



Universidade do Minho

Escola de Engenharia

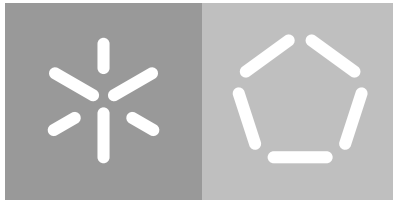
Departamento de Informática

João Tiago Pereira Dias

**Prometheus
A Generic E-commerce Crawler**

**For the Study of Business Markets
And Other E-commerce Problems**

August 2019



Universidade do Minho

Escola de Engenharia

Departamento de Informática

João Tiago Pereira Dias

Prometheus
A Generic E-commerce Crawler

For the Study of Business Markets
And Other E-commerce Problems

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
António Ramires

August 2019

ACKNOWLEDGEMENTS

First of all, I would like to thank Prof. António Ramires, responsible for this dissertations orientation providing insight and help on areas I lacked the required knowledge and pointing me into the right directions. For all the disponibility and time spent in the supervision of the project my most sincere thank you.

This project was developed during an internship at 360Imprimir and so these acknowledgments must be extended to all those who made me feel at home. Special thanks to Filipe Oliveira and Luís Barbosa, my internal supervisors at 360Imprimir, and their team for always being available whenever I needed any kind of help.

A special mention goes to Prof. Alberto Proença for providing me access to the machine where the models were trained and to André Pereira by the support in its use.

I would like to acknowledge my immense gratitude to all my friends for the help throughout this period full of ups and downs.

Finally, I want to thank my parents for all the support, love and incentive without them nothing of this would be possible.

ABSTRACT

The continuous social and economic development has led over time to an increase in consumption, as well as greater demand from the consumer for better and cheaper products. Hence, the selling price of a product assumes a fundamental role in the purchase decision by the consumer. In this context, online stores must carefully analyse and define the best price for each product, based on several factors such as production/acquisition cost, positioning of the product (e.g. anchor product) and the competition companies strategy. The work done by market analysts changed drastically over the last years.

As the number of Web sites increases exponentially, the number of E-commerce web sites also prosperous. Web page classification becomes more important in fields like Web mining and information retrieval. The traditional classifiers are usually hand-crafted and non-adaptive, that makes them inappropriate to use in a broader context. We introduce an ensemble of methods and the posterior study of its results to create a more generic and modular *crawler* and *scraper* for detection and information extraction on E-commerce web pages. The collected information may then be processed and used in the pricing decision. This framework goes by the name Prometheus and has the goal of extracting knowledge from *E-commerce* Web sites.

The process requires crawling an online store and gathering product pages. This implies that given a web page the framework must be able to determine if it is a product page. In order to achieve this we classify the pages in three categories: catalogue, product and "spam". The page classification stage was addressed based on the html text as well as on the visual layout, featuring both traditional methods and Deep Learning approaches.

Once a set of product pages has been identified we proceed to the extraction of the pricing information. This is not a trivial task due to the disparity of approaches to create a web page. Furthermore, most product pages are dynamic in the sense that they are truly a page for a family of related products. For instance, when visiting a shoe store, for a particular model there are probably a number of sizes and colours available. Such a model may be displayed in a single dynamic web page making it necessary for our framework to explore all the relevant combinations. This process is called scraping and is the last stage of the Prometheus framework.

Keywords: E-commerce, Web mining, Web Page Classification, Machine learning, Crawler, Scraper

RESUMO

O contínuo desenvolvimento social e económico tem conduzido ao longo do tempo a um aumento do consumo, assim como a uma maior exigência do consumidor por produtos melhores e mais baratos. Naturalmente, o preço de venda de um produto assume um papel fundamental na decisão de compra por parte de um consumidor. Nesse sentido, as lojas online precisam de analisar e definir qual o melhor preço para cada produto, tendo como base diversos fatores, tais como o custo de produção/venda, posicionamento do produto (e.g. produto âncora) e as próprias estratégias das empresas concorrentes. O trabalho dos analistas de mercado mudou drasticamente nos últimos anos.

O crescimento de sites na Web tem sido exponencial, o número de sites *E-commerce* também tem prosperado. A classificação de páginas da Web torna-se cada vez mais importante, especialmente em campos como mineração de dados na Web e coleta/extração de informações. Os classificadores tradicionais são geralmente feitos manualmente e não adaptativos, o que os torna inadequados num contexto mais amplo. Nós introduzimos um conjunto de métodos e o estudo posterior dos seus resultados para criar um crawler e scraper mais genéricos e modulares para extração de conhecimento em páginas de *E-commerce*. A informação recolhida pode então ser processada e utilizada na tomada de decisão sobre o preço de venda. Esta Framework chama-se Prometheus e tem como intuito extrair conhecimento de Web sites de *E-commerce*.

Este processo necessita realizar a navegação sobre lojas online e armazenar páginas de produto. Isto implica que dado uma página web a framework seja capaz de determinar se é uma página de produto. Para atingir este objetivo nós classificamos as páginas em três categorias: catálogo, produto e spam. A classificação das páginas foi realizada tendo em conta o html e o aspeto visual das páginas, utilizando tanto métodos tradicionais como Deep Learning.

Depois de identificar um conjunto de páginas de produto procedemos à extração de informação sobre o preço. Este processo não é trivial devido à quantidade de abordagens possíveis para criar uma página web. A maioria dos produtos são dinâmicos no sentido em que um produto é na realidade uma família de produtos relacionados. Por exemplo, quando visitamos uma loja online de sapatos, para um modelo em específico existe provavelmente um conjunto de tamanhos e cores disponíveis. Esse modelo pode ser apresentado numa única página dinâmica fazendo com que seja necessário para a nossa Framework explorar estas combinações relevantes. Este processo é chamado de scraping e é o último passo da Framework Prometheus.

CONTENTS

1	INTRODUCTION	1
1.1	Objectives and Motivation	2
1.2	Brief Framework Description	3
1.3	Thesis Outline	3
2	STATE OF THE ART	4
2.1	Web Crawling	4
2.1.1	Web	4
2.1.2	E-commerce	4
2.1.3	Data mining	6
2.1.4	Web mining	6
2.1.5	Solutions in E-commerce world	8
2.1.6	Web Crawler	9
2.1.7	Crawler Detection	15
2.1.8	Robots.txt	16
2.2	Web Page Classification	18
2.2.1	Traditional document comparison	18
2.2.2	Text representation	19
2.2.3	Machine Learning	20
2.2.4	Deep Learning and Types of Neural Networks	21
2.2.5	Ensemble classifiers	22
2.2.6	Subject based classification and genre based classification	22
2.2.7	Visual analysis	25
2.2.8	Link Classification	29
2.2.9	Feature selection	29
2.3	Web scraping	31
2.3.1	Pattern Matching	31
2.3.2	Published Approaches	32
2.3.3	Automation Tests	34
2.4	Summary	35
3	PROBLEM FORMALIZATION AND SOLUTION DEVELOPMENT	36
3.1	Business understanding	36
3.2	Data understanding	40
3.3	Data preparation	46
3.3.1	Textual Context	48

3.3.2	Structural Context	53
3.3.3	Visual Context	55
3.4	Web Page Classification	56
3.4.1	Modeling WPC techniques	57
3.4.2	Building WPC classifier models	57
3.4.3	Designing tests for WPC	60
3.5	Scraper	60
3.6	Contribution review	67
4	CASE STUDIES / EXPERIMENTS	69
4.1	Evaluation of wpc and scraper	69
4.1.1	Textual	71
4.1.2	Structural	73
4.1.3	Visual	74
4.1.4	Full tests report	75
4.2	Deployment	77
4.2.1	Crawler	77
4.2.2	Scraper	79
4.3	Solution overview	81
5	CONCLUSION	82
5.1	Summary and discussion	82
5.2	Future work	83
A	SPAM PAGES EXAMPLES	93
B	NEURAL NETWORK ARCHITECTURE EXAMPLES	94
C	SAME WEB SITES, DIFFERENT WEB PAGES TESTSET ANALYSIS	97

LIST OF FIGURES

Figure 1	Shopping Pipeline modeled as state transition diagram - Source: Srivastava et al. (2005)	5
Figure 2	E-commerce Taxonomy Page Scheme Example	6
Figure 3	Web Mining Tree - Adapted from: Srivastava et al. (2005)	7
Figure 4	Graph example	9
Figure 5	Flow of a basic crawler	10
Figure 6	Standard Crawl	11
Figure 7	Flow of a focused crawler	13
Figure 8	Focused Crawl by link	14
Figure 9	Focused Crawl Flow by Web page	14
Figure 10	Summary of daily Web traffic at the University of Minnesota computer science department Web server. The first picture represents a comparison between all HTML requests and the ones caused by robots. The anomaly in the right-hand figure (day 30) is due to HTML requests caused by a mapping robot called <i>linbot</i> . Tan and Kumar (2002)	15
Figure 11	Example of a robots.txt, from "www.nike.com"	17
Figure 12	RNN architecture	21
Figure 13	Interface suggested, red square show an approach in our subject; Adapted: Roussinov et al. (2001)	24
Figure 14	Datafiniti predict page type methodology - Source: Dat (accessed January 16, 2019)	25
Figure 15	Genre examples in E-commerce	26
Figure 16	Aesthetic classification - Source: de Boer et al. (2010)	26
Figure 17	Recency classification - Source: Videira (2013)	27
Figure 18	Confidence of the network for seven Websites on seven genres - Source: Doosti et al. (2017)	28
Figure 19	URL structure example	29
Figure 20	Pages Family Tree	30
Figure 21	Different price in the same product page - color change	34
Figure 22	Trinity for crawler functionality	38
Figure 23	Types of pages resulting of the search for "shoes" keyword - Google	41

Figure 24	Types of pages resulting of the search for "shoes" keyword - MySimon	41
Figure 25	Flow Chart Crawl	43
Figure 26	Genre examples in E-commerce	45
Figure 27	Visual table	45
Figure 28	Html representation of Figure 27	46
Figure 29	Restrictions over the Web Mining Tree	46
Figure 30	Example of robots.txt with bad Web crawlers	47
Figure 31	Span tags - E-commerce web page examples \$ and £	48
Figure 32	Full-size driver print of the page.	55
Figure 33	Other image sizes	56
Figure 34	Load training set directly from the web (method 1) - time	61
Figure 35	Load training set directly from the web (method 2) - time	61
Figure 36	Flow Chart Scraper	62
Figure 37	Dropdown example	63
Figure 38	Buttons example	63
Figure 39	Input box example	64
Figure 40	Options removal	65
Figure 41	Dormant options	65
Figure 42	Non input option	66
Figure 43	Marketing product E-commerce Web page	66
Figure 44	Non regular options	67
Figure 45	Check option selection	67
Figure 46	Same Web sites, different Web pages analysis; P - Product; C - Catalog; S - Spam	71
Figure 47	Misclassified examples - Structural approach	73
Figure 48	Misclassified examples - Visual approach	74
Figure 49	Different design tests	75
Figure 50	Tests in new Web sites in the same product domain (clothing) and in a different product domain (electronics - lamps)	76
Figure 51	Crawl blocked - message I	77
Figure 52	Blocked crawl examples	78
Figure 53	Uprinting block	78
Figure 54	Extra attributes from Figure 43 Web page - Non default	79
Figure 55	Product page price representation	80
Figure 56	Product scrap results	80
Figure 57	Spam page examples.	93
Figure 58	Neural Network for textual evaluation	94

Figure 59	Neural Network for structural evaluation	95
Figure 60	Neural Network for visual evaluation	96

LIST OF TABLES

Table 1	Different word count found in preprocessing stage for the entire training set	52
Table 2	Chosen HTML tags	54
Table 3	Accuracy results with static data-sets (best values in green).	70
Table 4	Textual (RNN) model assessment - Test 1	97
Table 5	Structural (dense layers) model assessment - Test 1	97
Table 6	Visual CNN model assessment - Test 1	97

GLOSSARY

AI Artificial Intelligence. 2, 19

ANN Artificial Neural Networks. 20, 22, 24, 35, 57, 82

BF Breadth-First Traversal. 11

BI Business Intelligence. 1, 9, 37

BoW Bag of Words. 19, 56, 69, 71, 82

CNN Convolutional Neural Networks. ix, 21, 27, 56, 57, 59, 69, 70, 97

CREAD Charter for Responsible and Ethical Acquisition of Data. 17, 47, 77, 78, 81

CRISP-DM Cross Industry Standard Process for Data Mining. 36, 69

CRM Customer Relationship Management. 6, 36

CSS Cascading Style Sheets. 30, 33

DF Depth-First Traversal. 11

DM Data mining. 6

DNS Domain Name System. 16

DOM Document Object Model. 33, 64

ERP Enterprise Resource Planning. 6, 36

FAQ Frequently Asked Questions. 18, 44

GloVe Global Vectors for Word Representation. 22

GUI Graphic User Interface. 34

HTML HyperText Markup Language. vi, 7, 15, 18, 25, 27, 28, 30, 31, 32, 33, 45, 53, 60, 63, 64, 67, 68, 82

HTTP Hypertext Transfer Protocol. 29, 60

IDF Inverse Document Frequency. 19, 72

IP Internet Protocol. 15, 16, 79

IR Information Retrieval. 18, 19, 23, 49

JSON JavaScript Object Notation. 33

LSI Latent Semantic Indexing. 8, 19, 56, 57, 69, 72, 82

LSTM Long short-term memory. 57, 58

ML Machine Learning. 8, 57, 71, 73

NB Naive Bayes. 8, 20, 56, 57, 69, 72, 73, 82

NP Non-deterministic Polynomial time. 11

POP Point of Purchase. 6, 36

POS Point of Sale. 6, 36

ReLU Rectified Linear Unit. 59

REP Robots Exclusion Protocol. 16

RNN Recursive Neural Networks. ix, 21, 56, 57, 69, 70, 72, 73, 76, 77, 81, 82, 97

RPA Robot Process Automation. 2

SERP Search Engine Result Pages. 41

SVM Support Vector Machines. 8, 20, 56, 57, 69, 73, 82

SWT Structure-oriented Weighting Technique . 30

TF Term Frequency. 19, 72

TF-IDF Term Frequency Inverse Document Frequency. 8, 19, 56, 57, 69, 72, 82

URL Uniform Resource Locator. 10, 11, 12, 14, 16, 24, 27, 28, 29, 42, 48, 49, 51, 52, 69, 70, 71, 72, 82

UX User Experience. 23

VPN Virtual Private Network. 79

VSM Vector Space Model. 12, 19

WPC Web Page Classification. 18, 20, 22, 23, 27, 29, 30, 35, 45, 53, 56, 68, 72, 73, 81, 82

WWW The World Wide Web. 4, 18

XPath XML Path Language. 33

INTRODUCTION

“Sell me this pen” - The Wolf of Wall Street [Belfort \(2007\)](#)

Bettman and Park in 1980 postulated that the two main aspects in consumer choice environment were product information and prior knowledge [Bettman and Park \(1980\)](#). It is obvious that in the web environment product information is not only already available but also enormous, hence, business market needs to keep tabs on the information flow between the competition and the consumers and make its proposal as appealing as possible to the public.

Many companies such as credit, insurance, online shops, and banking companies require a constant connection to the market to help in setting the institution’s marketing goals. Data mining techniques are efficient when the commercial monopoly is divided between different entities. Knowledge about the market universe has an obvious impact on the relationship with possible competitors, as well as with the public target.

360Imprimir, where this thesis is being developed, is a company that was born in 2013, in Portugal. They seek differentiation through excellence and customer satisfaction, their mission is to help and inspire small and medium Enterprises to have successful communication, changing the way they develop and implement their marketing strategy. This is possible through an E-commerce platform where all type of merchandise is sold.

360Imprimir share the same problems and needs of other companies, thus the continuous supervision of dynamic market trends is important. Traditional methods of monitoring market direction (e.g. monthly or quarterly updates, analyst reports) don’t fulfill the need to provide data for emerging business models.

Companies need to be connected and access key market events if and when they happen. As markets and consumers become more segmented, a comprehensive approach to competitive intelligence is critical to a functional Business Intelligence process. By understanding competitors’ pricing, consumer sentiment, and even inventory levels, the appropriate range of responses and diligence can be considered and taken within the business. In that sense, by incorporating web data into their critical information flows, businesses can now have more robust [BI](#) systems and better data to make informed management decisions.

In short, there is a need for the extraction of knowledge for later decision making within the business area.

1.1 OBJECTIVES AND MOTIVATION

The main objective of this thesis is the study and creation of a solution that may help in the extraction of information about concurrency products. Posteriorly, this information can be used to feed a rule engine for prices optimization or simply be used as information given to the BI team of a company for decision making.

Nowadays, this gathering of information is commonly a manual process that results in a never-ending work for a group of people that must obtain their possible competitors' information (e.g., tv, mouth-to-mouth, Internet, adds). After the acquisition of the "shop on the Internet" of a determined competitor, the second level of information acquisition is also manually started. There is a need for learning what is of interest in that domain, the reader may imagine this process as the drawing of a map of that web site, and in this map the extraction of the locations that have useful information. After all this, the real knowledge extraction is started.

Most of the web pages that present product information present also possibilities to personalize that product. The addition or dismissal of those features in a product may increase or decrease the price accordingly. So at this level, there are being implemented two types of approaches. First, the manual selection of the features and the posterior extraction of the interest information. The second approach is the usage of a Robot Process Automation (RPA), where employees configure advanced software or a "robot" to perform routine processes over the manual selected features. The big benefit for companies is that the tasks are executed more efficiently and with a smaller margin of error, allowing them to reduce labor costs. The downside of this approach is that web pages are highly mutable and a change may disrupt the routine that the robot follows, so high maintenance will be needed.

The main goal, in a macro perspective, is to shift the burden of browsing through web pages, and the market information gathering, away from the user, in an automated process, that will be translated in a reduction of man-hours spent in market studying, and a general growth in the efficiency and quality of the data extracted.

Data extraction can be performed with traditional approaches, or resorting to Deep Learning techniques. These later techniques can be used for page classification, an essential part of the information extraction process. Through this approach, by comparing to traditional methods, the goal is to evaluate which has better efficiency and accuracy. From an academic standpoint, this subject is already being discussed, in Brooks (2008) they explored Web Crawling as a AI Project.

With this thesis, we also hope to show that the academic and the business worlds can walk hand in hand and not as two separate entities. Promoting the idea that the usage of academic findings and techniques should be applied in a real-world scenario is a win-win

situation, paving a better future and better findings and revealing that this symbiosis more than possible is needed.

In a final look, we hope this approach will improve the competitive ability and earning profit for each online company that has access to it finding gaps in their competitors' offering or even finding where new markets are emerging.

1.2 BRIEF FRAMEWORK DESCRIPTION

The proposed solution consists in developing and perfecting of a two-stage solution: a crawler and a scraper. The crawler will act over the E-commerce web sites, finding them if necessary and navigating through Web pages guided by a page classification system that will divide the crawled pages into 3 types of web pages: Spam pages, Catalog pages, and Product pages.

For the second stage, the scraper will use the classified Web pages by the crawler in the Product class. From those pages, the scraper will gather information about the possible attributes and values from the product (e.g., size, quantity, color), combine them, and change those specifications in the page extracting the price information for posterior business analysis.

1.3 THESIS OUTLINE

The remaining of this document is organized as follows: in Chapter 2 a theoretical background is provided on the areas of performance-based Web crawling, Web page classification problem and Web scraping. Chapter 3 then formalizes the competition analysis and supervision problem. It describes the available data and development of a crawler and scraper for the issue, covering the first four phases of the CRISP-DM methodology [Chapman et al. \(2000\)](#) business understanding, data understanding, data preparation, as well as the design of the Web page classifier (i.e., the "brain" that will guide the crawler) and the scraper. Following the same methodology, Chapter 4 comprises the remaining phases: evaluation and deployment of the crawler and scraper in a real case scenario. Finally, Chapter 5 summarizes the work done with a few final remarks and concludes with some proposals of possible future work.

STATE OF THE ART

This chapter will provide the relevant theoretical background in three main areas. First, Section 2.1 will briefly review the topic of performance-based crawling over the *Web*. Section 2.2 will then introduce the field of classification methodologies for classifying Web pages, detailing some of the most popular techniques and relevant concepts. Lastly, Section 2.3 will describe how to collect relevant information from a Web page.

2.1 WEB CRAWLING

2.1.1 *Web*

The Web was invented in 1989 by Tim Berners-Lee. He coined the term World Wide Web, wrote the first Web server, `httpd`, and the first client program (a browser and editor).

The World Wide Web ([WWW](#)) serves as a huge, widely distributed, global information center for news, advertisements, consumer information, financial management, education, government, and E-commerce. With the explosive growth of the information sources available on the [WWW](#), it has become extremely important the use of automated tools to find the desired information/resources.

According to the statistics data available on Internet Live Stats Online, by the end of May 2018, there were more than 1.8 billion Websites [Int \(accessed March 22, 2019a\)](#). By the end of 2016, the number of Internet users was bigger than 3.3 billion [Int \(accessed March 22, 2019b\)](#). Taking these numbers into consideration, it is safe to say that Web pages affect peoples lives and the way they act through that life.

2.1.2 *E-commerce*

The methods and means for transactions in commerce are steadily evolving in close step with technological advances. For instance, the traditional ways of selling products and services, known as "*brick and mortar*" approach, have expanded into online sales, Electronic commerce (*E-commerce*).

