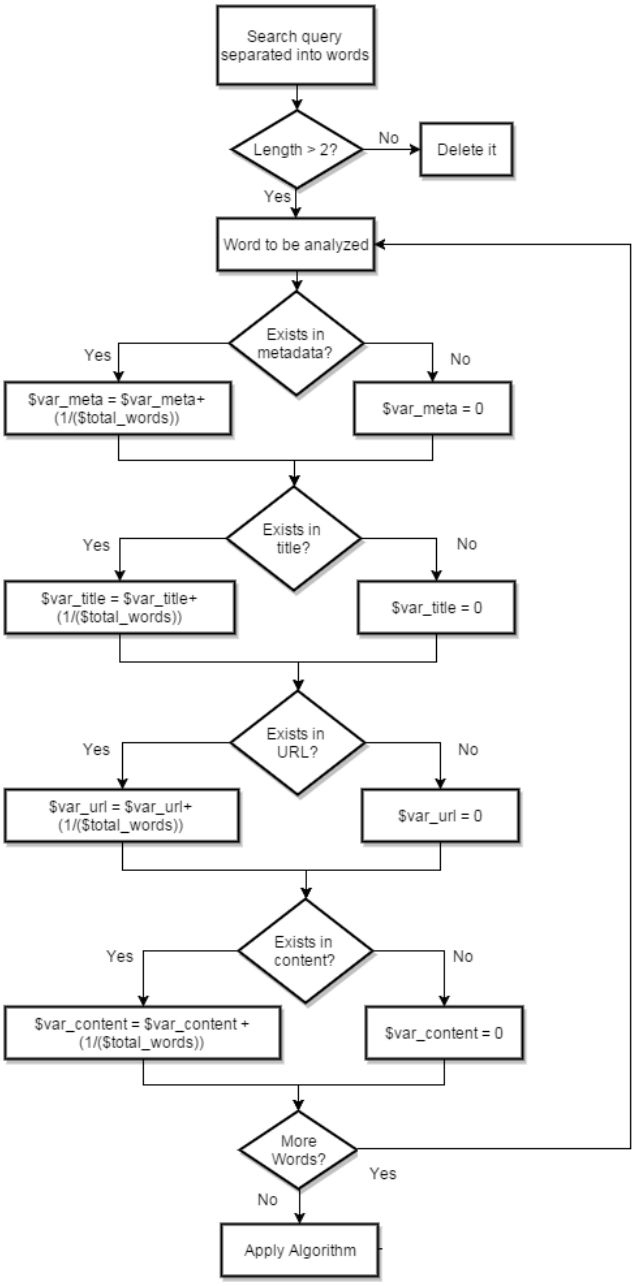

Figure A.4: Web Crawler - Get URL Classification

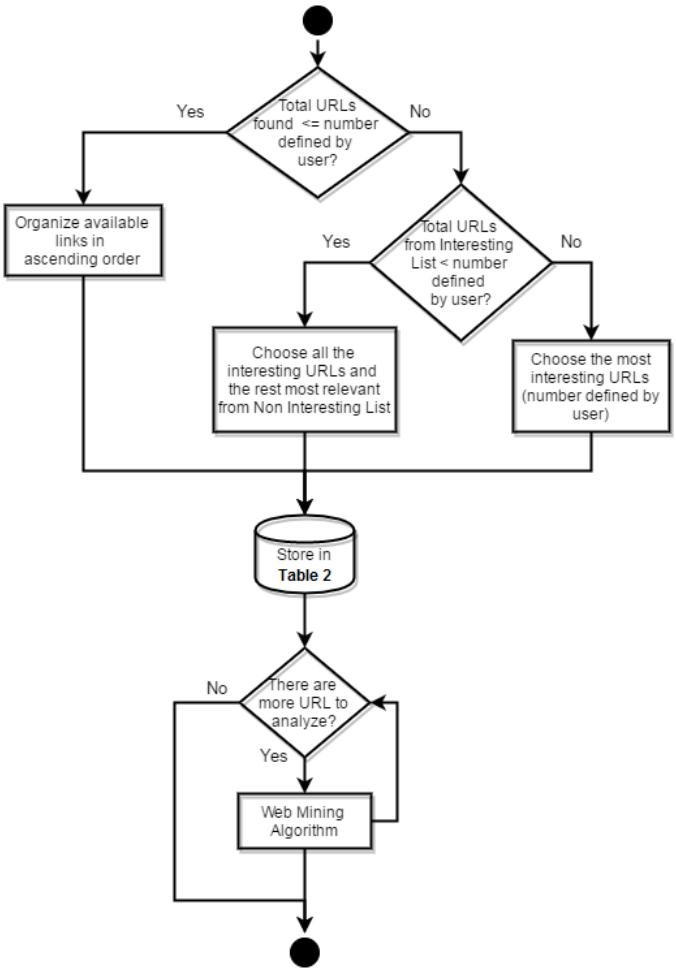

Figure A.5: Web Crawler - URL Analysis

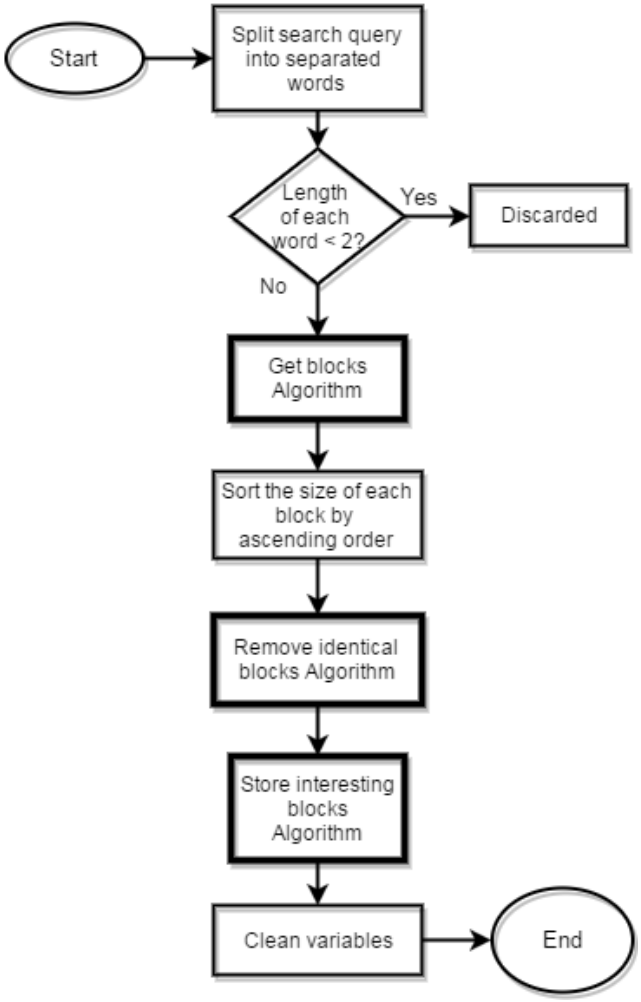