An E-commerce application presents a Web site that creates a virtual, user-readable, “storefront”. The storefront is nothing more than a series of downloadable Web pages, structured hierarchically with embedded hyperlinks, connecting to other related Web pages and content. The organization of the Web site is essentially a catalog of goods and services and includes resources for secure purchasing.

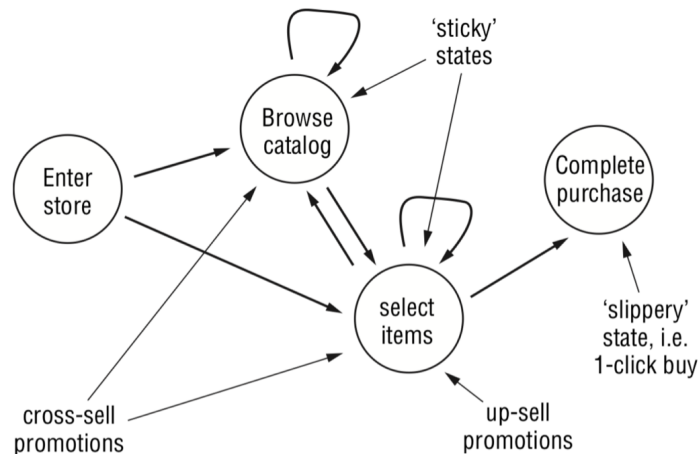


Figure 1.: Shopping Pipeline modeled as state transition diagram - Source: [Srivastava et al. \(2005\)](#)

From the user standpoint, he is being induced to navigate an *E-commerce* Web site in a way that maximizes the number of purchases, as shown in Figure 1. A good product placement (navigability: **minimal path** and **shortcut** facility) will affect the effectiveness of selling a product to a customer. Therefore, it has a huge impact on the success of the business, as stated by [Rababah and Masoud \(2010\)](#).

Even though the selection of items can be achieved through multiple paths, browsing a catalog (list of products) is always possible in that selection, an example of such is represented in Figure 2.

However, the problems of maintaining this kind of platform share the same issues of a traditional shop, being the success/unsucces of the business directly connected to its competitive market place, hence, the study of the data presented in the corporate world is a necessity.

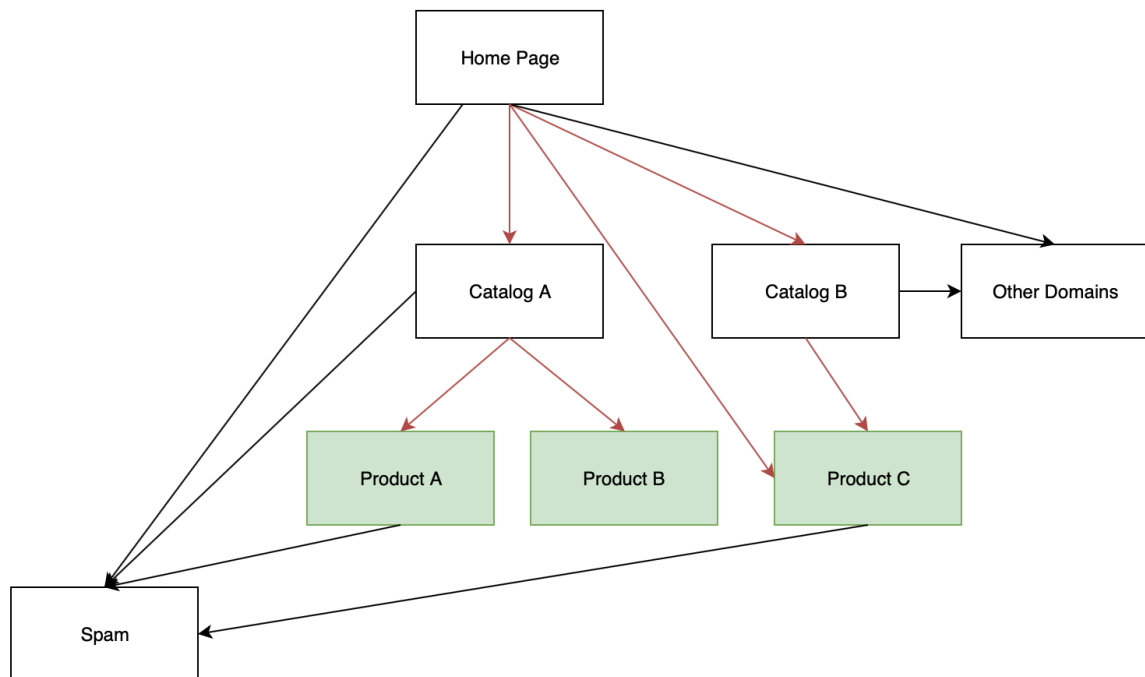


Figure 2.: E-commerce Taxonomy Page Scheme Example

2.1.3 Data mining

Data mining (DM) refers to the extraction of information from a large amount of data [Kernighan and Ritchie \(2001\)](#). In a business perspective, this data can be used to find relevant information, and after analyzing it knowledge may be achieved to support the decisions of business professionals [Kleissner \(1998a\)](#).

A company most valuable source of data does not reside in a CRM software, POS or POP systems, or even in ERP platforms. The biggest untapped source of data is the Web itself, where a monolith of information that connects all the parties of the *E-commerce* process is present.

2.1.4 Web mining

The term Web mining was coined by Etzioni (1996) to denote the use of data mining techniques as a "*process-centric view*", which defined Web mining as a sequence of tasks: automatically discover Web documents and services, extract information from Web resources, and uncover general patterns on the Web. This concept has been discussed in [Madria S.K. and E.P. \(1999\)](#) [Kosala and Blockeel \(2000a\)](#). In [Cooley et al. \(1997\)](#) data mining in a Web

context is seen as a *data-centric view* process. Web mining is a term that focuses more on Web content, Web structure, and Web usage data as can be seen in Figure 3.

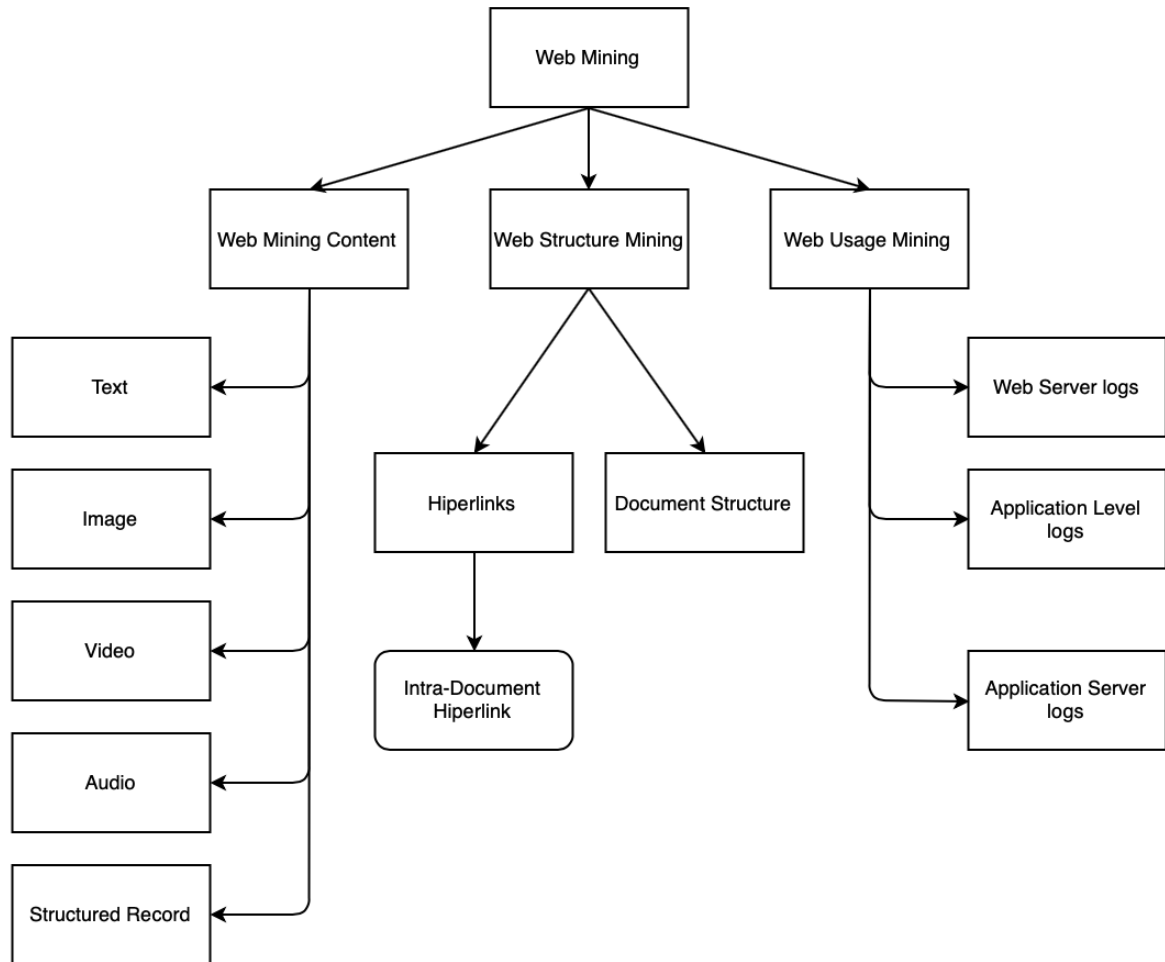


Figure 3.: Web Mining Tree - Adapted from: [Srivastava et al. \(2005\)](#)

- **Web content mining** has been studied extensively by researchers, search engines, and other Web service companies. It is used to examine the content of Web pages as well as results of Web searching. The content may include text, graphical or audio data.
- **Web structure mining** is the process of using graph and network mining theory and methods to analyze the nodes and connection structures on the Web. It extracts patterns from hyperlinks, where a hyperlink is a structural component that redirects a Web page to another location. It may also mine the document structure within a page (e.g., analyze the treelike structure of page structures to describe [HTML](#) and [XHTML](#) tag usage), the posterior study of this components can help understand the Web contents.

- **Web usage mining** is the process of extracting valuable information from server logs (e.g., user click-streams). It is mostly used for finding navigation patterns. Being the potential disclosure of this data a concern on the privacy of personal information.

In Liu (2011) and in Xu et al. (2011) a discussion is presented regarding several basic topics of Web document representation, Web search, short text processing, topic extraction, and Web opinion mining. In these articles, we can see reports of techniques (e.g. TF-IDF, LSI) for the document representation, domain reduction and recurrent problems about Web mining. They even introduced the idea of ML methods, such as NB and SVM, for a classification set of relevant and irrelevant documents. In Kosala and Blockeel (2000b) a survey showing different ML techniques applied to Web Mining is presented.

In Kleissner (1998b) Data Mining for the Enterprise is approached, presenting the issues in creating a tool to collect information about the competition, partial paths for solutions, and the motives for needing those solutions.

For *E-commerce*, the aggregation of the data from different *E-shops* is one of the main issues. An E-commerce Web site usually suffices as a showcase for the products of one specific company, facilitating the comparison of products of that company/E-commerce Web site, however, there is **not** a cluster where all data is gathered from **multiple** E-commerce Web sites, being this information scattered all over the Web.

2.1.5 Solutions in E-commerce world

The clustering of disseminated information all over the Web in multiple *E-commerce* Web sites creates a business opportunity.

Web sites like Momondo Mom (accessed June 20, 2019) and Trivago Tri (accessed June 20, 2019), inserted in the traveling category, are solutions that gather information from multiple Web sites and present them to the user, allowing them to choose from a vast range of options of the same "product".

An example of a mobile app that integrated the *E-commerce* world is SnapTell. Acquired by Amazon, it was created to improve the shopping experience Lif (accessed December 21, 2018), but was also recently dropped. It mainly used products like CDs/DVDs, books or games where users could upload an image of the cover of those products to find their prices and reviews from Amazon inventory.

Some applications still prevail in the market, like CamFind and Slyce which provide price based comparison of different products across different but specific *E-commerce* Web sites Clo (accessed December 20, 2018) Sly (accessed December 21, 2018).

For physical products, there are price comparison based search engines (e.g. Google Shopping Engine Goo (accessed December 23, 2018), Kuantokusta Pimenta (accessed December 23, 2018)). These platforms also work as a showcase for a company's products,

however, the data presented is only a small percentage of the full data presented around the global market.

For the niches in which those solutions weren't enough, more sophisticated and oriented solutions were created and companies like [Dat](#) (accessed June 20, 2019), [Imp](#) (accessed June 20, 2019), and [Scr](#) (accessed June 20, 2019) emerged. These companies produce solutions for market study in the Web for other companies, this symbiosis has been prospering with time. However, the presented solutions make the company (client) have to integrate the values provided by them in their BI systems (i.e., adaptability by the client to the solution). Another negative aspect is that all these solutions only present the default price values of the web page (i.e, do not introduce dynamic values from different specifications of products) ignoring valuable information for decision making.

2.1.6 Web Crawler

A Web crawler is an automated program that scans through the Web's graph structure jumping from page to page as illustrated in Figure 4. Also known as wanderers, ants, automatic indexers, bots, robots, spiders, fish, and worms, words that are quite evocative of Web imagery.

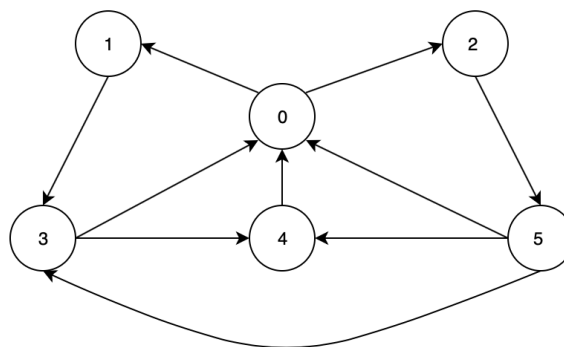


Figure 4.: Graph example

The first generation of Web crawlers was based heavily on traditional graph algorithms, such as **breadth-first** or **depth-first traversal**, to index the Web. A crawler's input is a seed *URLs*. Henceforth, all new and not repeated *URLs* will be indexed to the frontier (i.e., a queue where *URLs* are inserted) for every crawled Web page, until this frontier does not have any more uncrawled *URLs*. This process is described in Figure 5.

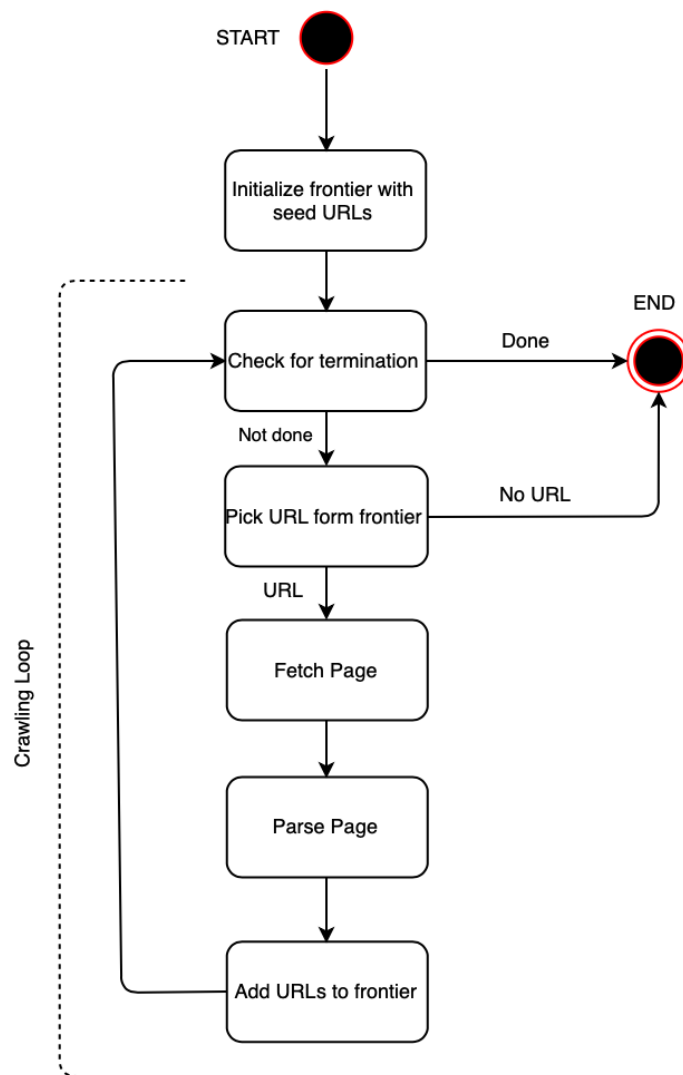


Figure 5.: Flow of a basic crawler

With the exponential growth of the Web as stated in Section 2.1.1, fetching information about a special-topic gained importance.

Having in consideration the possible complexity of the problem, this kind of "blind" approaches such as BF or DF will not suffice. Considering a core set of URLs used as a seed set, the algorithm would recursively follow hyperlinks down to other documents.

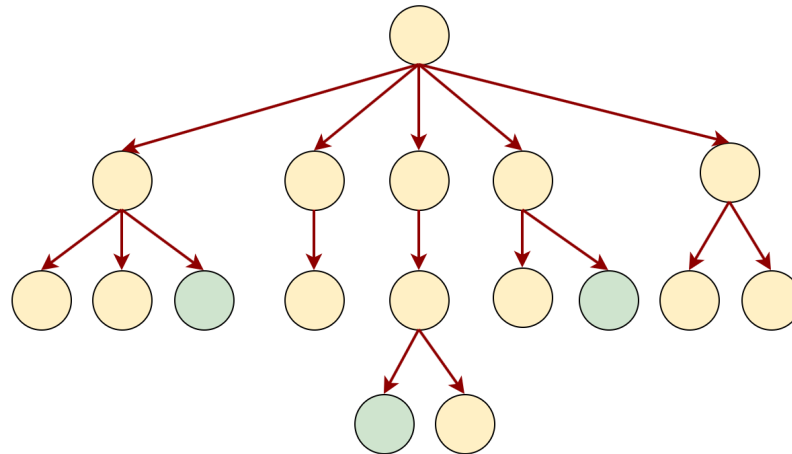


Figure 6.: Standard Crawl

As illustrated in Figure 6 in a simple crawl where 3 elements are relevant (in **green**) the crawler would parse all the other pages (in **yellow**). This approach to the discovery of a specific result will not work in a real-world scenario since the crossing between domains will translate in a NP problem. Hence, the need for more target-based algorithms, and this necessity opened doors to the creation of multiple approaches and studies of these solutions such as [Udapure et al. \(2014\)](#), [Chakrabarti et al. \(2014\)](#), [kumar et al. \(2016\)](#), [Menczer \(2004\)](#) or even [Nigam \(2014\)](#), that aim to compare or even evolve the current solutions to this problem.

In a general sense, a crawler (particularly a topical crawler) may be evaluated on its ability to retrieve relevant pages. However, a major hurdle is the problem of recognizing the relevant pages. In an operational environment, real users may judge the relevance of pages as these are crawled allowing us to determine if the crawl was successful or not. For the crawl itself, some algorithms have been extensively studied, many of these are variations of the **best-first** scheme. The difference is in the heuristics used to score the visited or unvisited URLs with some algorithms adapting and tuning their parameters before or during the crawl. Some relevant examples are:

- **Best-First Search Algorithm** [Chakrabarti and Ghose \(1992\)](#): As the name suggests, nodes are visited one at a time according to a score attributed by a previous classifica-

tion system, typically an estimate of the cost of a solution passing through that node, top scores having priority.

- **Fish-Search Algorithm:** An example of early crawlers that prioritizes unvisited URLs on a queue for a specific search goal. The Fish-Search approach assigns priority values (1 or 0) to candidate pages using simple keyword matching. One of the disadvantages of Fish-Search that there is no variance in the priority of all relevant pages.
- **Shark Search Algorithm** [M. et al. \(1998\)](#): A modified version of Fish-Search. The difference resides in the use of a Vector Space Model (VSM), and the priority values (more embracing than just 1 and 0) are computed based on the priority values of **parent pages, page content, and anchor text**.
- **Similarity Search Algorithm:** This approach proposes to use the full content of the pages already visited to infer the similarity between the driving query and the pages that have not been visited yet.
- **Learning Anchor Algorithm:** The predictor uses the anchor text of links to define its relevance. These strategies are very effective for short crawls, while more sophisticated techniques are known to perform best over longer crawls.
- **PageRank** [Benincasa et al. \(2006\)](#): This approach measures the importance of a Web page counting the quantity and quality of the links that point to it.

Focused crawling is one of those solutions. The Focused crawling was first introduced in 1999 [Chakrabarti et al. \(2000\)](#). A focused crawler or topical crawler is a Web crawler that attempts to download only Web pages that are relevant to a predefined topic or set of topics as can be seen in Figure 7.