Figure A.6: Web Mining Main Representation

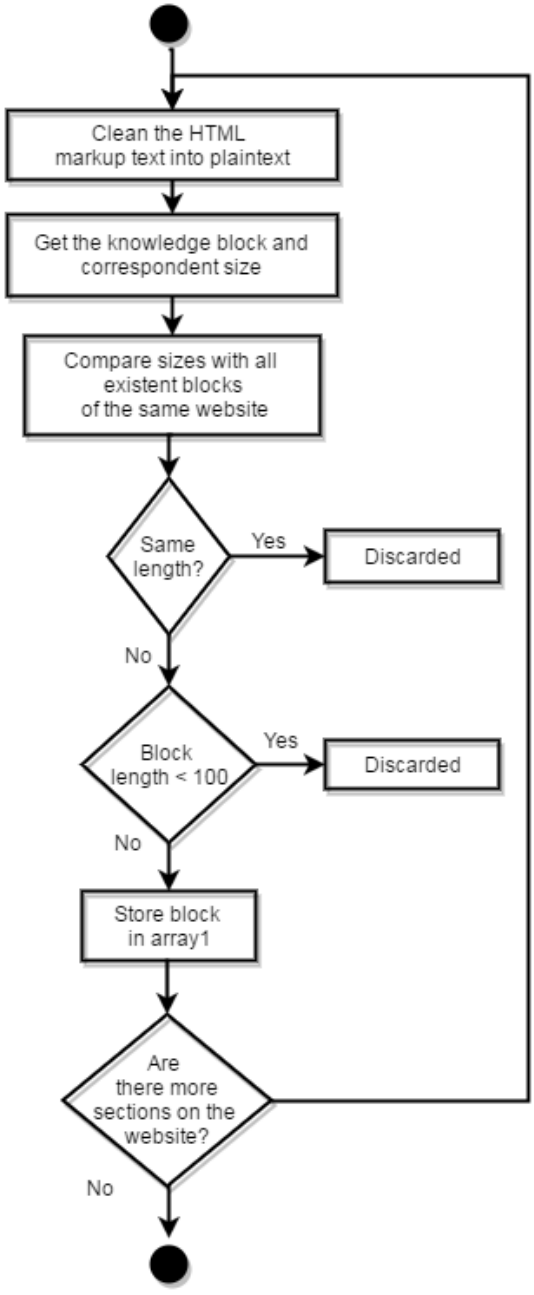

Figure A.7: Web Mining - Get Block.

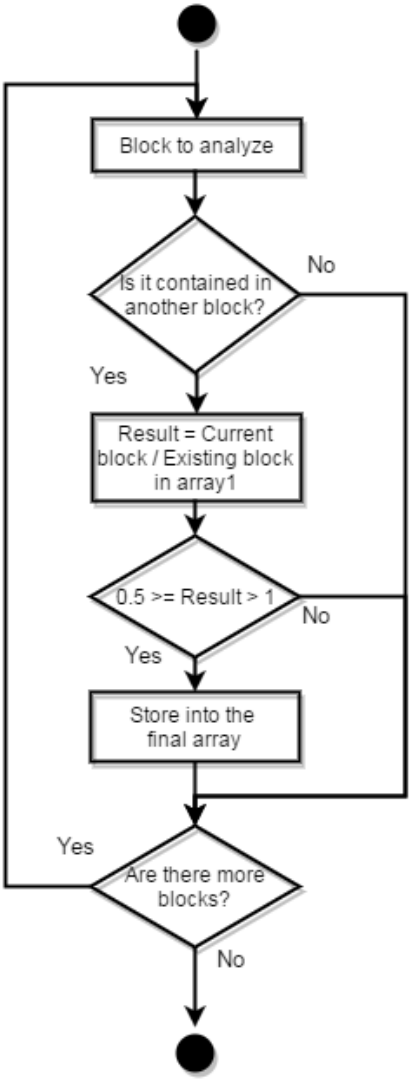

Figure A.8: Web Mining - Remove Identical Blocks.

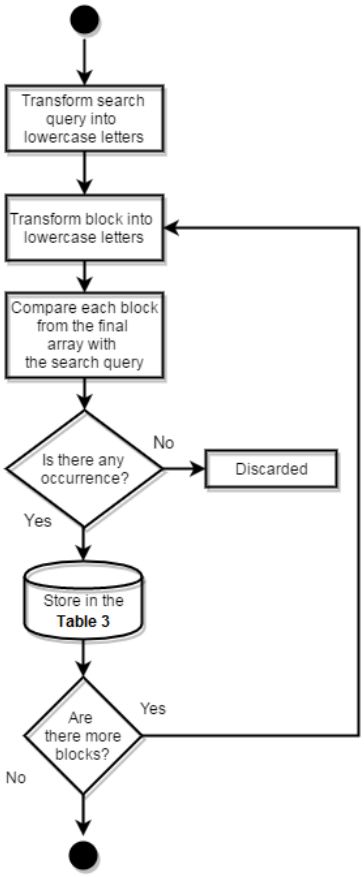

Figure A.9: Web Mining - Store Interesting Blocks.

APPENDIX B

More Results

B.1 Tourism Related Results

With the initial search query "Where to kitesurf in Portugal" it is obtained:

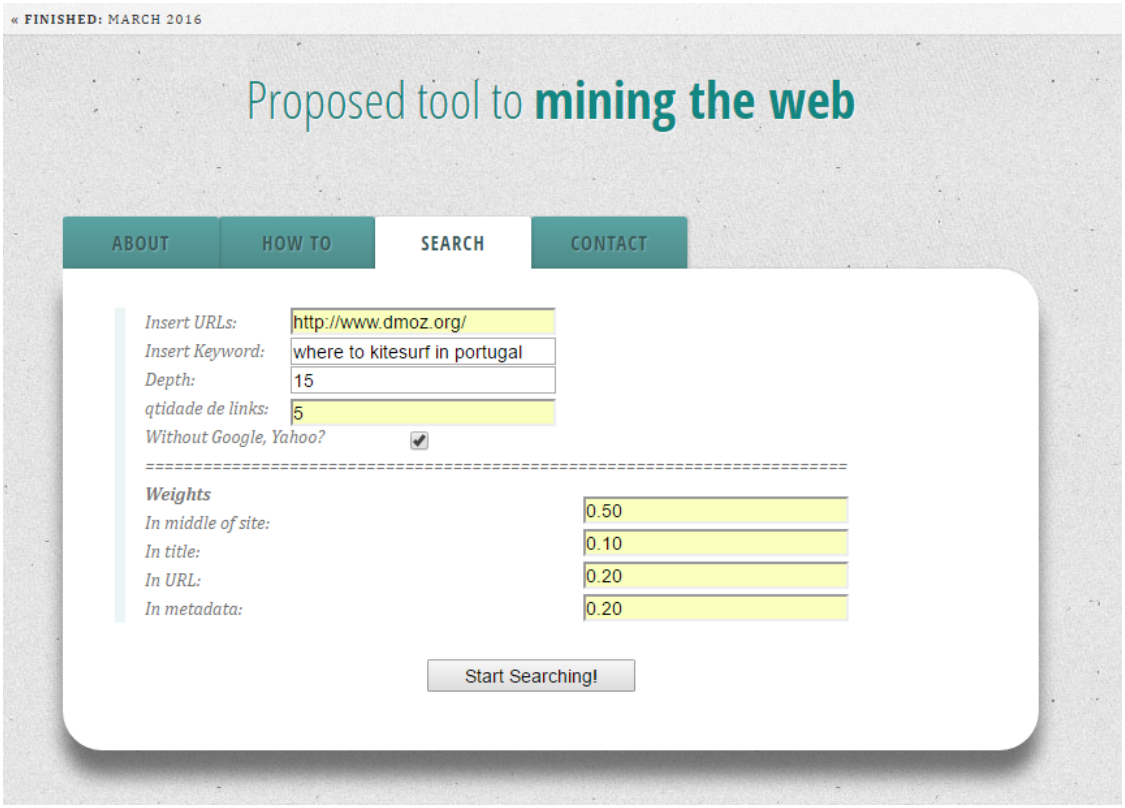

Figure B.1: Main menu with the initial inputs.

B.1.1 Obtained Results

Figure B.2: Results.