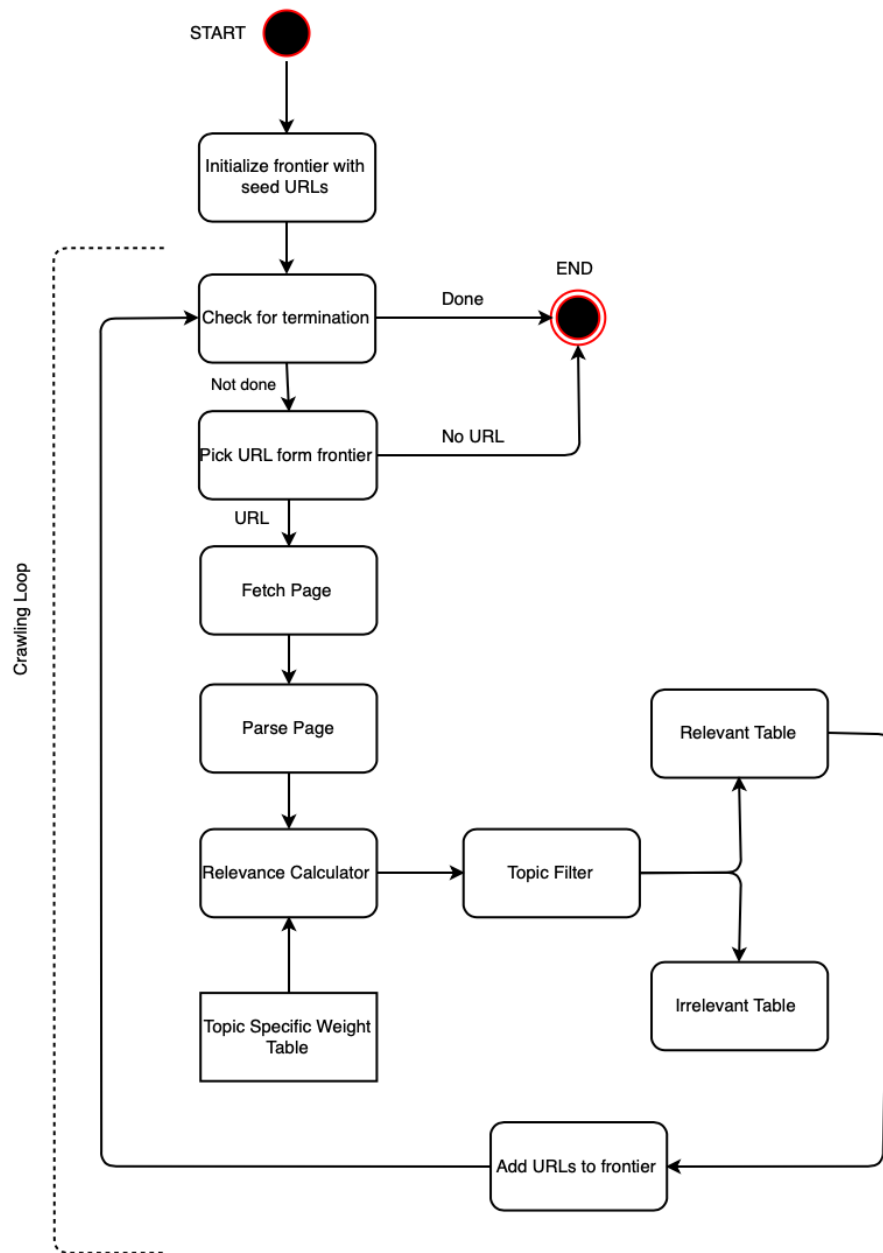


Figure 7.: Flow of a focused crawler

Ideally, it would only download pages that are relevant to a particular topic and avoid downloading all others. Therefore, a focused crawler **must accurately predict if a link to a particular page is relevant before actually downloading the page** as shown in Figure 8.

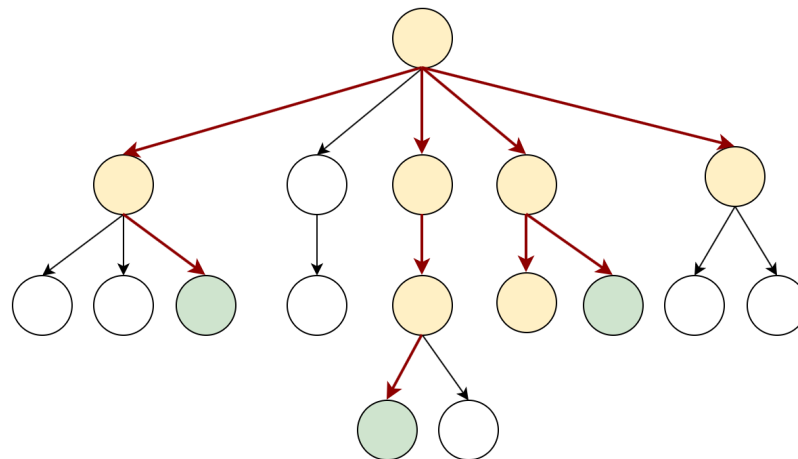


Figure 8.: Focused Crawl by link

In another approach, **the relevance of a page is determined after downloading its content**. Relevant pages are sent to content indexing and their contained [URLs](#) are added to the crawl frontier; pages that fall below a relevance threshold are discarded (in **red**) as seen in Figure 9. The overhead of the document content is paid little heed in this case, since the ultimate goal of the crawl is not to cover the whole Web but to have a better classification for a micro-universe of Web pages.

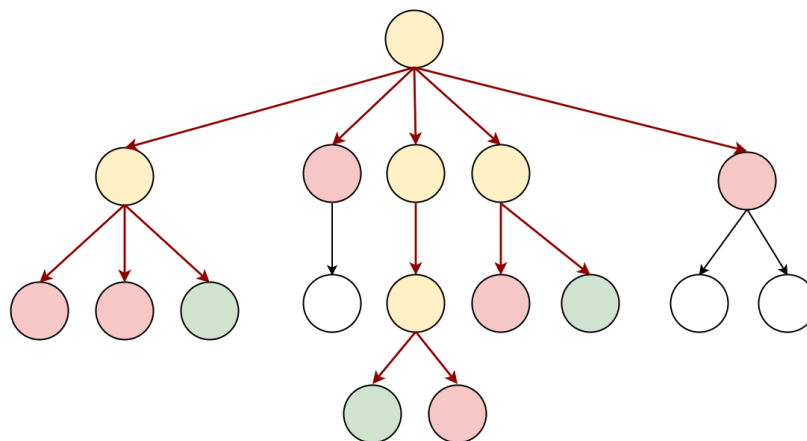


Figure 9.: Focused Crawl Flow by Web page

The final goal is to narrow the number of parsed/downloaded pages, hence impacting *performance/usability*. A direct comparison between Figure 6, Figure 8 and Figure 9 shows the difference between all approaches.

The performance of a focused crawler and the search mechanism it uses depends mostly on the richness of links in the specific topic being searched and it usually relies on a general Web search engine for providing starting points.

2.1.7 Crawler Detection

As said before, crawlers are software programs that navigate the hyperlink structure of the Web to locate and retrieve information. The importance of separating robot behavior from human behavior before building user behavior models is not a new problem [Kohavi and Provost \(2001\)](#).

To the *E-shoppers* (*E-commerce* retailers), the usage of crawlers may also represent a problem because the presence of automated crawlers makes it difficult to perform click-stream analysis effectively on the Web data and tend to consume considerable network bandwidth. Even more relevant, the unauthorized deployment of crawlers for gathering business intelligence at their Web sites by other business competitors.

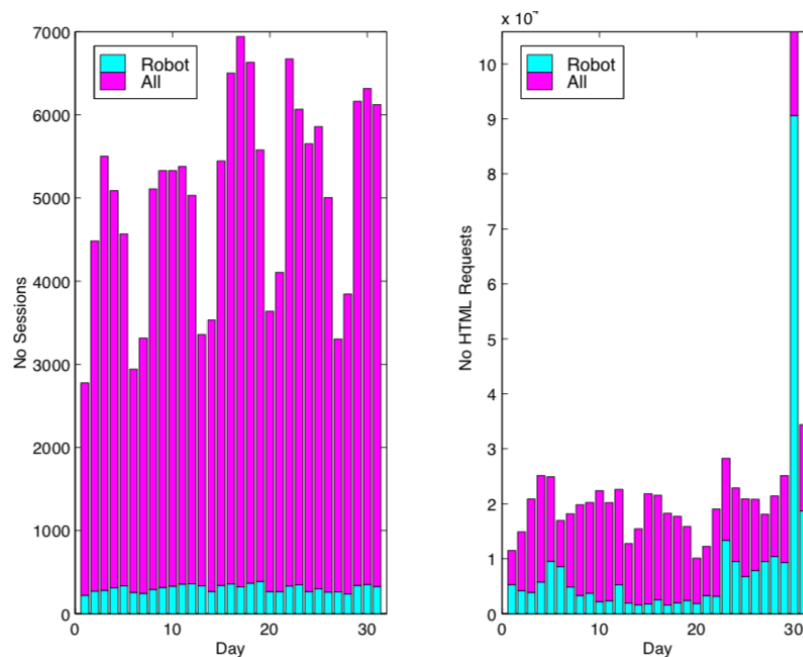


Figure 10.: Summary of daily Web traffic at the University of Minnesota computer science department Web server. The first picture represents a comparison between all HTML requests and the ones caused by robots. The anomaly in the right-hand figure (day 30) is due to HTML requests caused by a mapping robot called *linbot*. [Tan and Kumar \(2002\)](#)

To address this issue there are some conventional techniques for detecting crawlers based on identifying the IP address and user agent of the Web clients. These techniques apply

to many well-known robots, however, they are not sufficient to detect camouflaged and previously unknown robots. To fight this problem there are already studies on the subject. Some examples are:

- [Tan and Kumar \(2002\)](#) proposed a classification based approach that uses the navigational patterns in click-stream data to determine if it is due to a robot. Experimental results have shown that highly accurate classification models can be built using this approach. Furthermore, these models can discover many camouflaged and previously unidentified robots.
- [Huntington et al. \(2008\)](#) set out to verify robots in the scholarly information environment via identity and behavior methods. Robot identities were researched by checking what IP numbers had visited the Robots.txt document, looking for robot names in the DNS name, selecting IP numbers based on existing lists of undeclared robots and finally checking browser details for robot identities. Behavioral methods looked specifically for a metric footprint of robots. They hypothesized that robots would be likely to generate a high daily usage count and that usage will be patterned over 24 hours accessing content at a rapid rate.

Ethical questions about the use of crawlers may appear, being a fairly sensitive topic, considering that the misuse of crawlers may put in cause the integrability of a Web site and its functionality (Figure 10) and the legal repercussions that may have in the business context. To prevent the weaponization of this technology some good metrics have been created to guide in the construction of a crawler with “*good behavior*”.

2.1.8 Robots.txt

Robots.txt is a text file webmasters created to instruct Web robots (typically search engine robots) how to crawl pages on their Website. The robots.txt file is part of the robots exclusion protocol (REP) [Koster \(1994\)](#), a group of Web standards that regulate how robots crawl the Web, access and index content, and serve that content up to users, an example is presented in Figure 11.

In some cases there is also a **Sitemap**, here webmasters inform search engines about pages on their sites that are available for crawling. In its simplest form, a Sitemap is a XML file that lists URLs for a site along with additional *metadata* about each URL (e.g., when it was last updated, how often it usually changes, and how important it is comparative to other URLs in the site) so that search engines can more effectively crawl the site. However, as **robots.txt** this file is optional and the information in it is all under the webmaster control, being the omission of information a common trait.

```
# https://www.nike.com robots.txt -- just crawl it.

User-agent: *
Disallow: /us/en_cs/
Disallow: /il/en_il/
Disallow: /us/en_eu/
Disallow: */member/inbox
Disallow: */n/
Disallow: */p/
Disallow: */checkout/
Disallow: */w?q=
Disallow: /*.swf$
Disallow: /*.pdf$
Disallow: /ar/help/
Disallow: /br/help/
Disallow: /hk/help/
Disallow: /kr/help/
Disallow: /ph/help/
Disallow: /uy/help/
Disallow: /xf/help/
Disallow: /xl/help/
Disallow: /xm/help/
Disallow: */retail*/sitemap***
Disallow: */retail*/frontend/*
```

Figure 11.: Example of a robots.txt, from "www.nike.com"

Despite being an agreed-upon as good heuristic many crawlers do not comply with the [REP](#) since this standard is voluntary and there is no real backlash for not following these rules.

Dat (accessed June 20, 2019) along with several partners and customers has embarked on an initiative called the *Charter for Responsible and Ethical Acquisition of Data (CREAD)*. Every customer or partner they interact with has welcomed their initiative. Simple and straightforward to implement, they have 5 principles:

- **Self-Identify the Web agent** that is acquiring the data;
- **Obey** the robots file;
- A self-regulated **rate limiter**;
- **Restrict** data gathering to **public data** only;
- Facilitate **open access** to Web data;

By adhering to these principles in spirit and practice, customers and partners can be assured of avoiding the negative consequences of unethical data acquisition. More importantly, they send a strong signal to their customers and the ecosystem they serve on where they stand on data, transparency, and trust.

2.2 WEB PAGE CLASSIFICATION

At Subsection 2.1 we stated a need to calculate the relevance of a Web page to define its role in the crawl. The question addressed in this section is how to proceed for this calculation. In Subsection 2.1.4, we stated that the Web page can be considered as an object that has information such as the text, visual and structural content.

Most of the Information Retrieval (IR) research on hypertext classification in the WWW has been focused primarily on two areas: textual features of a document and the topological structure of a body of hypertext document (Lin (2016) and Choi and Yao (2005)). In the last few years another topic has been studied, the usage of the **visual context** of the page. In (Videira (2013) and de Boer et al. (2010)) the rendered page is used as input to the WPC.

The general problem of Web page classification can be divided into sub-problems such as:

- **Subject classification** concerns about the subject or topic of a Web page (i.e., arts, business, sports, games, shopping);
- **Sentiment classification** focuses on the opinion that is presented in a Web page, (i.e., the authors attitude about some particular topic);
- **Genre classification** refers to the role/functionality that the Web page plays (e.g., personal home page, course page or admission page, FAQ, product page, list of products page);

2.2.1 Traditional document comparison

Web page classification (WPC) problem is not the same as text classification. For instance, traditional text classification is typically performed on plain text confined to some sort of lexical/grammatical structure, Web collections do not have such properties. Also, Web pages are semi-structured documents in HTML or XML, which translates in the possibility of having embedded information that can be rendered visually for the user. Another important aspect is that a Web document is not a stand-alone file. A Web page exists within a hypertext, with connections to and from other Web documents (Web structure), which is central to the nature of the Web and is not present in typical text classification problems.

The usual approach in document classification usually consists of analyzing a collection of similarities such as the words and the way they are presented (lexical context) in the document. However, many concepts or objects can be described in multiple ways (different words/expressions) due to the context or even personal language preferences.

To boost the retrieval of pages, synonyms are commonly used to enhance context search for relevant pages. For example, "car" and "automobile" are synonyms in the context of

vehicles. If a query only has the word "car", relevant documents that contain "automobile" but not "car" will not be retrieved unless synonyms are taken into account.

2.2.2 Text representation

In Xu et al. (2011) other topics of Web document representation and text processing are approached such as:

- **Bag of words (BoW)**: Is a model where the exact ordering of the terms in a document is ignored, but the number of occurrences of each term is relevant. Thus, the text "John bought a gift for Mary" is identical to "Mary bought a gift for John". Nevertheless, two documents with a similar bag of words representations generally share similar content. This similarity can be used in the comparison of documents Cam (accessed July 16, 2019).
- **Term Frequency Inverse Document Frequency (TF-IDF)**: Is one of the most successful term weighting methods. Term Frequency (TF) is the number of times a term occurs in a document. Inverse document frequency IDF is the number of documents in which a term occurs at least once. The value, is obtained by the product of the local term (TF) and the global term (IDF) Aizawa (2003).
- **Latent Semantic Indexing (LSI)**: Is also based upon term frequency, however, it creates its list of features that aim to better model the relationship between terms (e.g. footwear and shoe are related). It aims to deal with this problem through the identification of statistical associations of terms. It is assumed that there is some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice.

Using these methods, the notion of a document vector that captures the relative importance of the terms in a document is created. The representation of a set of documents as vectors in a common vector space is known as the VSM. From these vectors, a correlation matrix can be created to see how similar any two documents are, which can be used for IR operations (e.g., scoring documents on a query, document classification, and document clustering). The standard way of quantifying the similarity between two documents d_1 and d_2 is to compute the cosine similarity of their vector representations $V(d_1)$ and $V(d_2)$:

$$\text{sim}(d_1, d_2) = \frac{V(d_1) \cdot V(d_2)}{|V(d_1)| \cdot |V(d_2)|} \quad (1)$$

where the numerator represents the dot product (also known as the inner product) of vectors $V(d_1)$ and $V(d_2)$, while the denominator is the product of their *Euclidean* lengths.

2.2.3 Machine Learning

Machine learning is a branch of Artificial Intelligence (AI) devoted to the development of algorithms and techniques that allows to recognize patterns from raw data. This allows learning from existing data without hardcoded instructions, just like humans are capable of doing. Essentially, machine learning systems use a vector of feature values as input data and may output a discrete value traduced by a label (classification) or a continuous value (regression).

As described earlier, the size of the Internet makes it impossible for a human to perform Web page classification manually. Therefore, the use of machine learning, mainly classification, to automate this process is expected to be a viable solution to tackle WPC problem. This learning process can be described in several ways, such as Supervised learning and Unsupervised learning.

Supervised learning consists on exposing a set of examples that are previously labeled, to establish a set of characteristics or rules enabling it to later identify and classify new examples. On the other hand, Unsupervised learning consists of not exposing any type of indication about the labelling. On this approach, the classifier receives only examples and has the burden to find out common characteristics to group the data. Some known classifiers are:

- **Naive Bayes (NB):** Is a statistical classifier. It is based on the naive assumption that the effect of different features within the same label are independent of each other. Although the assumption of independence among features is not a reality in practical datasets, Naive Bayesian is often good for cases where the input data is not overly complex.
- **Multi-class Support Vector Machines (SVM):** Each example is a dot in the n-D hyper-space. The purpose of the algorithm is to determine the set of hyperplanes (if possible) that separate the examples between classes so that in each region only exist elements of one class.
- **Decision Tree:** consists of a tree-like structure where the leaves are the labels and the branches are built by the patterns found within the features. When new data is introduced in the classifier, it performs a series of questions about its features by traversing the branches of the tree until it reaches a leaf (label).
- **Artificial Neural Networks (ANN):** These algorithms are based on nodes and connections between them. These nodes are referenced as neurons as this family of algorithms is inspired by the neural networks described in biology. One neuron usually outputs a single value while it receives multiple values from multiple connections to

different neurons. Its output is the result of an activation function that is applied to the sum of inputs the neuron receives. These networks may be organized in multiple layers where each layer receives its input from the outputs of the previous layer, thus information travels the network. In Subsection 2.2.4 we will delve into the specifics of some types of neural networks relevant to our work.

2.2.4 Deep Learning and Types of Neural Networks

Deep learning is a sub-field of machine learning based on multiple levels of representation and abstraction. Some examples of Neural Networks architectures are:

- Convolutional Neural Networks (CNN): is an efficient recognition algorithm which is widely used in pattern recognition and computer vision. There also have been recent studies that use CNN for the resolutions of text classification problems, presenting interesting results Johnson and Zhang (2015) and Li et al. (2018).
- Recursive Neural Networks (RNN): are characterized by having the notion of state and by the existence of a connection with the various passages over time. That is, each neuron receives information not only from the previous layer of the network but also information from itself, coming from the previous iteration. RNNs have a feed cycle in which the output of one layer feeds the next, along with the next *input*.

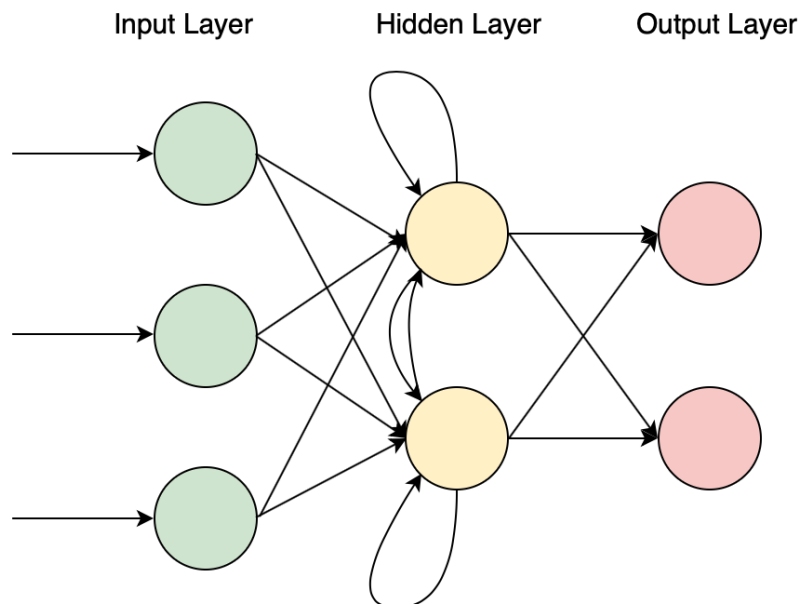


Figure 12.: RNN architecture

The notion of internal memory, which allows knowing the temporal behavior done so far, makes them ideal for data sets in which their samples are composed of large sequences. Figure 12 shows a possible architecture of a RNN in which a characteristic recurrence of these networks is visible.