Observing Figure B.2 and Figure B.3, it is possible to analyze that the first result

obtained does not corresponds totally to what the user was searching for (despite the fact that it is related). However analysing more results, it is possible to find a more correct answer for what is searching for. Figure B.4 and Figure B.5 represents other results.

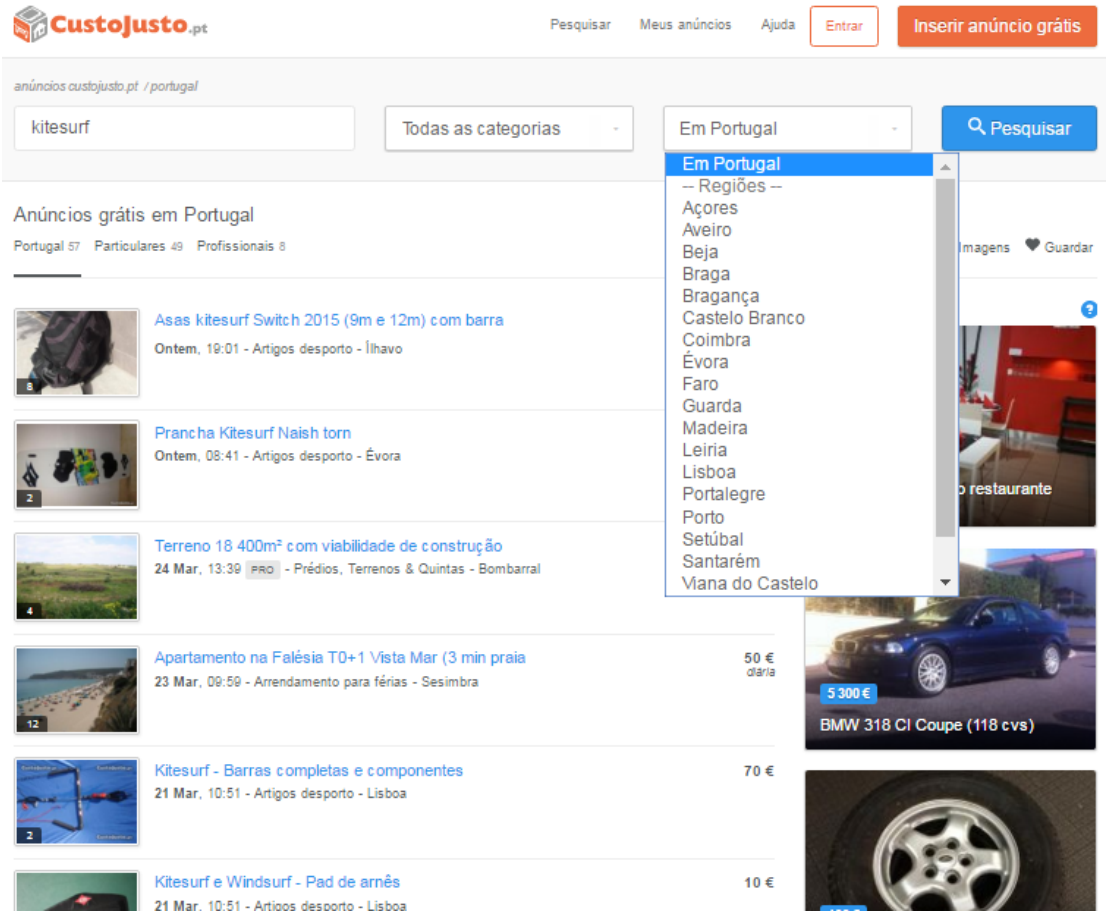

Figure B.3: Correspondent website.

More info in:
http://www.wannakitesurf.com//spot/Europe/Portugal/Viana_do_Castelo/Cabedelo/photo/index.html?wdaction=lib.WDPagePhoto.show&file_id=204#wanna-content
Classification of the URL: 70 - from deepth = 9

id: 5028 - Data: wannakitesurf.com 24/24 wannakitesurf.com on your mobile http://m.wannakitesurf.com rss all the rss feeds of wannakitesurf.com newsletter all news by email

---

id: 5031 - Data: lost password? new user? register my kitemy profileedit my profilemy travel mapmy kite spotsmy kite sessionsmy kite tripsmy photosmy messagesmy blogswrite a blog wannakitesurferscommunityforumchat photosblogsmembers listmembers groupskite tripskite sessions

---

*id: 5030 - Data: home spots europe **portugal** viana do castelo cabedelo photo submit new photos cabedelo viana do castelo photo 1 of 1 show all (1)... it's not the right spot? hoax? copyright infringement? tell us ! photo by anonymous, 05-11-2006 comment : cabedelo vento leste*

---

*id: 5032 - Data: africa (46 countries) algeria angola benin burkina faso burundi cameroon cape verde comoros congo democratic republic of the congo djibouti egypt equatorial guinea eritrea ethiopia gabon gambia ghana guinea guinea-bissau ivory coast kenya liberia libya madagascar malawi mauritius mayotte morocco mozambique namibia nigeria reunion island rwanda s?o tom? senegal seychelles sierra leone somalia south africa sudan tanzania togo tunisia western sahara zimbabwe asia (23 countries) azerbaijan bhutan british indian ocean territory brunei burma china east timor hong kong india indonesia japan kazakhstan malaysia maldives north korea philippines russia singapore south korea sri lanka taiwan thailand vietnam australia & pacific (21 countries) australia christmas islands cook islands easter island federal state of micronesia fiji french polynesia kiribati nauru new caledonia new zealand niue island northen mariana island papua new guinea robinson crusoe islands samoa american solomon islands tonga vanuatu wallis and futuna western samoa central america (39 countries) anguilla antigua and barbuda aruba barbados belize bermuda bonaire island british virgin islands cayman islands costa rica cuba cura?ao dominica dominican republic el salvador grenada guadeloupe guatemala haiti and navassa honduras jamaica martinique mexico montserrat nicaragua panama puerto rico saba saint andr?s and providencia saint bartheleny saint eustatius saint kitts and nevis island saint martin saint vincent and the grenadines sainte lucia the bahamas trinidad and tobago turks and caicos islands virgin islands europe (36 countries) albania andorra austria azores belgium bosnia and herzegovina bulgaria canary islands croatia czech republic denmark estonia faroe islands finland france germany greece iceland ireland italy lithuania luxembourg madeira malta montenegro netherlands norway poland **portugal** romania serbia spain sweden switzerland ukraine united kingdom middle east (16 countries) bahrain cyprus gaza strip iran iraq israel jordan kuwait lebanon oman quatar saudi_arabia syria turkey united arab emirates yemen north america (2 countries) canada united states of america south america (13 countries) argentina brazil chile colombia ecuador falkland islands french guiana guyana peru south georgia and sandwich islands suriname uruguay venezuela*

---

id: 5029 - Data: © wannasurf.com ltd - all right reserved terms of use | copyright | advertising info | contact wanna atlas: wannasurf.com | wannadive.net | wannakitesurf.com | wannask8.com |

==========

Figure B.4: Results.

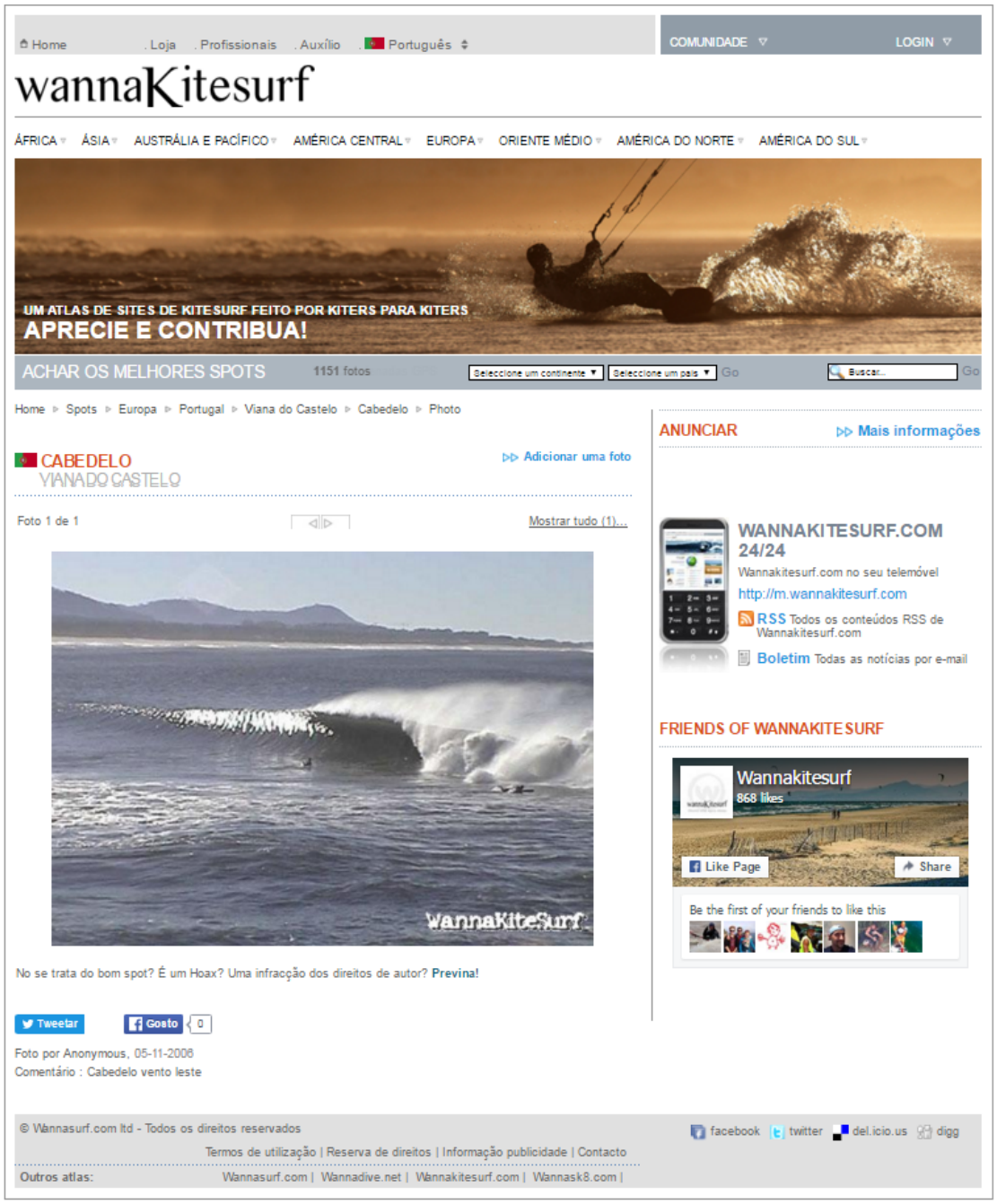

Figure B.5: Correspondent website.