2.2.5 Ensemble classifiers

Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions, the combination of several base classifiers built within a given learning algorithm, to improve generalisation/robustness over a single classifier. We can distinguish ensemble methods in two types of families:

- **Averaging methods:** The principle of its creation is the join of multiple classifiers averaging their predictions. In general, the predictor will perform better than a single classification method because its variance is reduced (e.g. Bagging methods, Forests of randomized trees).
- **Boosting methods:** the classifiers are built sequentially, we can visualize this approach as a type of decision tree in which one tries to reduce the bias of the combined estimator. Usually, the combination of several weaker models can produce a powerful ensemble (e.g. AdaBoost, Gradient Tree Boosting).

AbdulHussien (2017) shows that ensemble methods can be applied in the WPC, where an ensemble method (Random Forest) achieved better results (87.26 %) than the sole use of an ANN (84.82%).

2.2.6 Subject based classification and genre based classification

In our problem there will be a restriction both of subject and genre, i.e. a product page is respectively bound to the *E-commerce* and to the **single** item records presentation (e.g. price, specifications) that the user may buy. Some solutions found to both these type of problem are:

- Lin (2016) used pre-trained GloVe word embedding as the input of a residual network for a Web page subject classification. The text on the Web page was cleaned and converted to a semi-structured text using both tags and the text presented in them;
- Roussinov et al. (2001) Create a "subject" classification one of them was "**shopping sites**", being that subject composed by product information, advertisements, organi-

zational/business home pages, product lists, search result lists, and table of contents. So a "genre" was transformed into a "subject" problem;

What exactly is a genre?

In general, a genre could be described as a style of a Web page that is used to send a message to the user. This message has a topic, for example, a shoe brand. To a shoemaker, it will give a high number of objective facts about shoes, from type of materials to the process of making the shoe itself. When wishing to promote a product, it will communicate a message about shoes to amuse the user by presenting pictures and video material. In light of the previous explanation, the genre can be described as intentional styling of a Web page to communicate the topic in a specific manner.

In genre-based classification, Web pages are classified depending on functional or genre-related factors. In a broad sense, the word "genre" is used as a literary substitute for "a kind of text" or "a type of functionality". Per example an IR query such as *Nike* would retrieve many documents related to the company when submitted to a Web search engine, but they may be of different genre. This results may range from pages, such as a company homepage, brand history, product specification, product advertisement, or even a review of a product. Genre provides a new dimension for text retrieval and classification, in addition to topicality, and help users become more selective in their information seeking process and obtain higher quality information.

Confronted with heterogeneous domains, like the Web, has too many topics that no topic taxonomy can retrieve all of them in detail, genre classification can add another filter to WPC as a whole.

In (Roussinov et al. (2001)) is reported a study in the genre of Web pages to facilitate information exploration, identified which genres would meet searchers' information needs. This classification in groups of genres was used in an interactive search tool that allowed genre-based navigation, helping the user to narrow his search. They also created an interface supporting genre definitions to help in the UX.

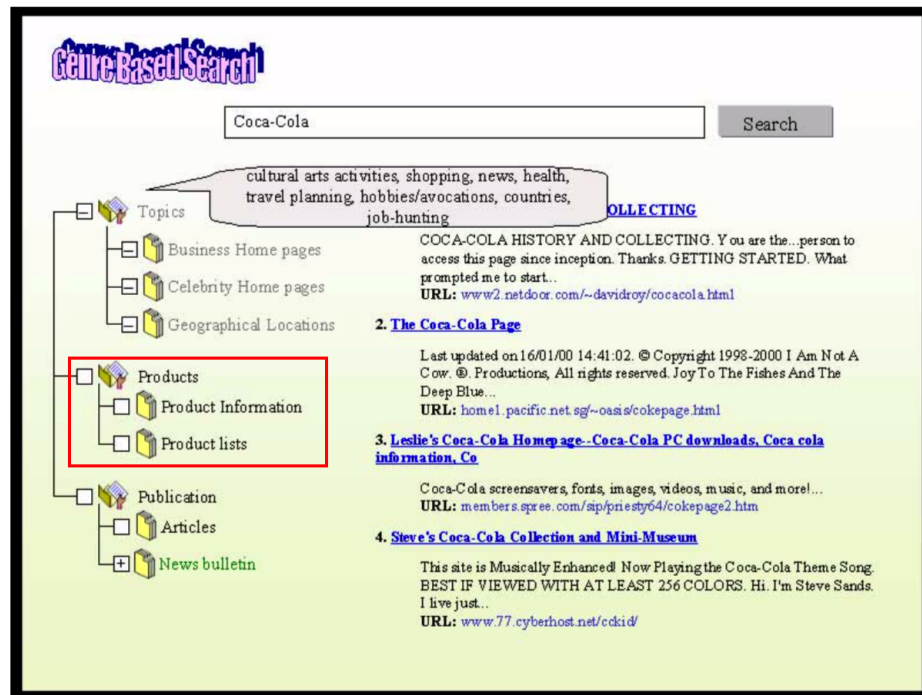


Figure 13.: Interface suggested, red square show an approach in our subject;
Adapted: Roussinov et al. (2001)

In Figure 13 they propose a subdivision for **Products**:

- **Product Information;**
- **Product List;**

One of the current market solutions for the Enterprise world guides their Web crawler with a classifier based in ANN that also proposes these labels as part of their solution. In **Datafiniti** solution (Dat (accessed January 16, 2019)) they explore an ANN with 3 layers, an input layer receiving data from the presence of some features in the Web page (i.e., price, image URL, number of clickable images adjacent to price values, keywords found in prominent positions (e.g., product detail, description)), an hidden layer and an output layer providing the probability of a page being a product page, catalog/category page, or some other category.

Their algorithm for the neural network took the following steps in Figure 14:

They state that with this approach they can accurately classify product pages about 90% of the time.

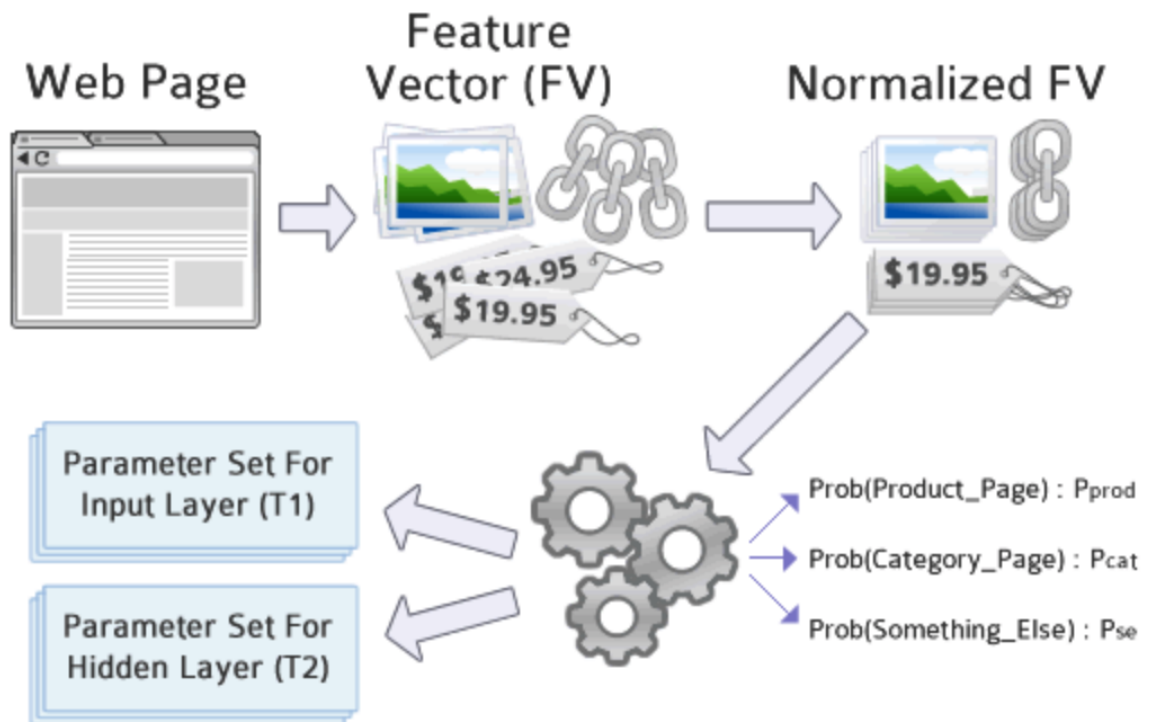


Figure 14.: Datafiniti predict page type methodology - Source: [Dat](#) (accessed January 16, 2019)

2.2.7 Visual analysis

Despite not being one of the first choices for genre classification as a global problem, the reality is that the visual information is one of the biggest resources humans use to identify the type of a Web page. Having each Web page at least two representations, text representation that is written in [HTML](#) and the visual representation rendered by a Web browser, it seems unproductive not exploring both possibilities.

Although the visual layout of a page relies on the tags, the use of the visual information from the rendered page can be considered a more generic approach for the document analysis, because a structure focusing on [HTML](#) tags may have multiple combinations that result in the same visual effect as can be seen in [Figures 15a](#) and [15b](#) respectively. Considering that Web pages are built by humans, and for humans, we can assume that the visual information should have more weight than the usage of tags.

In [Kovacevic et al. \(2004\)](#) and [Kovacevic et al. \(2002\)](#) is shown how visual layout analysis could be applied to improve the performance of a Web page classifier, when compared with more commonly used techniques, such as a standard bag-of-words approach.

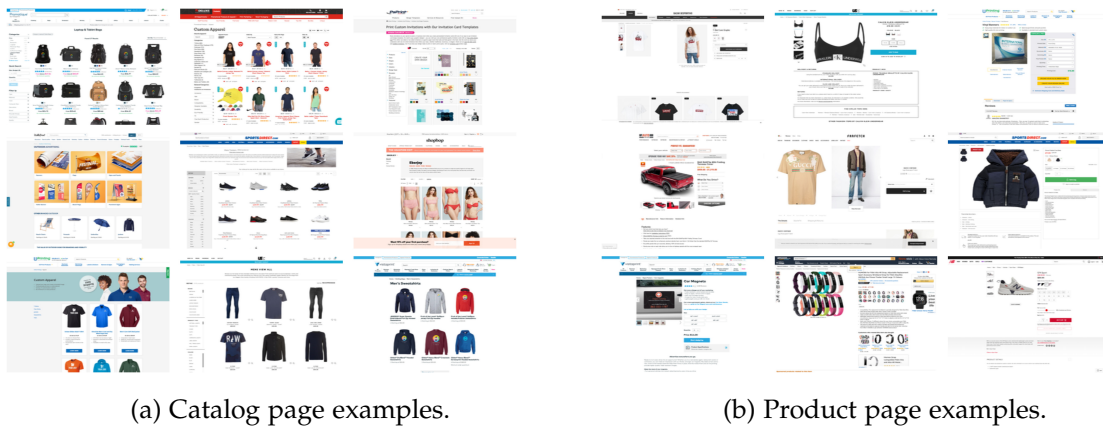


Figure 15.: Genre examples in E-commerce

In [Videira \(2013\)](#) and [de Boer et al. \(2010\)](#) they use the visual context to classify the pages based on their aesthetic value. Being this something subjective and not measurable, because "beauty" changes from person to person. They define "An ugly Web page, doesn't transmit a clear message, uses too much powerful colors, lacks clarity and consistent navigation. While a Beautiful Web page has an engaging picture, an easy navigation, the colors complement each other and its easy to find what the information needed." An example of this quote can be observed in [Figure 16](#).



Figure 16.: Aesthetic classification - Source: [de Boer et al. \(2010\)](#)

They also proposed a different classification using the visual context basing it on the page design, transforming the classification problem to a binary question of new-fashioned or old fashioned Web site (Figure 17) and achieved an accuracy of 95%.

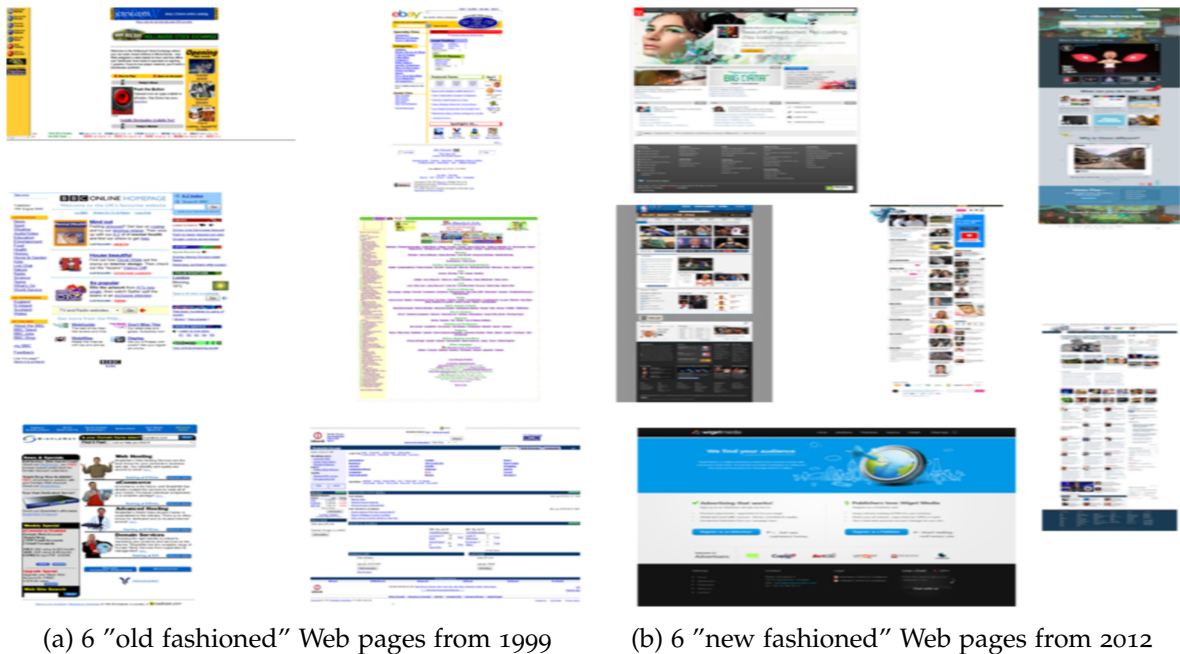


Figure 17.: Recency classification - Source: Videira (2013)

Both papers present one conclusion, the usage of visual context is a good path for improving results on WPC.

In Doosti et al. (2017) they show how state of the art technologies like CNNs could provide quantitative measures of Web site design styles, how to compare these designs, chart their evolution over time, and how to sample new designs from the CNNs. Their classification system presented results in WPC as seen in Figure 18, special attention should be given to the "Shopping" case.

For the WPC problem and the visual aspect of a Web page, we may consider the page as a solo image. In López-Sánchez et al. (2017) they do not, they obtain the HTML pointed by that URL, download every picture present in the document, disposing of advertisement and navigation images, and generate a case representation (i.e., a feature descriptor) grouping the images from the same class together and predicting the category of the Web page based on the individual predictions for the images.

In ao Mário Gonçalves da Costa (2014) both visual and textual context was used for WPC reportedly achieving an 84.38% accuracy on blog vs. non-blog classification and a multiple category classification (Games, Health, News and Shopping) of 98%.

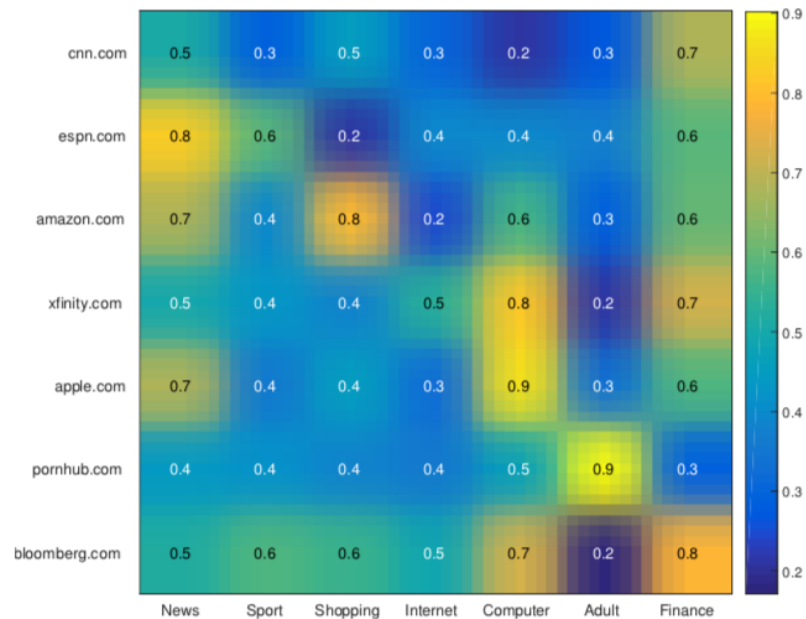


Figure 18.: Confidence of the network for seven Websites on seven genres - Source: Doosti et al. (2017)

In Levering et al. (2008) Web pages from different *E-commerce* sites were classified into several genres (e.g. Store Homepages, (Store) Product Lists, (Store) Product Descriptions, Store Others (Negative Class)). They compared three different sets of features for the WPC:

- just **textual features**;
- **HTML level features** added;
- **visual features** added;

They concluded that by using **HTML** features, more specifically, the **URL address features** they could improve classification beyond using textual features alone. They also show that adding visual features can be useful for further improving fine-grained genre classification.

In Kudelka et al. (2009) they developed a method for the detection of Web design pattern instances in Web pages. In their method, they consider 24 types of patterns (mostly e-commerce and social domain). Analyzing **architectural and semantical attributes** of solutions of the same tasks in the Web, the accuracy of the proposed method is about **80%**.

2.2.8 Link Classification

A uniform resource locator (**URL**) is the address of a resource on the Internet. An example is presented in Figure 19.

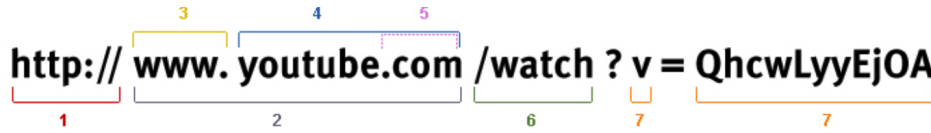


Figure 19.: URL structure example

1. The **Protocol** - this case: *HTTP* others: e.g. HTTPS, FTP;
2. The **Host** or **Hostname**: *www.youtube.com*
3. The **Subdomain**: *www.*
4. The domain name (**Domain**): *youtube.com*
5. The **Top-Level-Domain** (a Web-address suffix): *.com*
6. The **Path**: */watch* A path will usually refer to a file or folder (directory) on the Web-server (e.g. */folder/file.html*)
7. **Parameter** and **Value**: *v* (Parameter), *QhcwLyyEjOA* (Parameter value) Parameters are initialized by the "?" inside the **URL**.

Rather than deriving information from the page content, in Abramson and Aha (2012) and Kan and Thi (2005) is demonstrated that WPC can be done based on its URL. While not having as high accuracy (e.g., 76.18% in Kan and Thi (2005)), this approach eliminates the necessity of downloading the page. Therefore, it is especially useful when the page content is not available or **time/space efficiency** is strictly emphasized.

2.2.9 Feature selection

Feature selection aims to remove the less relevant features expecting to raise the precision of the model, and with less data, the training time will also diminish. Redundant or irrelevant features would just add complexity to the model which could cause overfitting and damage generalization, features that are not relevant for the target may make the model recognize false patterns. Further information can be found in Tang et al. (2014).

The features that are useful in WPC can be divided into two broad classes:

- on-page features: directly located on the page to be classified
- neighbors features: features that are found on related pages

Directly located on the page, the textual content is the most straightforward feature to consider using in *WPC*. However, due to the variety of uncontrolled noise in Web pages, directly using a bag-of-words representation for all terms may not achieve top performance.

A Structure-oriented Weighting Technique (*SWT*) [Riboni \(2002\)](#) can be used to weight the important features in Web pages. The idea of *SWT* is to assign greater weights to terms that belong to the elements that are more suitable for representing Web pages (such as terms enclosed in title tags, a header or large fonts in a page can indicate the major topic contained in the page). The same approach is followed in [Sarhan et al. \(2015\)](#). In general, this will be translated in an algorithm responsible for feature extraction of a *HTML* file Input.

This type of feature selection is not new to the classification for E-commerce Web pages, with some variations the majority of the solutions present some type of preprocessing in that aspect. In [Moiseev \(2016\)](#) is presented an algorithm that weights terms against the nearest tag they are nested in, and calculates the weight of tags inversely proportional to their frequency (i.e. the more frequent the tag is, the less valuable enclosed terms are). In [Petprasit and Jaiyen \(2015\)](#) they create an automatic product classification, using four components of *HTML* including **tag name**, **title tag**, **keyword tag**, and *CSS* properties.

Sometimes the selected features for *WPC* are missing, misleading, or unrecognizable for various reasons, so classifiers have problems making reasonable feature-based judgments. To address this problem, features can be extracted from neighboring pages that are related in some way to the target page [H. Haveliwala et al. \(2003\)](#). An example of such a connection is the hyperlink (Figure: 20).