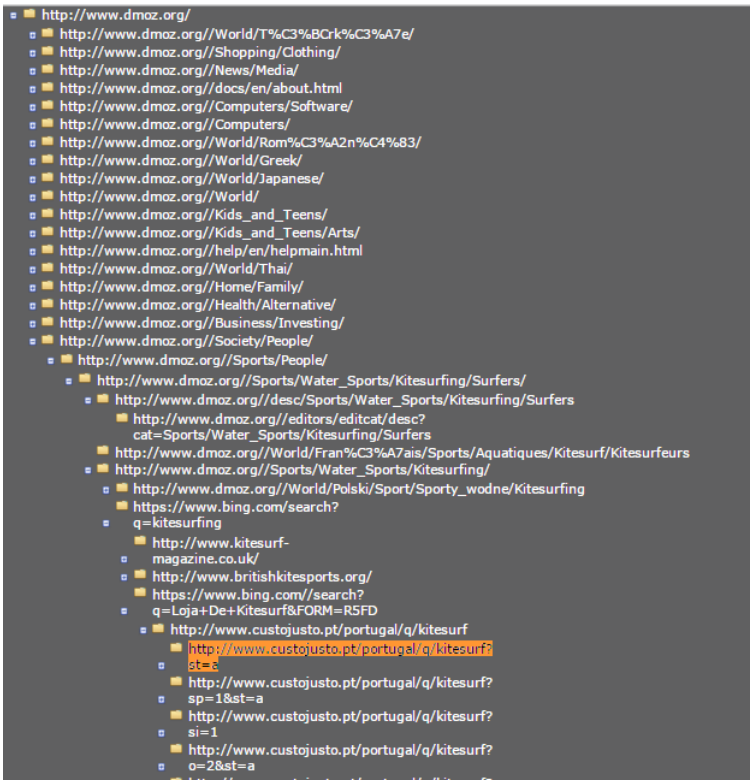

Figure B.6: Treeview.

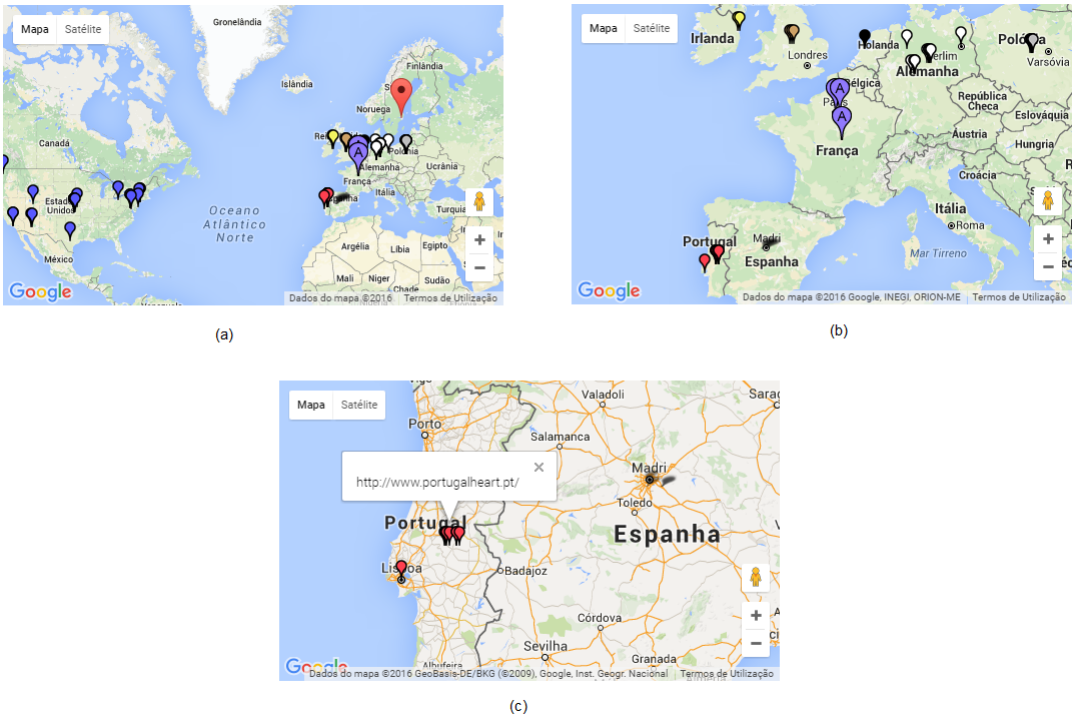

Figure B.7: Geographical representation.