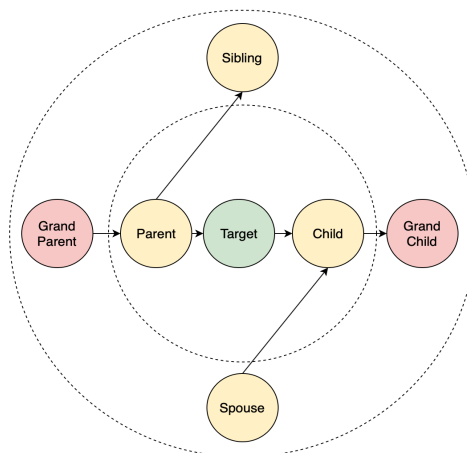


Figure 20.: Pages Family Tree

2.3 WEB SCRAPING

Web scraping, Web harvesting, or even Web data extraction is the action of extracting data from Web sites and can be performed manually by a software user. However, the term typically refers to **automated processes** implemented usually using a Web crawler where specific data is **gathered and copied from the Web**, typically into a central local database or spreadsheet, for later retrieval or analysis.

Web scraping a Web page involves fetching (downloading) a Web page, like a Web crawler or a browser acts upon the action of viewing a page, and extract data from it. The content of a page may be parsed, searched or even reformatted. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else e.g. find and copy prices from a product page for price change supervision and price comparison.

In a large number of sites, manual labeling is not suitable as it proves to be too big of an effort. Hence, there is a necessity of a general and automatic way to scrape. For example, if a shopping site wants to extract all the products sold on another e-commerce site, manual labeling becomes almost an impossible task.

The automation of this process presents its challenges, as a *special fit* scraper for a *specific target* may prove ineffective after a small period as the Web is a dynamic environment and Web sites change constantly. Since the scraper systems mainly rely on [HTML](#) formatting tags, if a Web site changes its formatting templates, the existing scraper for the site may become invalid.

This is a problem that still doesn't have a fully correct answer, automatic verification and repair are still difficult. Doing them manually is costly if the number of sites involved is too large. Due to these problems, automatic/unsupervised extraction is a theme that is still being studied by researchers.

The reality is that automatic extraction is possible because the data records in a Web site are usually encoded using a very small number of fixed *templates*. It is possible to find them by mining repeated patterns in multiple data records. The reference of the term "*templates*" does not refer to hidden templates employed by Web page developers. We use the term "*patterns*" to refer to regular structures that the system has discovered.

2.3.1 Pattern Matching

Once the encoding template pattern is found, it can be used to extract data from other pages that contain data encoded in the same or in a very similar way. There are three ways to perform the extraction:

- **Finite-state machines:** The encoding template pattern is usually represented as a *regex* (regular expression). Non-Deterministic finite-state automation can be constructed to

match occurrences of the pattern in the input string representing a Web page, in that process data items are extracted.

- **Pattern matching:** directly matching the string or tree pattern against the input to extract data. It is a more flexible method than finite-state machines because pattern matching allows partial matching. For example, on the page where the pattern is discovered, an optional item does not occur, but it occurs in some other pages. As this pattern matching is not static it can deal with this easily and in the process, the pattern can be enhanced as well by inserting the new optional item.
- **Extracting each page independently:** In the case that a Web site uses many different templates to encode its data, a recognized pattern in one type of template will not extract the information from another template. One solution is to make a **special fit** for each page **individually**, being this method **inefficient**.

For the detection of new templates without sacrificing the efficiency of mining extraction patterns from each page, a pre-screening strategy may be applied. In most applications, the user is interested in only a particular kind of data, e.g., products prices, products promotions or product reviews. It is usually possible to design some simple and efficient heuristics to check whether a page contains such data. If so, a full-blown extraction is performed using already generated patterns. If no data is extracted from the page, it is an indication that the page uses a different template and a new mining process can be initiated to discover that new template.

2.3.2 Published Approaches

In **Datafiniti**, the extraction of data is done using the static features of a page. However, since pages are mutable, the information they present is not an exception. In the market solutions, only **DIF** (accessed June 25, 2019) is starting to resolve these problems in their beta version allowing the user to perform actions (e.g., loading more results) before extracting the data.

Being proprietary solutions, information about the methodology used in these solutions is hard to find. Therefore, we also investigated published approaches to solve data extraction issues.

- In **More** (2016) the extraction system detects attributes in product titles of *E-commerce* retailers. The absence of syntactic structures in such short pieces of text makes extracting attribute values a challenging problem.

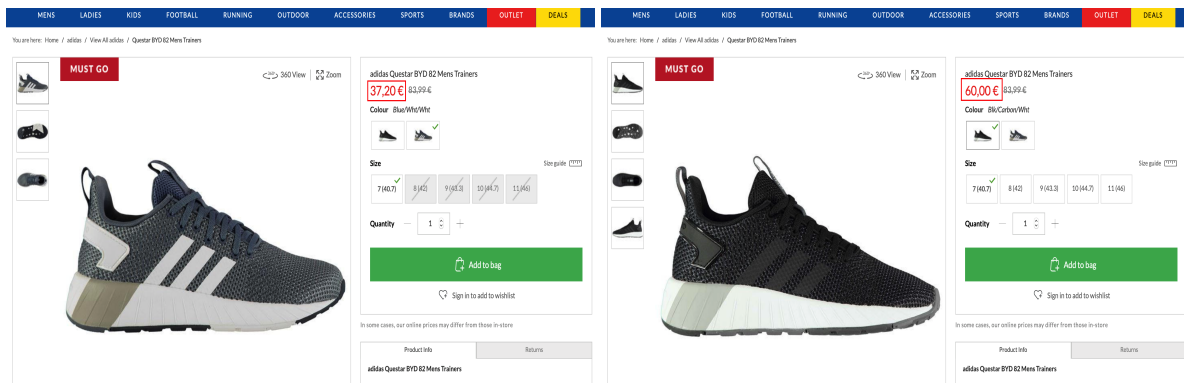
- In Liu et al. (2004) they compare child nodes of each node within a [HTML](#) tree, starting at the root node and identifying similar node combinations to discover data regions in Web pages.
- In Gurský et al. (2014) product attributes and their values are extracted directly from E-commerce sites using an annotation tool that is **integrated into a Web browser**. The identification of the data is **not automated**, being the users' responsibility to label the relevant product attributes of a sample product.
- In Liu et al. (2004) by repeating data regions, attributes and values from other products in the e-shops similar to the pre-annotated ones, they create a type of *community effect*. The extraction rules are represented in [JSON](#) and include the type of data that should be extracted, attributes and values (using [XPath](#) expressions).
- Cenys and Grigalis (2014) cluster the HTML elements of a Web page by the similarity of their [XPath](#) Strings and [CSS](#) elements. The product attributes are then extracted using "*HTML tree hopping*" technique which compares the items within the detected data records for determining the attribute separators within the HTML sub-tree of the data records.

This study is important in two key aspects, the final extraction of the price present in a product page and in the integration of some automation system that allow to perform actions over the page and see if those actions (i.e., selection of new values) will have an impact on the price of the product.

Most of these solutions find the encoding template (pattern) from a collection of encoded instances of the same type and create a *community environment effect*, where new examples may be combined to improve the performance of the extraction process. String matching, tree matching, and visual features matching are some techniques for the task. Tree matching is useful because [HTML](#) encoding strings also form nested structures due to their nested [HTML](#) tags. Such nested structures can be modeled as trees, commonly known as [DOM](#) (tag) trees and be of extreme use in the collection of relevant data.

However, the Web pages are action responsive, and the studies presented do not tackle that issue considering only the static features of the page. As can be seen in Figure 21 responsive design may change the product information.

In an *E-commerce* scenario this will be even more important, as the change of the product specifications may affect directly and dramatically the product price.



(a) Shoes - Blue/Wht/Wht

(b) Shoes - Blk/Carbon/Wht

Figure 21.: Different price in the same product page - color change

2.3.3 Automation Tests

Software testing can be defined as the process of confirming if a system satisfies specific requirements. The main goal behind the software testing process is to detect any possible anomalies in a software product. It can be done manually or automatically; manual testing besides being error-prone is a time-consuming process. In other words, **Automation Testing** is the use of testing tools to cut the need for manual or human intervention and discover defects manual testing cannot expose.

A testing automation framework is an execution environment for automated tests used to find problems at an earlier stage and solve them. Those testing tools are designed to generate automated tests and enhance testing performance. Hence, reducing time, cost and effort. However, they can be used for other purpose that just testing the users' own software. Since the Web pages of different domains and the user interaction with them share common points, these interactions can also be automatized following the same principles of testing. Some of these tools are:

- **Sikuli Yeh et al. (2009)**: Is a tool that uses image recognition powered by **OpenCV** to identify and control **GUI** components. This is handy in cases when there is no easy access to a **GUI's** internals or the source code of the application. Besides locating images on a screen SikuliX can **perform actions** (i.e., run the mouse and the keyboard to interact with the identified **GUI** elements).
- **Selenium Sel (accessed January 13, 2019)**: Is a tool that automates browsers, performing actions over Web pages, and provides a loarge number of possibilities. Primarily, it was created for automating Web applications for testing purposes however is certainly not limited to just that. In **Stouky et al. (2018)** and **and (2017)**, Selenium was directly compared against other testing tools having a positive classification.

2.4 SUMMARY

This chapter started by mentioning that the web is a great data platform from which companies may gather information. There already exist companies that collect and sell this information.

Those solutions are usually composed of two stages. The crawler that indexes target pages, and the scraper, that gathers information from those pages.

Hence, in Section 2.2, some of the most popular *WPC* models were enumerated for guiding a crawler. Emphasis was given to *ANN*, as they are currently one of the most popular models, justified by the increasing interest in deep learning.

In Section 2.3, published works in data extraction were analyzed. However, these works focus on stale data and Web pages are dynamic objects (i.e., actions over the page may change its content). Thus, we study test frameworks to add dynamism to the solution and increase the quality of the results.

PROBLEM FORMALIZATION AND SOLUTION DEVELOPMENT

This chapter describes three important stages that are a prerequisite for the development of the crawler and the scraper. These elements are the main pieces of the solution proposed in this project being their construction also addressed in this chapter. It is divided, according to the [CRISP-DM](#) methodology ([Chapman et al. \(2000\)](#)):

- **Business understanding:** The first phase is to understand **what must be accomplished**, from a **business perspective**, and plan the strategy and techniques to achieve that goal;
- **Data understanding:** This phase includes describing and exploring the available data for solving the issue;
- **Data preparation:** This stage consists of getting the data ready to be processed. It involves feature selection, data cleaning, deriving new attributes, or transforming values of existing attributes. This work focuses on the last point, comparing different encodings for converting categorical features into numeric values.
- **Modelling:** This phase details key aspects of the implementation, in the selection and application of modeling techniques, and the calibration of their parameters to optimal values.

3.1 BUSINESS UNDERSTANDING

A company's most valuable source of data does not reside in a [CRM](#) software, in [POS](#), [POP](#) systems or even in [ERP](#) platforms, the biggest source of data is the Web. Despite the company's internal data systems providing a lot of valuable information, they do not share the same potential that exists in a monolith of information that connects all the parties associated with the E-commerce process. This relates to problems such as:

- Sales lead generation and optimization;
- **Competition analysis and supervision** (project main focus);

- Product pricing and assortment;
- Brand and sentiment analysis;
- Marketing automation and research;

These can be aggregated and treated by a company's BI department. For a business focused on performance-based solutions, the effectiveness of those solutions can be measured in enhancing the sales of some product or service. Thus, supervising the competition (i.e., how often they change their products, products price range, stock), is valuable information e.g. product pricing. Note that this application does not return a product price suggestion, but a concurrent price of sale. Such information will provide valuable insight into the process of selection of the best price to display. Nevertheless, that is an optimization problem that is outside the scope of this project.

Hence, from a business perspective, a system that can instantaneously detect changes in the players on the market, the products they offer and their specifications is an advantage. This requires the identification and extraction of product attributes from diverse E-commerce Web sites, so there is a need to:

- **E-commerce Web sites detection:** Automated detection of e-commerce web sites of the same product domain;
- **Product page detection:** Automated detection of all Web pages containing single product records within the E-commerce Web sites.
- **Product attribute extraction:** Automated detection, extraction and structured storing of the product attributes from the product records. In Woods (accessed March 22, 2019) are stated issues from acquired web data in these type of cases:
 - **Identification of the information on a Web page or collection of Web pages and the assemble of information into a useful structure**
 - **Correctly** harvesting of data even if the **Web page changes** in some way
 - **Recognition of new information** in the Web page.
 - **Scheduling** of data **harvesting**
 - **Managing and performing** quality control on **multiple agents**
 - **Handling** complex ways of creating pages such as **responsive design**
 - Integrating harvested data into a **data warehouse** or other repository

Considering the current market situation of *E-commerce*, regarding the number of online retailers as well as the diversity of product categories, the **generality** of a classification method becomes also a point of interest:

- The globalization brings the possibility for an *E-buyer* to purchase products from *E-shops* of different countries, thus, a **cross-border competition**. This will lead to a need for a **language-independent approach**.
- The *E-commerce* is a cross-business platform, the products sold online are diverse and from different companies. Therefore, a solution **independent from a specific product domain and dimension of the E-commerce platform** is necessary.

This classification problem in a real-world scenario may be divided into a trinity (Figure 22) and the variation of the importance of each part is of the system user responsibility, not meaning that the improvement in one of those areas will directly negatively impact another.

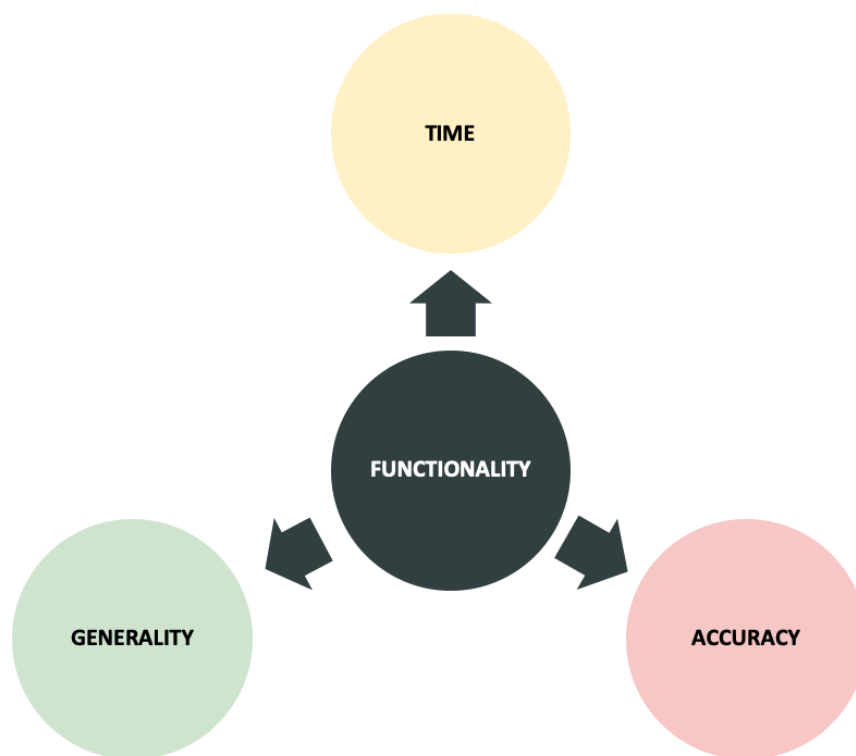


Figure 22.: Trinity for crawler functionality

- **Time:** The purpose, from a macro perspective, is to shift the burden of browsing through web pages and the market information gathering away from the user. The transition from a manual process into a low maintenance automatic process of
 - competition detection;
 - product Web pages detection;
 - product details extraction;

This will result in the reduction of man-hours spent in market studying, and a general growth in the efficiency and quality of the data extracted, allowing the business administrators to make correct marketing strategies for each product.

- **Generality:** The quality of the solution directly implies its ability to adapt. As stated in Section 3.1, the solution needs to account for:

- different Web sites;
- different languages;
- different product domains;

In two major stages of the solution:

- navigation classification (i.e. **crawler**);
- information extraction (i.e. **scraper**);

- **Accuracy:**

In the first stage, the metric used as the evaluation function of the classifier is the **overall accuracy**. This is a simple metric that gives the proportion of the correctly classified data. Overall accuracy is the easiest to calculate and understand but ultimately only provides basic accuracy information. It is most useful when used to evaluate the performance of distinct classifiers comparatively, the quality of a single **accuracy** value can often be interpreted as:

- 0.33, the performance of a random classifier (3 classes);
- 0.6 to 0.7, **poor**;
- 0.7 to 0.8, **fair**;
- 0.8 to 0.9, **good**;
- 0.9 to 1, **excellent**.

However classification accuracy alone may not be enough information to make a decision. Therefore, for some cases we also study the following items (Hossin and M.N (2015)):

- Precision: is a good measure to determine, when the cost associated with False Positives are high.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2)$$

- Recall: is a good measure to determine, when the cost associated with False Negatives are high.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (3)$$

- F-Score: is used when a balance between Precision and Recall is needed.

$$F - \text{Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

These metrics are usually applied to probabilistic, binary classifiers can also be derived to multi-classification problems (e.g., each class against the rest).

While the two approaches to statistics measures are likely to be correlated, they measure different qualities of the classifier.

A second stage is directly linked to the quality of the information extracted from those pages (i.e., selected attributes from the page (e.g., product color, size) and the price data). The quality of these results is manually verified, comparing them to the real values present on the respective Web page.

3.2 DATA UNDERSTANDING

In Section 2.1 the start point for the crawl is left unanswered. In our solution that data is gathered by two methods:

- Providing **Seed Pages** to crawl: This approach suggests a previous knowledge of the market or already have a target in mind. This may cause the introduction of a new player to pass unnoticed.
- Providing **keywords** to crawl: Crawlers usually use a general Web search engine for providing starting points (Subsection 2.1.6).

Google is the number one Web search engine (Sta (accessed December 26, 2018), Maddox (accessed December 28, 2018)). However, despite being a great font of information, its results are not limited to E-commerce sites, potentially delaying the crawler. An example can be seen in Figure 23 where a search for shoes can generate different types of pages (e.g., Online encyclopedia, blog, and E-commerce Web site).

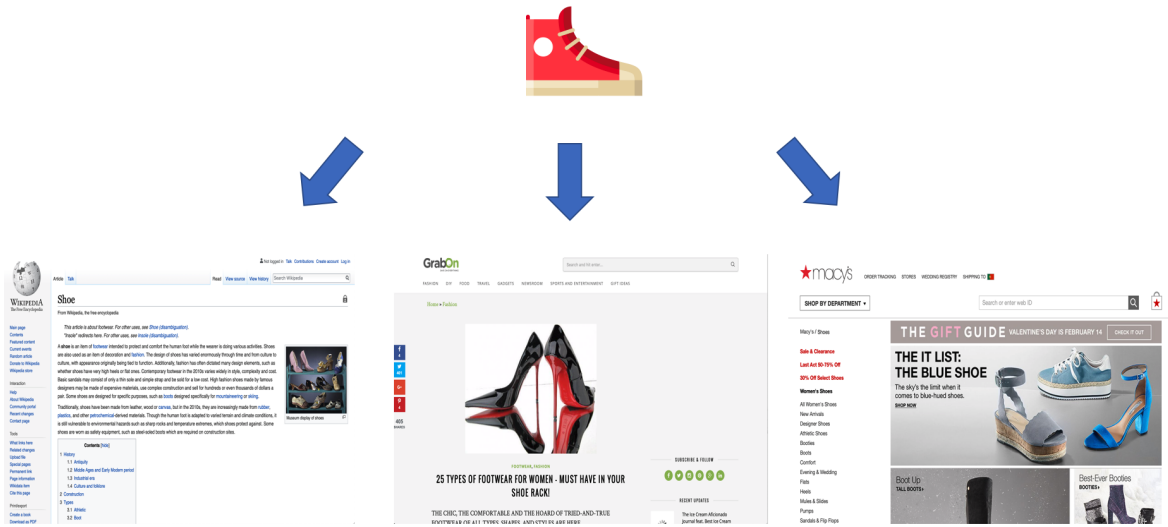


Figure 23.: Types of pages resulting of the search for "shoes" keyword - Google

To restrict the scope of solutions from SERP to E-commerce Web sites we use an E-commerce site shopping agent (mySimon (accessed December 23, 2018)) that will only provide E-commerce Web sites as solutions (Figure 24).



Figure 24.: Types of pages resulting of the search for "shoes" keyword - MySimon

In Horch et al. (2015) the schema of E-commerce Web sites is studied, they search until the great-grandson of the seed page (Homepage of the E-commerce Web site) is achieved. In their dataset 76% of the analyzed E-commerce Web sites have product lists (catalogs) on level 0, 96% have product lists on level 1, 74% show lists of products on level 2 and 12% of

the *E-shops* have product lists on level 3.

Therefore, for their dataset, in the majority of E-commerce Web sites, it is highly unlikely that a Product Web page would be found beyond the **fourth** level from the Homepage.

Following the results of the analysis of the distribution of the **product lists** within E-commerce sites, the approach crawls all Web site links for reaching all pages until **level 4**. These facts also portrait what is stated in [Rababah and Masoud \(2010\)](#) and even in our dataset, generically, they are confirmed. However, the approach taken by their work consists of doing an exhaustive search until the last level. In a practical solution, depending solely on the level of a *hops* for determining if a page should or should not be crawled has a huge impact in performance matters. Considering that a simple middle size site may have **200** links redirecting to other pages, the possible accumulation of those numbers for 4 levels (**1.600.000.000**) is too big to be easily deployed. Therefore, our approach tries to trim the excess in these pages.

In [Figure 25](#), we define the genre/class according to the functionality of the Web page (i.e., role it plays to the users). Before the **URLs** being added to the frontier we filter the ones that were disallowed to crawl in the *robots.txt* of that Web site.

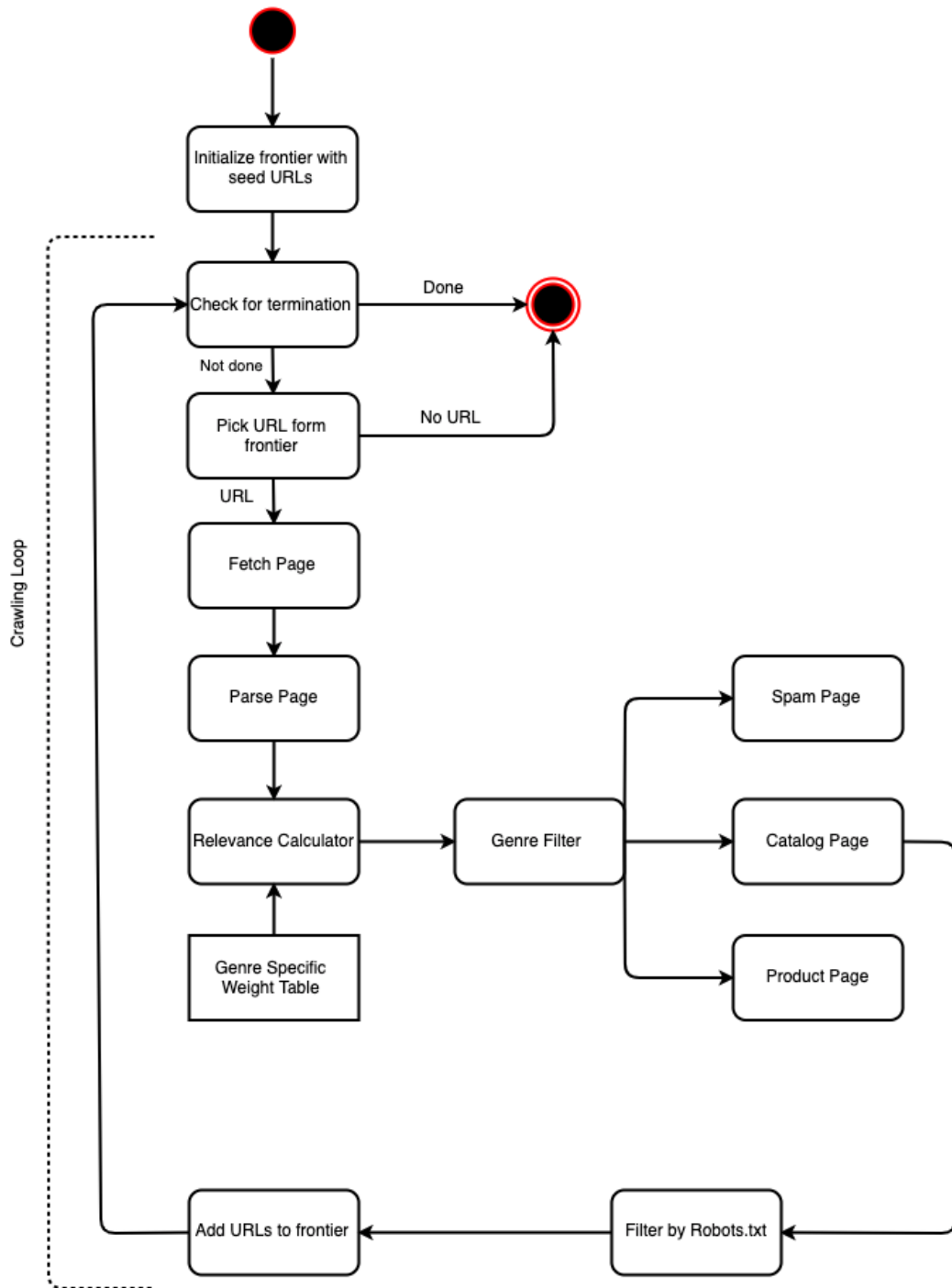


Figure 25.: Flow Chart Crawl

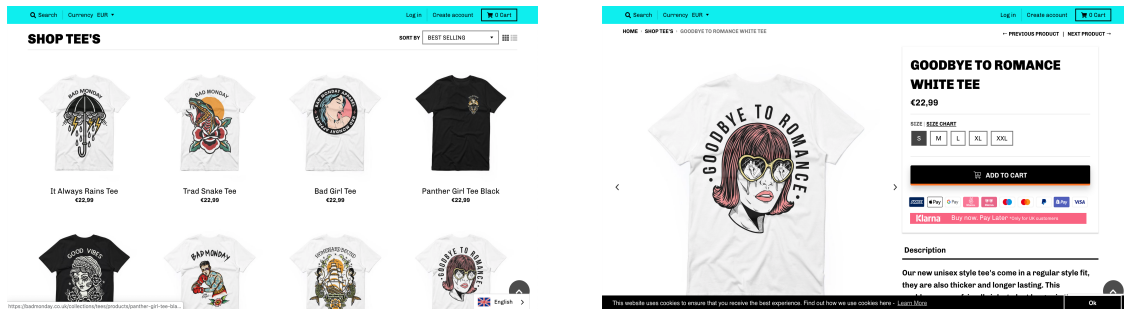
After selecting and analyzing hundreds of popular Web pages, and studying solutions (e.g., [Dat](#) (accessed June 20, 2019), [Horch et al. \(2015\)](#), [Roussinov et al. \(2001\)](#)) for the

problem and considering users requirements to these pages, we defined **three** type classes (Figure 26).

- **Catalog** pages: This class is the agglomeration of products, i.e, list of products, containing multiple sources of price information, and clickable images which can show some larger images or redirect to a product page.
- **Product** pages: This class refers to Web pages that have information about one specific product, allowing the user to buy it (i.e. add to the shopping cart). Here normally is presented a big picture of the product and also submit buttons that let users add the product to a shopping cart and posteriorly submit the order or even change some characteristics of it (e.g. size, color, quantity) by clicking on it.
- **Spam** pages: Other specifications of this type of page is everything that does not fit in the other two types of genres. Inside the E-commerce domain, spam pages have some common traces between them and can be included in the following sub-categories:
 - Resource download;
 - Pure information page (e.g., about the company);
 - Frequently asked questions ([FAQ](#));
 - Term paper;
 - Blogs;
 - News report;
 - Social media;

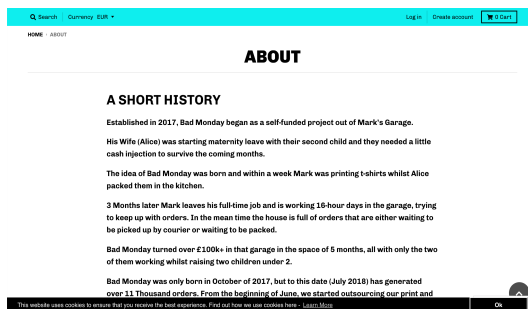
Each sub-genre has its functional purpose and characteristics, e.g., [FAQ](#) pages gather several frequently asked questions about a certain topic and usually organizes all the questions in **lists**. Most of the questions can be detected by matching the **interrogative at the beginning of the sentence** and the **question mark at the end of the question**. In many cases, each question follows "Q:", and each answer follows "A:".

Falling out from the E-commerce scope are Web pages of news reports about the company or a special product, they also tend to have some common characteristics. The title of the news should appear in the top, probably in a bigger font, and then followed by the authors name and the date/time it was released. Images are sometimes embedded in the content. After the news, there are several **links to other related news**. The same applies upon the entry in the company's social media accounts Web pages(e.g., *facebook, pinterest, instagram*) that have connections to other social media links.



(a) Catalog page example

(b) Product page example



(c) Spam page example

Figure 26.: Genre examples in E-commerce

The WPC inputs are gathered from the Web (i.e., Web Structure Mining (Figure 29)). However, a major problem for the quality of the data is the heterogeneity of the Web. Web pages **functionality**, **visual aspect** and **content** may be achieved in more than one way.

As Figure 27 shows the information about *shipping* costs are presented in a table.

DELIVERY ZONE	UNTRACKED SHIPPING	TRACKED SHIPPING	DELIVERY TIMES
UK	N/A	£3.50 (FREE OVER £60)	2 - 4 working days
USA, Canada, Australia	£7.99	£14.99 (FREE OVER £100)	5 - 10 working days
Europe	£5.99	£10.99 (FREE OVER £100)	5 - 7 working days
Rest of the World	£7.99	£14.99 (FREE OVER £100)	up to 14 working days

*All orders are print on demand so this includes printing and packing times. Delivery times are estimates and usually arrive in these time frames. Although sometimes it can be quicker or longer in special cases.

For more information on shipping please [click here](#)

Figure 27.: Visual table

However that does not translate in the usage of a `HTML` tag table, instead, the Web developer opted to use an image (Figure 27).

```

```

Figure 28.: Html representation of Figure 27

This is not an isolated case, however, inside the same domain, a web site tends to maintain truthful to one type of template that better suits its functionality (genre).

3.3 DATA PREPARATION

The challenge with using data from the Web (Web mining) is that it is incredibly diversified in structure and content and scattered across a vast number of Web sites. First, we need to understand and preselect the relevant data present in the Web for solving our issues (Figure 29).

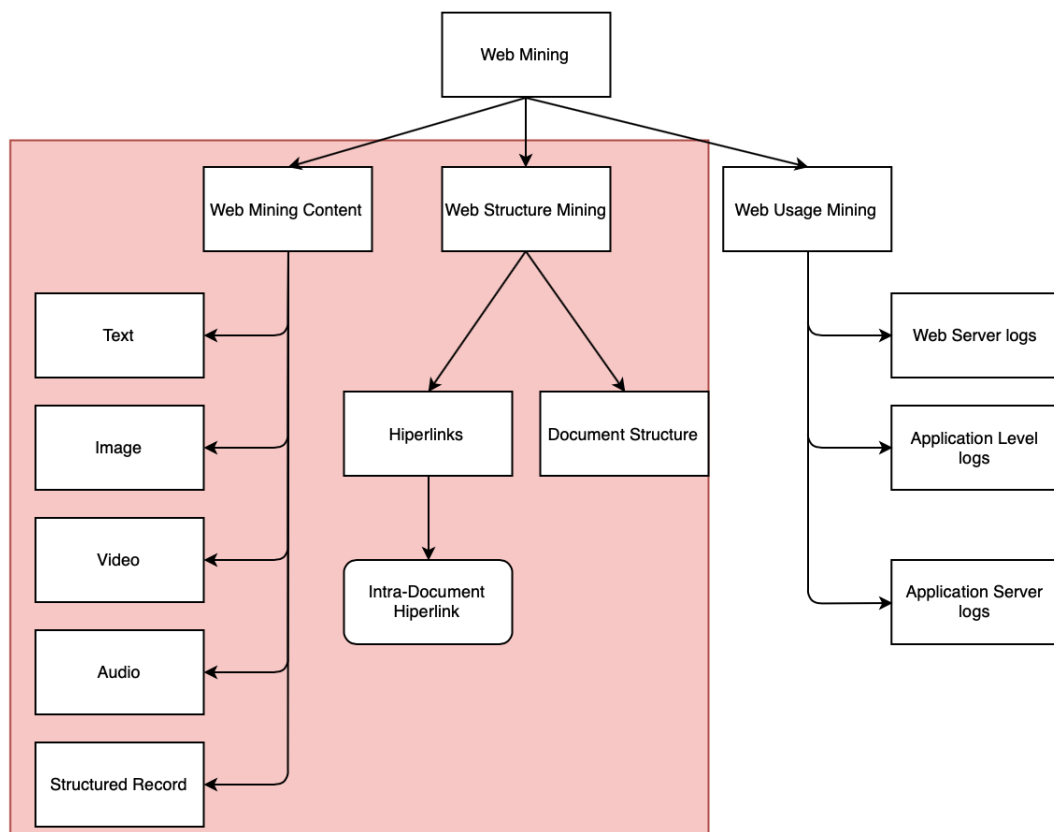


Figure 29.: Restrictions over the Web Mining Tree

We develop a set of methods to extract features from Web pages to identify its genre/class, mainly centered in three types of Web information:

- **Textual content;**
- **Structural content;**
- **Visual content;**

Another problem besides the unnormalised content of a Web page is the acquisition of the information. This project tries to make Web data accessible to businesses. Among the many challenges in doing so, it is imperative to gather Web data ethically and responsibly. In Subsection 2.1.8, we explain the *robots.txt* file and its purpose. However, it is ignored by many crawlers, in Figure 30 the problem is stated, and even Webmasters satire the situation.

```
User-agent: Download Ninja
Disallow: /

# Misbehaving: requests much too fast:
User-agent: fast
Disallow: /

#
# Sorry, wget in its recursive mode is a frequent problem.
# Please read the man page and use it properly; there is a
# --wait option you can use to set the delay between hits,
# for instance.
#
User-agent: wget
Disallow: /

#
# The 'grub' distributed client has been *very* poorly behaved.
#
User-agent: grub-client
Disallow: /

#
# Doesn't follow robots.txt anyway, but...
#
User-agent: k2spider
Disallow: /

#
# Hits many times per second, not acceptable
# http://www.nameprotect.com/botinfo.html
User-agent: NPBot
Disallow: /

# A capture bot, downloads gazillions of pages with no public benefit
# http://www.webreaper.net/
User-agent: WebReaper
Disallow: /
```

Figure 30.: Example of robots.txt with bad Web crawlers

Despite having benefits (i.e, not having the crawl/scraping blocked), the collection of data in an unethically/stealthily way would only prove detrimental in the long run. Web data acquisition has become a vital necessity for any data-driven company. Often, such a company is willing to cut ethical corners or outsourcing these tasks to companies who will.

Acquiring Web data by disguising our crawler (e.g, as a Web browser) abuses a privilege that has been granted willingly and in the long term will make the data ecosystem less open and less trustworthy. Thus, in this project we follow [CREAD](#) (Subsection 2.1.8).

3.3.1 Textual Context

Before proceeding to the classification phase, there is a need to process the information, to clean our data. For textual content, besides the full content of the page, we pay particular attention to several relevant items ([Sara-Meshkizadeh and Rahmani \(2010\)](#)):

- URL ([Anagnostopoulos et al. \(2002\)](#), [Abramson and Aha \(2012\)](#) and [Kan and Thi \(2005\)](#))
- Title content;
- Meta-description of the Meta tag;
- Span content
- Anchor content
- Headers content;

Although the span tag is rarely found in previous works, we found it to be necessary for the *E-commerce* context. In the majority of *E-commerce* Web sites the the product price is enclosed in **span** tags (Figure 31).

```
<span class="price-total-value js-price-total-value" data-analytics-total-price="39">$39.00</span>
    <span id=
      "dnn_ctr103436_ViewTemplate_ctl00_ctl08_lblSellingPrice">
      £70.00</span>
```

Figure 31.: Span tags - E-commerce web page examples \$ and £

We follow the model proposed in [Lin \(2016\)](#) to created a normalized version of the Web page text content. To help in this work we imported *BeautifulSoup* library for tags detection and text extraction. Examples of the normalized text are:

```
<url>https://www.neimanmarcus.com/en-pt/ <title> Designer Clothing, Shoes,
  Handbags, & Beauty | Neiman Marcus <meta> Free Shipping & Free Returns
  at Neiman Marcus. Shop the latest styles from top designers including
  Michael Kors, Tory Burch, Burberry, Christian Louboutin, kate spade &
  more <meta> Michael Kors, Tory Burch, Burberry, Cole Haan, kate spade <
  meta> <heading> My Account <heading> Explore <heading> Customer Service
  <anchor> Shoes <anchor> Handbags <anchor> Jewelry & Accessories <
```

```

anchor> Beauty <anchor> Men <anchor> Kids <anchor> Home <anchor> Gifts
<anchor> Sale <anchor> Call Us 24/71.888.888.4757 <anchor> Overview <
anchor> Order History <anchor> Address Book <anchor> Payment Info <
anchor> Shipping & Delivery <anchor> Contact Us <anchor> Store
Locations & Events <anchor> Magazine <anchor> The Heart of NM <anchor>
Need Help? <anchor> Order Tracking <anchor> Returns & Exchanges <anchor>
Payment Options <anchor> Security & Privacy <anchor> Need Help? <
anchor> Order Tracking <anchor> Returns & Exchanges <anchor> Payment
Options <anchor> Security & Privacy <anchor> Shipping & Delivery ...

```

For the full content of the page:

```

<url>https://www.neimanmarcus.com/en-pt/ <body> Skip To Main Content FREE
SHIPPING + FREE RETURNS EVERY DAY Neiman Marcus Sign In / Register my
favorite icon SHOPPING BAG DESIGNERS WOMEN'S CLOTHING CONTEMPORARY
SHOES HANDBAGS JEWELRY & ACCESSORIES BEAUTY MEN KIDS HOME GIFTS SALE
Last Call Sale Up to 75% off select women's styles Up to 70% off men's
& accessories Up Next - Look forward to fall with street-style inspo
from the latest designer collections Step Into Fall - Upgrade your
stride with new booties, slides & more Shop Shoes Shop All Handbags NEW
FOR YOU Gentle Fluidity - New exclusive fragrance from Maison Francis
Kurkdjian Dresses - Polished styles perfectly suited for office hours
(& happy hour) MEET YOUR NEW 9 TO 5 Daily Uniform - Look sharp in
modern sport coats with fresh shades Shop Sportcoats & Blazers Tote It
...

```

After the normalization of the text we move on to the next stage (**Text Processing**) Liu (2011):

1. Data normalization

Considering a web page some items deserve particular attention regarding the classification problem. For example, price is composed of the dollar sign or other currency sign followed or preceded by digits, we need to normalize this data in a way that a computer system can relate them. To address this issue we can resort to regular expressions to extract the required features.

- **Case of Letters:** All the letters are converted to lower case.
- **Digits:** Numbers and terms containing digits are removed in reviewed IR systems except for some specific types, e.g., dates, times, phone numbers, and other prespecified types expressed with regular expressions. In our application

prices and discounts (number followed/preceded by %) are particularly relevant. In our approach, we group a term with an index representative of different prices/discounts presented in such page, e.g. "price" + index. When considering URLs numbers are indexed in the same way.

- **Hyphens:** Breaking hyphens are usually applied to deal with the inconsistency of usage. For example, some people use "first-order", but others use "first order". Note that there is more than one way to deal with the hyphen issue. We considered two options:
 - the hyphen is replaced with space (" ");
 - the hyphen is removed without leaving a space so that "first-order" will be transformed to "firstorder";

In some systems, both forms are indexed as it is hard to determine which is correct. For our approach, we chose to replace the hyphen with space.

- **Punctuation Marks and special characters:** These characters are always replaced with spaces.

2. Stop Word Elimination

Stopwords are irrelevant frequently occurring words in a language that help construct sentences but do not represent any content of the documents.

Articles, prepositions and conjunctions and some pronouns are some of the common elements in this list. Stopwords in English may include words like:

```
a, about, an, are, as, at, be, by, for, from, how, in, is, it, of, on, or
, that, the, these, this, to, was, what, when, where, who, will,
which, with
```

Removing this kind of words reduces the number of terms, which usually makes the classification process easier.

3. Stemming

In many languages, a word has various syntactical forms depending on the contexts that it is used. For example, in English, nouns have plural forms, verbs have gerund forms (by adding "ing"), and verbs used in the past tense are different from the present tense. These are considered as syntactic variations of the same root form. This may present as a problem for a retrieval system because a relevant document

may contain a variation of a query word but not the exact word itself. This problem can be partially dealt with by stemming.

Stemming is a technique used to reduce the words to their stems or grammatical roots. This kind of technique is useful in the form of retrieving data or information and applications of data mining.

A stem is the portion of a word that is left after removing its prefixes and suffixes. In English, most variants of a word are generated by the introduction of suffixes (rather than prefixes). Thus, stemming in English usually means suffix removal, or stripping. For example, "computer", "computing", and "compute" are reduced to "comput". "walks", "walking", "walked" and "walker" are reduced to "walk". The stemming process is applied to remove suffixes such as "ed", "ing", "ily". By removing this suffixes we can reduce the diversity of terms in a document. This not only reduces the size of the document but also reduces its complexity.

To perform stemming we used *PorterStemmer* from *NLTK library*, as the dataset is mostly composed by English web sites.

Here we present an example of a processed [URL](#) considering two types of page categories. The reader may notice that the biggest difference between them is the number of prices inside the pages, with the product page having a much smaller number.