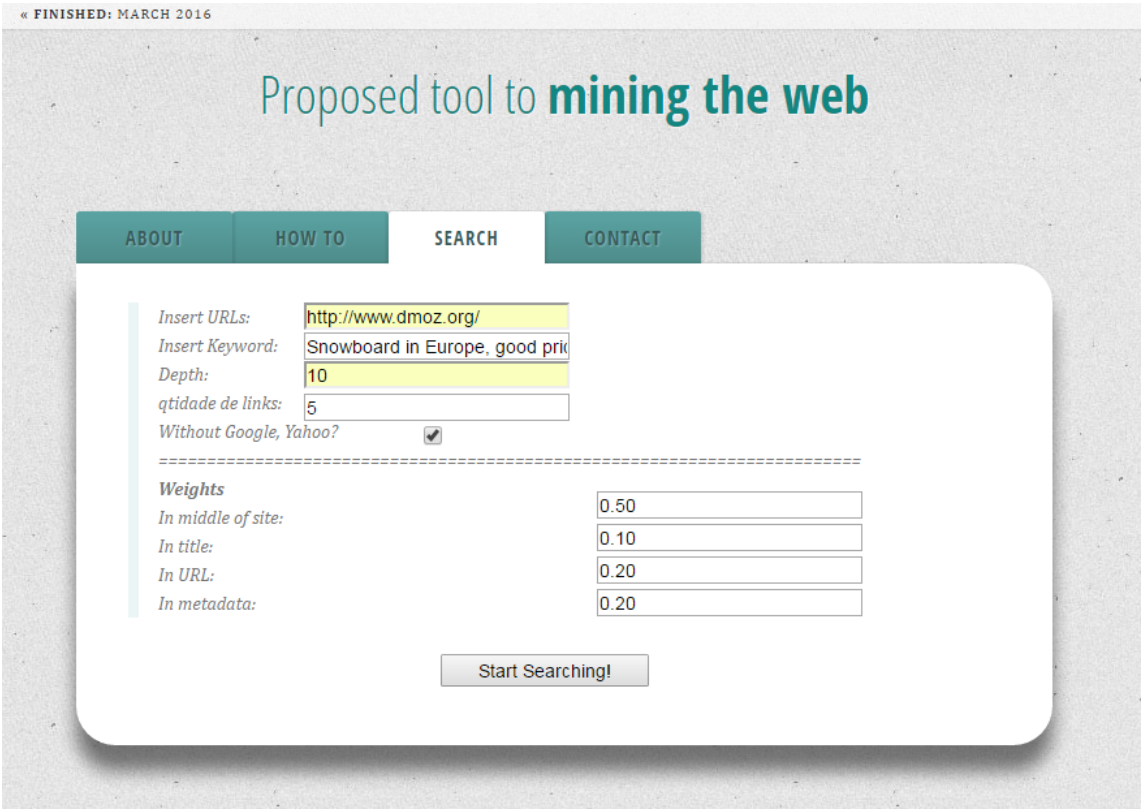

Figure B.8: Main menu with the initial inputs.