- Example [URL](#):

```
www costco com product number1 html
```

- Example of **catalog**:

```
url badmonday co uk collect hat cap titl hat bag accessori bad
monday apparel span new sale item ad click here span search span
search span close menu span close menu span menu span menu span
search span search span search span search span close menu span
close menu span hat bag accessori span grid view span grid view
span list view span list view span price1 span price1 span
price1 span price1 span price2 span price2 span price2 span
price2 span price2 span price2 span price3 span price3 span
price4 span price4 span price3 span price3 span sold out span
price5 span price5 span sold out span price5 span price5 span
price5 span price5 span sold out span price6 span price6 span
sold out span price6 span price6 span sold out span price6 span
price6 span sold out span price6 span price6 span sold out span
price6 span price6 span price7 span price7 span price7 span
price7 span this websit use cooki ensur receiv best experi find
use cooki learn more
```


- Example of **product**:

```
url badmonday co uk collect tee product break rule tee titl break
the rule tee bad monday apparel meta break rule tee our new
unisex style tee s come regular style fit also thicker longer
last this enabl eco friend ink last longer wash without fade all
tee s print uk front print regular fit tee design one design
discount1 comb ri span new sale item ad click here span search
span search span close menu span close menu span menu span menu
span search span search span search span search span close menu
span close menu span break the rule tee span price1 span price2
span size chart span add cart span discount1 comb ring spun
cotton span pleas see t shirt size span long line span chest size
half chest pleas doubl measur make full chest span this websit
use cooki ensur receiv best experi find use cooki learn more
```

4. String Tokenize

Each word can be represented as a token, being easier to identify and represent words relative to plain text.

The preprocessing phase can be divided into 2 stages: Text normalization and Text Processing.

Resumed in Table 1, we show the total number of different words (**tokens**) in our dataset by the selected parser that uses the tags selected in the begin of Subsection 3.3.1.

#	Parser	Text Normalized	Text Processing
1	URL	11.775	10.137
2	URL + Title	17.321	14.379
3	URL + Title + Meta	72.013	62.166
4	URL + Title + Meta + Span	83.293	74.352
5	URL + Title + Meta + Anchor	117.820	103.733
6	URL + Title + Meta + Span + Anchor	122.306	100.425
7	URL + Title + Meta + Span + Anchor + Headers	168.938	145.716
8	URL + Body	215.941	174.873

Table 1.: Different word count found in preprocessing stage for the entire training set

3.3.2 Structural Context

Although web pages text content is a useful feature, as stated throughout Subsection 3.3.1, in some web pages these features are sometimes missing, misleading, or unrecognizable for various reasons, e.g., the use of large images or flash objects but little textual content.

In such cases, it is difficult for classifiers to make reasonable judgments based on textual features from the page. However, from the Web page information standpoint, other types of connections can also be derived; and some of them have been proved useful for web page classification (e.g., Sara-Meshkizadeh and Rahmani (2010)).

In structural content preprocessing, the information will follow the same process presented in Subsection 3.3.1, however, instead of saving the textual information in the tags we implement a counter for the number of selected tags in that Web page. The idea is to test if these counters can be representative features of a genre, and a possible solution for the WPC.

The choosing of the HTML tags is based in 3 main methods:

- Input methods (user interaction);
- Emphasized information;
- Format and visual representation of the information;
- Information type;

We started by using the preselected tags from Subsection 3.3.1. However, some of the information they presented, for the scope of this solution, did not seem relevant (i.e., Meta-data will not be displayed on the page and the Title tag is required in all HTML documents). From this baseline we added new tags, posteriorly comparing them to the ones presented in our dataset and eliminated those that were not representative (do not appear in relevant quantity) or presented redundant information for the WPC. In Table 2 this selection can be seen.

#	Tag	Function
1	a	Defines a hyperlink
2	h1	Defines HTML headings
3	h2	Defines HTML headings
4	h3	Defines HTML headings
5	h4	Defines HTML headings
6	h5	Defines HTML headings
7	h6	Defines HTML headings
8	strong	Defines important text
9	b	Defines bold text
10	em	Defines emphasized text
11	button	Defines a clickable button
12	select	Defines a drop-down list
13	input	Defines an input control
14	form	Defines an HTML form for user input
15	img	Defines an image
16	table	Defines a table
17	th	Defines a header cell in a table
18	tr	Defines a row in a table
19	ul	Defines an unordered list
20	li	Defines a list item
21	div	Defines a section in a document
22	section	Defines sections in a document (e.g., chapters, headers, footers)
23	span	Defines groups of inline-elements in a document.
24	audio	Defines sound, such as music or other audio streams.
25	video	Defines videos, such as a movie clip or other video streams.

Table 2.: Chosen HTML tags

An example of a page content expressed by these counters:

```
{'a': 301, 'h1': 0, 'h2': 1, 'h3': 24, 'h4': 5, 'h5': 0, 'h6': 24, 'strong': 165, 'b': 0, 'em': 0, 'button': 0, 'select': 0, 'input': 7, 'form': 0, 'img': 135, 'table': 0, 'th': 0, 'tr': 0, 'ul': 11, 'li': 101, 'div': 575, 'section': 4, 'span': 21, 'audio': 0, 'video': 0}
```

3.3.3 Visual Context

Another approach selected is using visual information on a Web page. We choose to use the full page as an input, however, we recognize there are other ways to have a visual-based approach, such as the download of all the images from the page and the posterior classification, having those as an input (López-Sánchez et al. (2017)). Having in consideration Figures 15a, 15b and 57, we recognize some traits that are common in the different genres and cross E-commerce Web sites. For example, the presence of a big central image and various small elements in a listing approach are common features of both a product page and a catalog page, respectively.

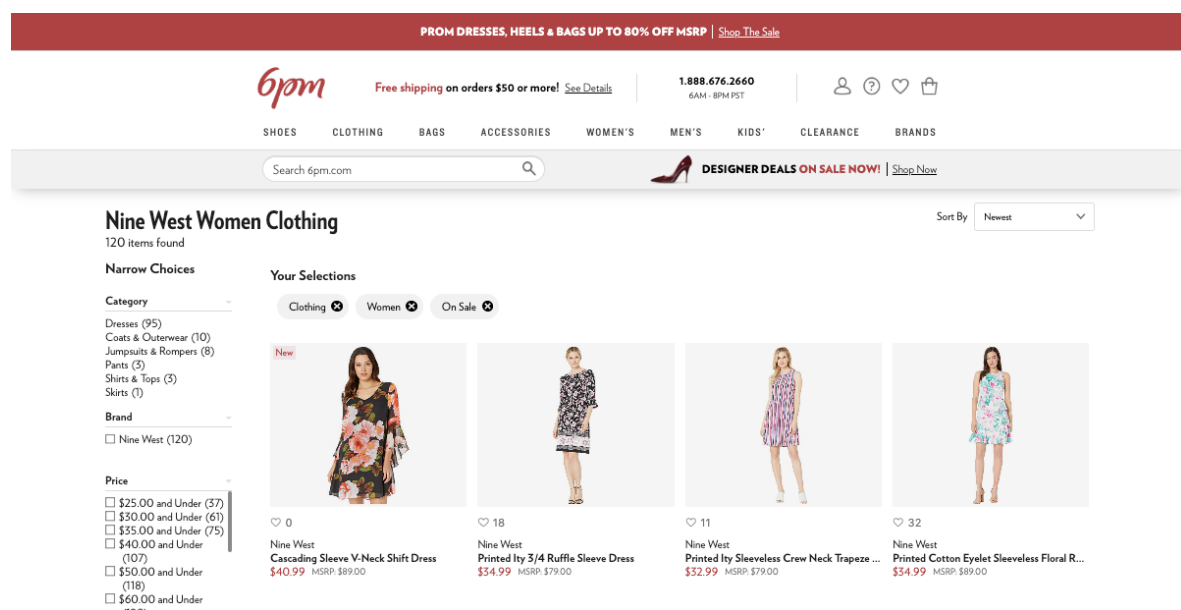
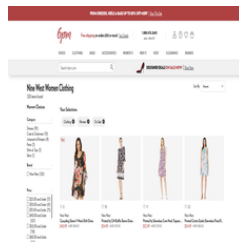


Figure 32.: Full-size driver print of the page.

As these web site snapshots are to be fed to a classifier we must decide on an input image size. A model trained on small images will learn fewer features than one trained on large images, the ones that it does learn should be the most important. Thus, a model architecture based on small images should be more generic. Using lesser features small-image models are **faster to train**.

We resized the Web page image in two different sizes presented in Figure 33 (values used in related research):



(a) Figure 32 resized to a 224x224 image



(b) Figure 32 resized to a 64x64 image

Figure 33.: Other image sizes

3.4 WEB PAGE CLASSIFICATION

To address the classification problem we have implemented multiple approaches that can be divided into 3 main types:

- Non-Machine Learning-oriented :
 - Document comparison:
 - * TF-IDF
 - * LSI
- Machine Learning-oriented
 - BoW
 - NB
 - SVM
- Deep Learning-oriented
 - RNN
 - Dense
 - CNN

Initially, we built a dataset for analysis with 7.000 Web pages as training samples. The input for those modelling techniques also change from **textual and/or structural** inputs to the **visual** aspect of a Web page. This WPC grasps into subject-based classification (also known as topic-based classification), Web pages are classified according to their contents or subjects, in our case that can be defined as the usage of the Spam class *vs* the Product/Catalog being the subject in focus the not commerce *vs* commerce. Alternatively, we can affirm that the problem also rests on genre-related factors, such as the purpose of the page. In that sense, the Catalog (List of Products) and a Product page itself have different functionalities, the same maybe even said about the pages presented in the Spam category (e.g. "FAQ", "About us").

3.4.1 Modeling WPC techniques

- **BoW**

Following the examples given in Subsection 3.3.1 this approach dwell in a count of price words, i.e., count of words that are related to the price value.

- Spam: $\text{price}[\text{index}] = 0$
- Product: $0 > \text{price}[\text{index}] \leq 2$
- Catalog: $\text{price}[\text{index}] > 2$

- **Document comparison**

Follows the approach presented in Subsection 2.2.2 where we first represent the text in two different ways: **TF-IDF** and **LSI**. New documents will be directly compared to the gathered database. Being chosen the class of the document that presents a bigger similarity to the evaluated one.

- **ML approaches**

Following the **Textual context ML** systems were also deployed, in our case we based our models in two main types:

- **NB** and **SVM**: Using the *sklearn library*
- **RNN**: This approach is mainly supported in **LSTM** layers; An example of this model may be found in Figure 58.

Structural context

- This approach is mainly supported in **dense** layers; An example of this model may be found in Figure 59.

Visual context

- **CNN**: This approach is mainly supported in **convolutional** layers; An example of this model may be found in Figure 60. Other **CNN** architectures (e.g., inceptionV3, alexnet Das (accessed July 17, 2019)) were also tested.

3.4.2 Building WPC classifier models

For the **ML** cases we initiate other stage of parameter settings, here we adjust parameters chosen for the training of the models.

1. **Number variation in training epochs**

An (*epoch*) is a full passage of all the dataset. Usually, an **ANN** is trained until an acceptable error, and this is normally achieved after several passes are made through the complete data set. For our case study we defined multiple values:

- 25;
- 50;
- 75;
- 100;

These values served as a shared point to all tests performed.

2. Variation of *batch size*

The term *batch size* defines the number of samples that will be propagated through the network until the weights of the neurons are updated. Changing this parameter to reduced values has some advantages like requiring less memory. Because network training uses fewer sample numbers, the overall training procedure requires less memory. Typically, networks are also faster with a smaller number of *batches*. On the other hand, increasing the values of this parameter causes the neural network to have more time and more cases to learn something before updating its weights, which can be very useful for large *datasets*. The selected **batch sizes** were as follows:

- 32;
- 64;
- 128;
- 256;

3. Variation in the number of layers

By increasing the number of layers it would be possible to give a greater learning ability to the network, something important for larger sets of files in which there is a greater variety of words to predict and more patterns to learn. Thus, for the different type of neural networks we induced changes in the model varied in a maximum of 3 inductions of the combination to the original model:

- **RNN**
 - LSTM layer;
 - dropout layer;
- **Dense**
 - dense layer;
 - batch normalization
 - dropout layer;
- **CNN**
 - convolutional layer;

- convolutional layer;
- max Pooling layer;
- dropout layer;

4. Activation functions

Two types of activation functions were used:

- **ReLU**: It is the most commonly used activation function in neural networks, especially in **CNNs**.
- *softmax*: It was used in the last dense layer and is used to solve the multi-classification problems.

5. Optimizer type

Optimization algorithms help to minimize (or maximize) a *loss* function, which is a mathematical function dependent on the internal parameters that result from the learning process of the model. Several optimizers were tested in the training process, such as:

- *rmsprop*: is a gradient method that is usually a good choice for recursive neural networks and for relatively small batch sizes;
- *adam*: this is a stochastic optimization method that starts from a relatively large learning rate that gradually decreases as needed to ensure convergence of the algorithm, being essentially a *rmsprop* with *momentum*;
- *nadam*: consists of the *adam* optimizer with Nesterov *momentum*.

Thanks to the flexible architecture of **TensorFlow**, our experiments were run on NVIDIA GeForce GTX 1070 GPU to accelerate the training process. After intensive and thorough testing, the best results came starting at 50 epochs, and beyond that point, the winnings were slightly better, until it hit a barrier at 75 epochs, where they stopped increasing beyond that number. After trying numerous batch sizes (i.e., 32, 64, 128, 256), the one with better results for our problem in specific is 64 for all the approaches. Regarding the number of layers, the structural approach needs only 2 layers, but the same does not apply to the image and text approaches, because these 2 have better results with an extra layer, this could be due to the size of the input in these two last approaches. The usage of more complex architectures (i.e.inceptionV3, alexnet) in the visual approach did not give us any performance gains. Regarding the optimizers, as a rule, adam has a good performance compared with other adaptive algorithms.

3.4.3 Designing tests for WPC

For studying the models quality and validity we created test designs that divide the train data (training and validation set) and test data in a way that will allow the performance and study of the following cases, not introducing any bias into the data, i.e., making a model that is perfect for one dataset, but no other. The training data is composed of diverse Web pages in English from several E-commerce Web sites of different product domains. We use *holdout* data, data that is not used during the model training process, for the tests.

- Same E-commerce Web sites, new Web pages: Is usual for a E-commerce Web site to be established in diverse markets (different countries) with that as stated in Section 3.1. Thus, in this case another important study is performed:
 - Different culture (**language**);
- Same product domain, different E-commerce Web sites: the interest in this specific test consists in testing the impact that products similarity may have in the classification process.
- Different product domain, different E-commerce Web sites: From a more generic approach, through this test we try to understand if the evaluation model is dependent from the type of product, or if it can be independent from those factors finding common points in genre classification inside the E-commerce domain.

3.5 SCRAPER

In the application development, we created two types of extraction methods for the **HTML** information. The first one was using an **HTTP** Request from the **Python** library **urllib.request**. However, some websites use *JavaScript* to load the information to the page, so the information gathered by the request that we do will not have all the information necessary to crawl to other links or even classify that page. An example of such a site is **Pri** (accessed July 17, 2019).

Therefore, to deal with lazy loading pages (i.e., the loading of only partial sections of the Web page delaying the remaining until it is needed by the user), we created another method using a tool named **chrome drive**, basically in the same way that is used in **ao Mário Gonçalves da Costa (2014)**, with this we copy the behavior of a *browser* load the full page and only after that we download its information, allowing us to give a more educated guess about the genre of page.

The problem with this later approach is the required time, being the extraction of the information a major bottleneck. The render of the page proved to be 10 times slower than using a simple request as may be seen in Figures 34 and 35. Nevertheless, there are some

```

Setting up Categorizer using links
Using links provided by spam.ini , catalgo.ini & product.ini
Spam
100%|██████████████████████████████████████| 16/16 [00:14<00:00, 1.33it/s]
Catalog
100%|██████████████████████████████████████| 16/16 [00:14<00:00, 1.34it/s]
Product
100%|██████████████████████████████████████| 16/16 [00:15<00:00, 1.15it/s]
Found 16 spamDocs & 16 catalogDocs & 16 productDocs

```

Figure 34.: Load training set directly from the web (method 1) - time

```

Setting up Categorizer using links
Using links provided by spam.ini , catalgo.ini & product.ini
Spam
100%|██████████████████████████████████████| 16/16 [03:19<00:00, 12.30s/it]
Catalog
100%|██████████████████████████████████████| 16/16 [03:45<00:00, 12.03s/it]
Product
100%|██████████████████████████████████████| 16/16 [03:22<00:00, 12.29s/it]
Found 16 spamDocs & 16 catalogDocs & 16 productDocs

```

Figure 35.: Load training set directly from the web (method 2) - time

cases that even using [35](#) to load a page does not translate in the presence of all information, for these cases, an extra step may be required. This step can be the as simple as a **scroll down** to load more content, or the **movement of the mouse** to a specific region of the web page such as the menu symbol. This will have its toll on a performance standpoint.

However, it is by using this tool and the **Selenium** framework that we will be able to provoke actions over the page that will have an impact over the gathered information.

As shown in [Figure 36](#) the information feeding the scraper are the output values from the Product class gathered by the crawler. Nevertheless, the scraper may act as a standalone application if the user wishes to preselect some products of higher interest for its search or already has its product dataset created.

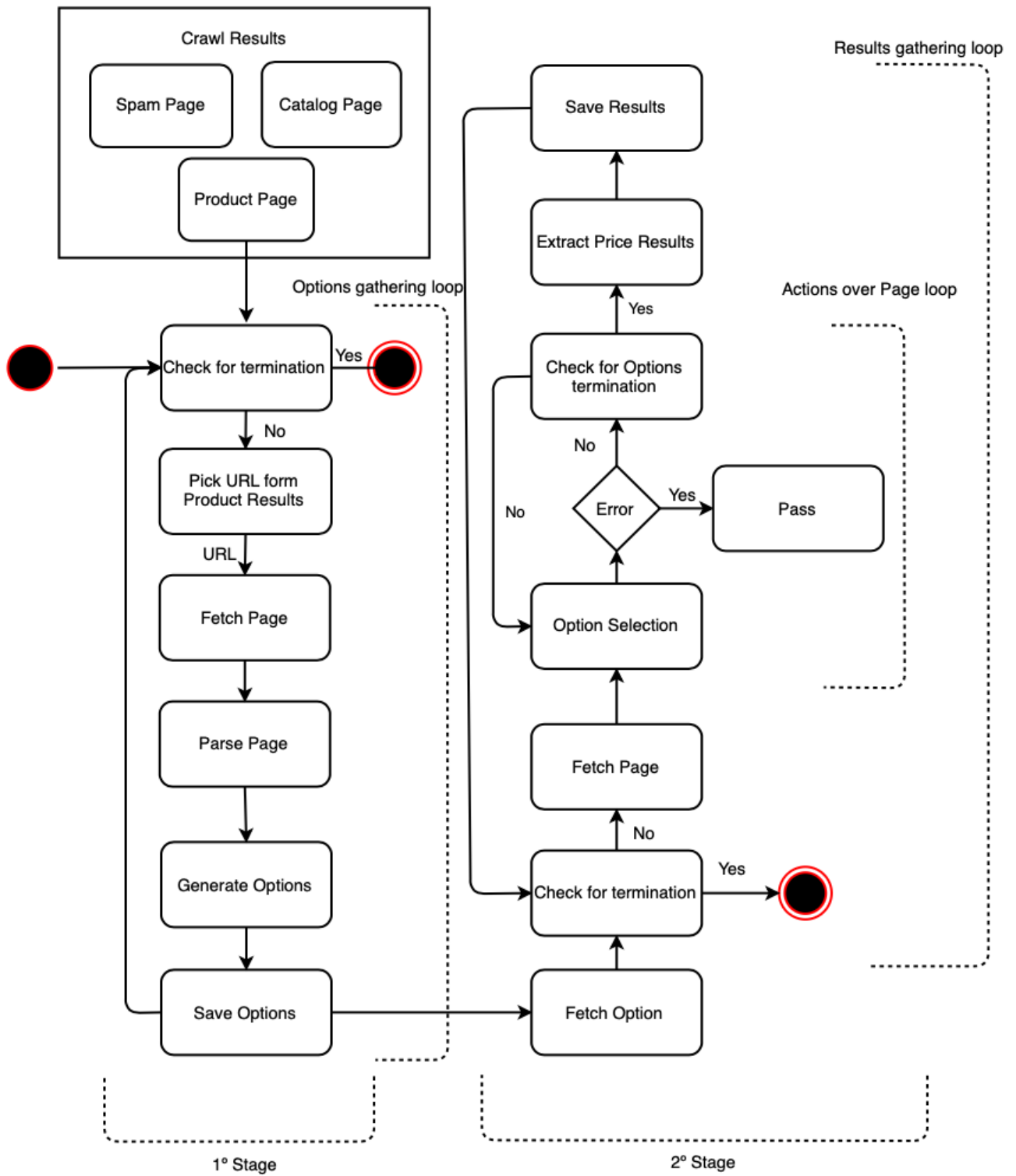



Figure 36.: Flow Chart Scraper

In Subsection 2.3 we introduced the extraction of information from a web page. Our proposed solution is similar to the one approached in [Cenys and Grigalis \(2014\)](#). However, we introduced a new step to the process where we found a connection on the presence of 3 types of input/selection methods and the specifications of the products. Those methods are:

- Dropdown (Figure 37);