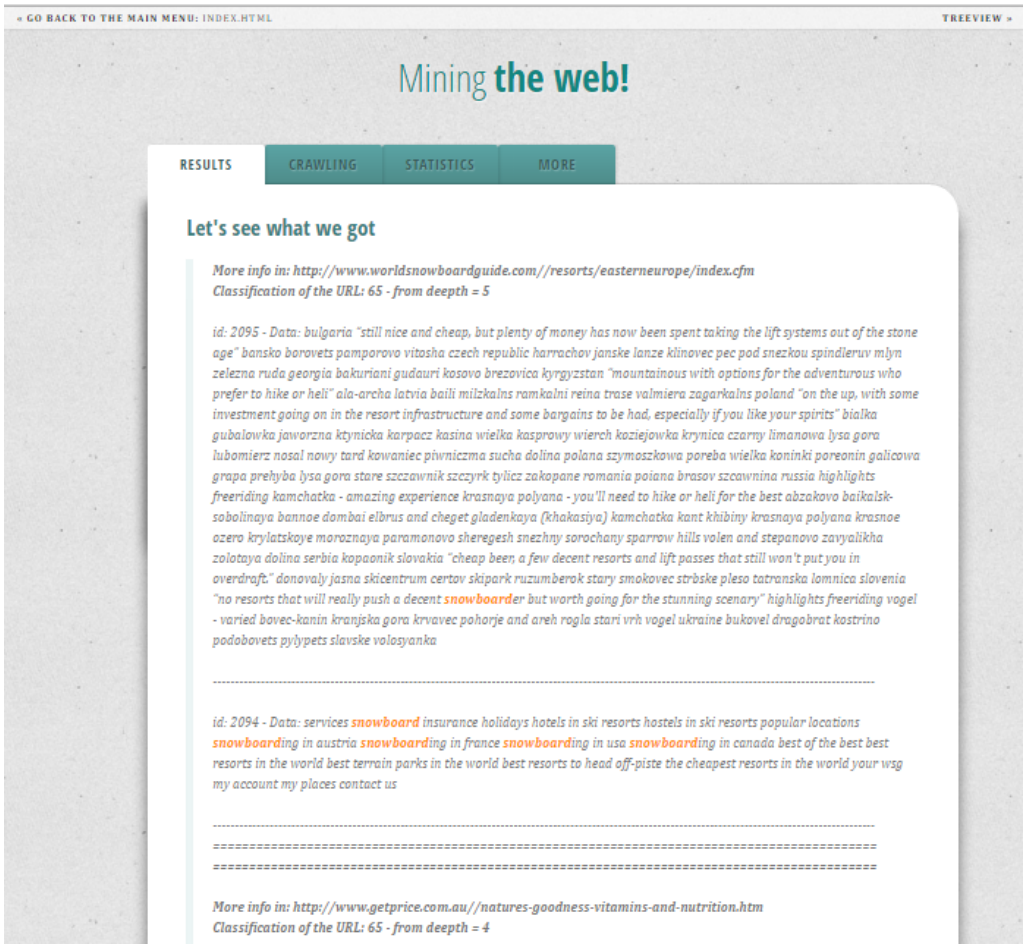

Figure B.9: "Snowboard in Europe, good prices"' Results.

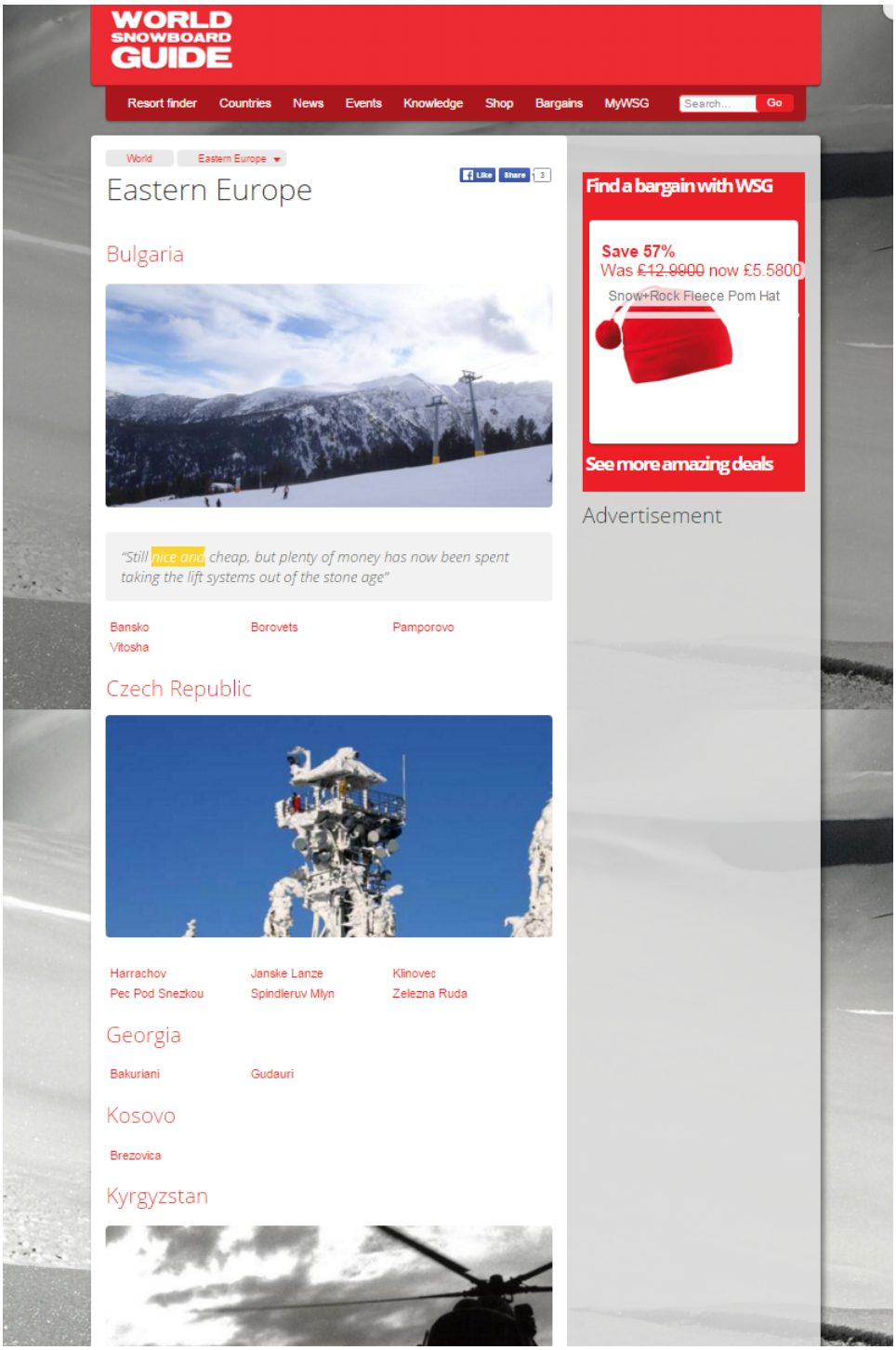

Figure B.10: Correspondent website.

Figure B.11: Treeview.

Figure B.12: Geographical representation.