(a) Example of dropdown selection

```

<div class="quantity">
  <div class="quantity-label more-info-icon-label">
    ::before
    <span>Quantity</span>
    ::after
  </div>
  <div class="quantity-select js-quantity-select">
    <span class="stylized-select-container ">
      ::before
      <select class="stylized-select" name="quantitySelect">
        <option value="100">100</option>
        <option value="250">250</option>
        <option value="500" selected="selected">500 Recommended</option>
        <option value="1000">1000</option>
        <option value="1500">1500</option>
        <option value="2000">2000</option>
        <option value="2500">2500</option>
        <option value="5000">5000</option>
        <option value="10000">10000</option>
      </select>
    </span>
    <div class="quantity-minimum" style="display: none;">...</div>
  </div>
</div>

```

(b) Example of dropdown HTML representation

Figure 37.: Dropdown example

- Buttons (Figure 38);



(a) Example of buttons selection

```

<div class="variant-attribute-label more-info-icon-label">
  ::before
  <span class="variant-attribute-name">Paper stock</span>
  ::after
</div>
<div class="variant-attribute-select js-variant-attribute-select">
  <div class="js-configurator-error configurator-error configurator-error-complex">...</div>
  <div class="option-set option-set-skin-buttons-wide" name="Attribute-233857">
    ::before
    <div class="js-variant-attribute-choice option-set-option-wrapper js-default-choice" data-choice-id="238821" data-choice-name="Matte" data-sku for="238821" aria-checked="false">
      <span class="js-variant-attribute-choice-radio option-set-option stylized-radio">
        <input class="stylized-radio-input" id="238821" name="Attribute-233857" type="radio" value="238821">
        <label for="238821">...</label>
      </span>
      <div class="option-set-contents">
        Matte
      </div>
    </div>
    <div class="js-variant-attribute-choice option-set-option-wrapper" data-choice-id="238822" data-choice-name="Glossy" data-sku for="238822" aria-checked="false">
      <span class="js-variant-attribute-choice-radio option-set-option stylized-radio">
        <div class="option-set-contents">
          Glossy
        </div>
      </span>
    </div>
  </div>
</div>

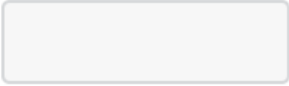
```

(b) Example of buttons HTML representation

Figure 38.: Buttons example

- Input boxes (Figure 39);

Digite seu CEP



(a) Example of input box selection

```

▼<div class="mol-settings-delivery-method">
  <h4>Digite seu CEP</h4>
  ▼<div class="mol-settings-delivery-method-content">
    ▼<div class="app_zipcode_validation">
      ▼<div class="app_zipcode_input">
        ▼<div class="app_input_masked">
          <input type="text" id="matrix-zipcode-input" value>
        </div>
      </div>
    </div>
  </div>
</div>

```

(b) Example of input box [HTML](#) representation

Figure 39.: Input box example

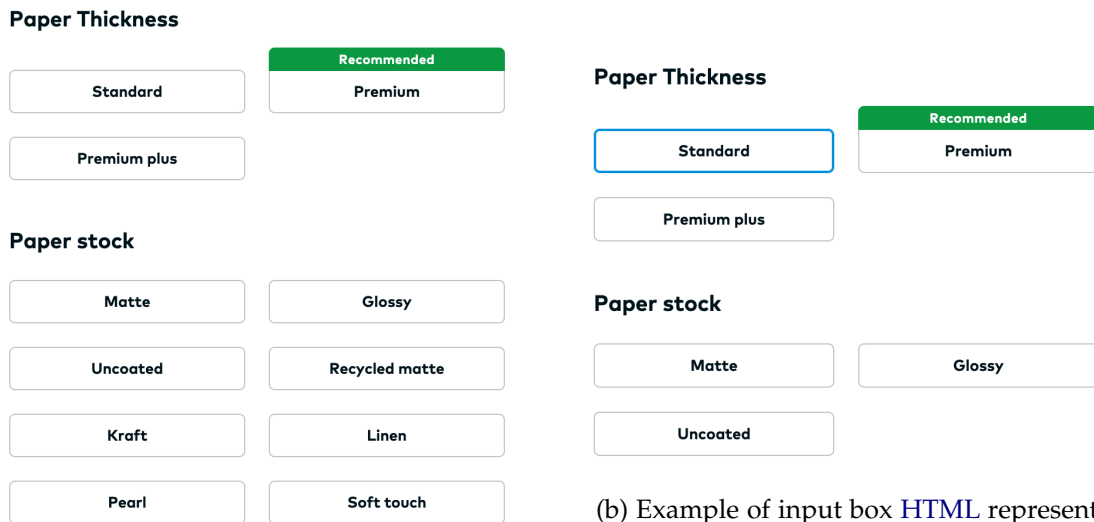
In general, these input methods will be used to produce some type of change inside the product Web page and consequently the information the page presents.

Our approach to the subject consists of an algorithm that:

- Finds inside the [HTML](#) these input methods;
- Travels in the [DOM](#) tree (ancestors, i.e., "..") until it finds the attribute characteristic (in text), the number of hops necessary is saved and used in the extraction of other attributes. In Figure 38a "Paper stock" is the attribute being the text in the buttons (e.g., "Mate", "Glossy") the values.
- Generates all the possible combinations of the values collected from a determinate product attribute;
- Saves this information in an *excel* page;

The actions over the options of a Web page may generate results such as:

- The removal of all options that are incompatible with the antecedent (Figure 40);

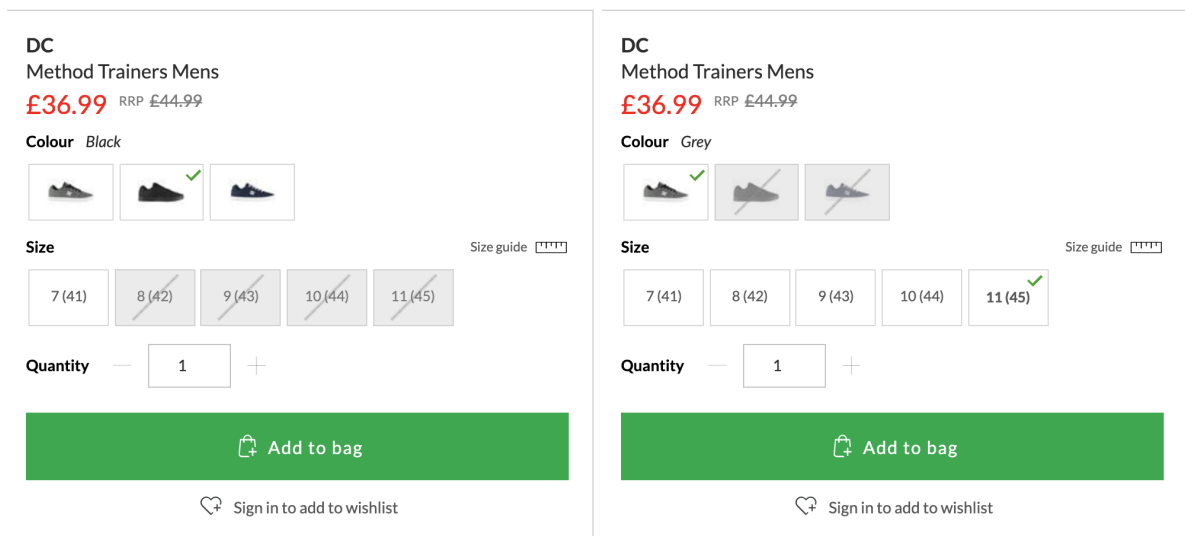


(a) Default values

(b) Example of input box HTML representation

Figure 40.: Options removal

- The options remain dormant on the page but a selection triggers an action on the page that may be the retraction to a previous option (Figure 41).



(a) Selection of colour "Black"

(b) Selection of Size "11(45)" in colour "Black"

Figure 41.: Dormant options

- An option is no longer an input option and no action can be taken over it (Figure 42).

COLOUR




Figure 42.: Non input option

One product page may generate thousands or even **millions** of possibilities (Figure 43). The analyses of such a large number of values may present an overwhelming problem. Thus, a previous pre-selection of the region of interest inside the universe of possibilities could be integrated into the solution, i.e, a custom made approach for some products.

Notepads ★★★★★ 4.8 (139) [Write a review](#)

- ✓ Lively colors on premium paper
- ✓ Ready to ship in 1 business day
- ✓ 25, 50, or 100 sheets per pad
- ✓ Custom sizes available



CONFIGURE & PRICE

Size:

Paper Type:

Printed Side:

Padding:

Glued Edge:

Sheets Per Pad:

Number of Pads:

Printing Time:

- ✓ 10
- 20
- 40
- 50
- 60
- 80
- 100
- 200
- 300
- 400
- 500
- 600
- 800
- 900
- 1,000
- 1,200
- 1,400
- 1,500
- 1,600
- 1,800
- 2,000
- 2,200
- 2,400
- 2,600
- 2,800
- 3,000
- 3,200
- 3,400
- 3,600
- 4,000

Figure 43.: Marketing product E-commerce Web page

Another issue is when the input is an **input box**. In this case, the integration of the input requires the user providing beforehand the text for these fields.

Besides that, the functionality of this gathering is also dependent on the Web site. Despite the major examples studied allowed to use this approach there are also other input methods in Web sites and are not englobed in our solution, an example may be seen in Figure 44.

For these exceptions, a custom approach needs to be done (e.g., Figure 44a do the *cartesian product* for the two options and treat them like one).

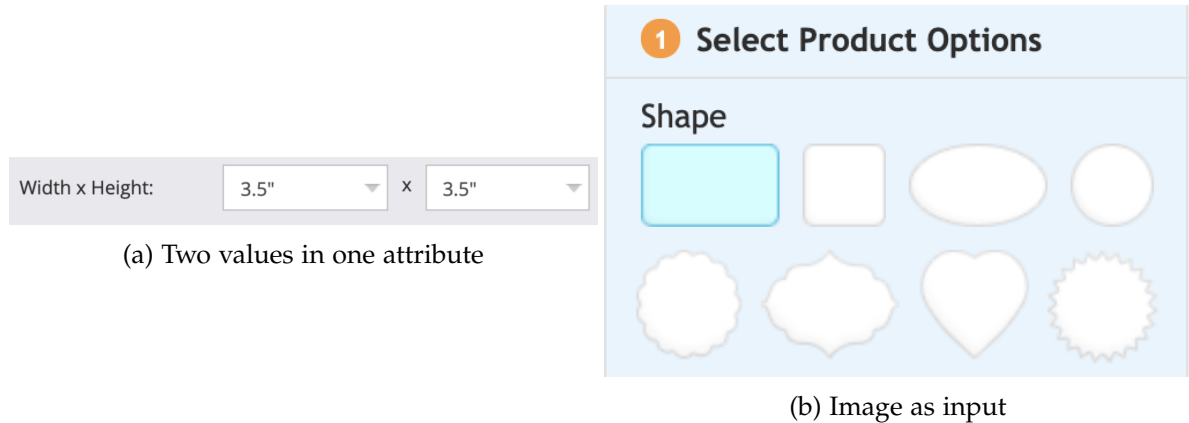


Figure 44.: Non regular options

Before proceeding to the price extraction, we have to confirm that the options that are selected are correct (i.e. custom check per site what options have been checked Figure 45). This takes particular importance in cases like Figure 41 where the non-verification can produce wrong price results. If the option no longer exists (Figure 40) or is not a valid input option (Figure 42) *Selenium* itself confirms the error in the action.

Paper stock



(a) Checked option

```
><div class="js-variant-attribute-choice option-set-option-wrapper js-default-choice checked" data-choice-id="238821" data-choice-name="Matte" data-sku for="238821" aria-checked="false">_</div>
><div class="js-variant-attribute-choice option-set-option-wrapper" data-choice-id="238822" data-choice-name="Glossy" data-sku for="238822" aria-checked="false">_</div>
```

(b) Checked option HTML example - in red

Figure 45.: Check option selection

3.6 CONTRIBUTION REVIEW

In this chapter we describe a crawler which seeks, acquires, indexes, and maintains pages on a specific set of genres (Spam, Catalog, and Product) that represent a relatively narrow segment of the Web. For the Product genre, we created a scraper to deal with the dynamic elements on the page and promote a more cohesive extraction of the information directly connected to the product price.

It requires a small investment in hardware and network resources and yet achieves respectable coverage and results. Thus, we believe that Web content can be managed by a distributed team of focused crawlers and scrapers.

Here are some compelling examples of what was achieved with this work:

- **Dataset for genre classification:** Most existing Web page classification datasets are either too small or obsolete. Using the presented tool we created a large-scale dataset (the training set plus new examples) with approximately **11.000** Web pages from roughly around **100** E-commerce Web sites. This dataset may serve as a baseline to future datasets.
- **HTML tags importance:** **HTML** tags play an important role in Web page classification. Not every tag contributes equally to the genre of a target Web page. To solve this issue, we designed a **HTML**-like textual format with pre-selected tags to be the inputs of our models. As a result, models learn the accurate impacts of different **HTML** tags to genre classification.
- **Visual and Structural context in genre classification:** Some Web pages offer too little textual content or contain too much noise. For that, we study the impact in the use of structural elements without any textual content and the visual look of the rendered page as a possible solution to **WPC** in E-commerce genres.
- **Automation in data extraction on E-commerce Product pages:** Usual extraction in Web pages is preformed over stale data. We introduce actions over the page before proceeding to data extraction. The tests performed are solid proof that some actions over E-commerce domains may be usually automated and a plus in the quantity and quality of the data extracted.
- **Discovering linkage sociology:** The crawl results from one E-commerce Web site of other E-shop shows a relationship between those two different competitors (e.g., the crawl in *Vistaprint* Web site showed a connection to *Printi* Web site).

CASE STUDIES / EXPERIMENTS

This chapter covers the last two phases of the **CRISP-DM** project methodology (Chapman et al. (2000)).

- **Evaluation:** This stage analysis results, it assesses the degree to which the model meets the business objectives, and seeks to determine if there is some business reason why this model is deficient. It compares results with the evaluation criteria defined at the start of the project.
- **Deployment:** This phase is also covered, although partially. That is, in this last phase, the algorithm is extended to operate in near real-time, continuously extracting the latest data from a real concurrent, but still in an experimental environment.

4.1 EVALUATION OF WPC AND SCRAPER

In this section, the performance of the models is compared regarding the different encodings with static data presented in Section 3.3, for the textual approach. The selected numbering is corresponding to the Table 1:

1. URL;
2. URL + Title;
3. URL + Title + Meta;
4. URL + Title + Meta + Span;
5. URL + Title + Meta + Anchor;
6. URL + Title + Meta + Span + Anchor;
7. URL + Title + Meta + Span + Anchor + Headers;
8. URL + Body

A summary of the results obtained is given in Table 3. For the ones we consider more relevant (in green) a more detailed interpretation is given using the precision, recall and f-score metrics seen in Figure 46 (source - Tables: 4, 5 and 6).

Encoding	Textual						Structural	Visual
	BoW	TF-IDF	LSI	NB	SVM	RNN	DENSE	CNN
1	-	0.59	0.61	0.73	0.63	0.76	-	-
2	-	0.55	0.56	0.73	0.53	0.79	-	-
3	-	0.54	0.53	0.76	0.62	0.79	-	-
4	0.69	0.63	0.62	0.73	0.69	0.84	-	-
5	-	0.64	0.67	0.76	0.64	0.92	-	-
6	-	0.62	0.64	0.74	0.62	0.96	-	-
7	-	0.59	0.59	0.74	0.62	0.96	-	-
8	-	0.51	0.47	0.71	0.55	0.82	-	-
*	-	-	-	-	-	-	0.88	-
224*224	-	-	-	-	-	-	-	0.83
64*64	-	-	-	-	-	-	-	0.85

Table 3.: Accuracy results with static data-sets (best values in **green**).

Accounting for the crawl solution design (Figure 25, the class with the more impact if misclassified is the **Catalog**).

In cases of False Negatives (i.e., a Catalog example classified as a Product or a Spam page), the overall solution will lose all the links to the products of that catalog page.

In the case of False Positive (i.e., a Product or a Spam page example classified as a Catalog page), the model will overload pages to classify, since it is by the catalog that URLs are added to the crawling frontier.

For these cases, **False positives** and **False negatives**, we study the precision and recall respectively.

Looking at Figure 46 we may see that the **RNN** approach that uses textual input has the best performance in terms of precision, in the recall the **CNN** that uses the visual input has the worst results.

Studying the final F-score (that use both precision and recall), for the Catalog class, the results follow the same path of the global accuracy, the **RNN** is the one that has best results. To the other classes, the **RNN** keeps its superiority.

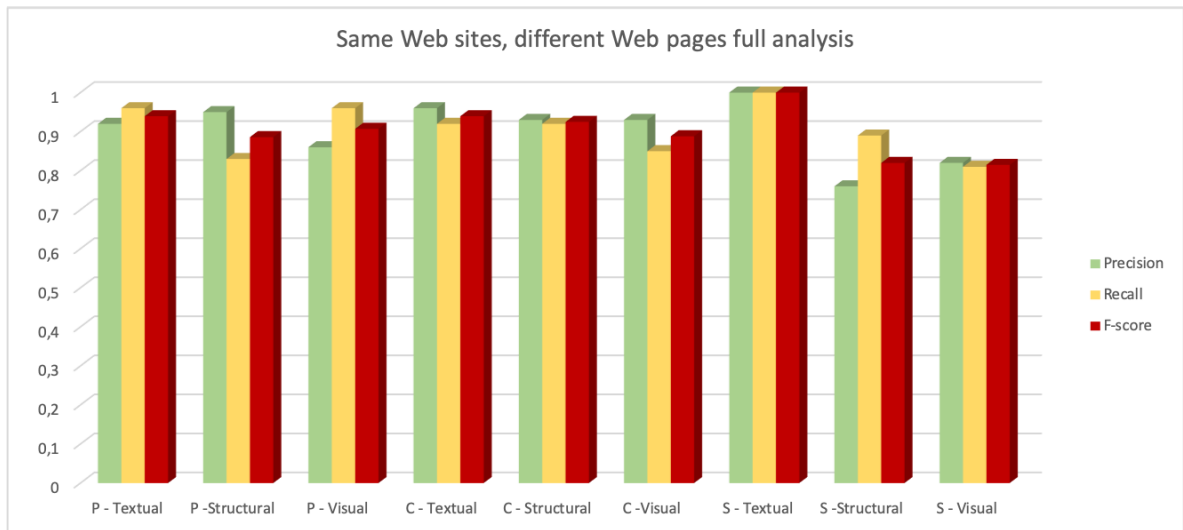


Figure 46.: Same Web sites, different Web pages analysis; P - Product; C - Catalog; S - Spam

4.1.1 Textual

An analysis of Table 3 may help us draw some inferences about the value of each classification method and input. First, ML methods do achieve better accuracy results than other models.

In particular for the BoW case we realize that a simple rule can achieve acceptable results and that the reasons for the performance not being better can be imputed to the fact that the Web is highly unnormalized in terms of structure and content-wise. A single rule is too narrow to differentiate all genres. Therefore, more rules and the creation of a decision tree building like approach should achieve even better results. The BoW approach is based on the detection of information inside the *Span* tags (i.e., URL + Title + Meta + Span), as stated in Section 3.3, price information is normally contained inside the *Span* tags, in our dataset roughly 90% of the E-commerce web pages analyzed present their prices inside these tags. Thus, the study on other *textual encodings* was **not** considered to the model accuracy.

This simple rule can not surpass problems such as:

1. Pages do not present the price information inside the selected tags;
2. Product pages present information about other products inside the page (i.e., catalog inside a Product page);
3. Catalog pages that only present one product;
4. Spam page present information about flash deals;

5. Product page presents multiple prices for a product considering different options.

For the URL analysis, encoding 1, we detect two main issues both in training and test set:

- The URL does not have any type of structure, thus, inference over this data with plausible results is not feasible.
- In our case some Web sites have "products" as a representation for their Catalog pages, after the *stemming* this word is presented as "product", information common in Product pages.

However, in general, the analysis of the URL proves to be important information for the WPC problem. In particular, when the URL presents useful information in its semantic creation, the model can infer the correct class, i.e., the usage of special words like:

- **Product** class: e.g., "product", "p";
- **Catalog** class: e.g., "catalog", "categorie", "collection", "c", "k";
- **Spam** class: e.g., postfixes "edu", "ac" indicate web pages belong to university or academic web sites, or directory names, such as "FAQ", "forum", "blog", and hostnames "instagram", "youtube", "facebook";

We also believe that preprocessing of the numbers inside the URL to "number" + index had a positive influence in the classification process.

This classification is the one in which the difference between using the NB and RNN is one of the smallest, i.e., a difference of 3%, we believe that this difference caused by the smaller amount of different words in the dataset as can be seen in Table 1 and the smaller size of each input.

The bad results of *stemming* over the model are not unique to URLs. It can likewise hurt precision because many irrelevant documents may be considered relevant. For example, both "productive" and "products" are reduced to the stem "product". However, if one is looking for documents about products, a document that contains only productive is unlikely to be relevant.

In the document similarity approach (TF-IDF and LSI) we use weight word-based features in the classification tasks. TF-IDF is the term frequency of a feature is multiplied by its IDF score. However, this study shows that IDF weights are not a good option for domain-specific datasets because they favor rare features. This is particularly noticeable for genre classification tasks when detecting that relevant words (e.g., "buy", "add", "bag", price information) do not have the required score.

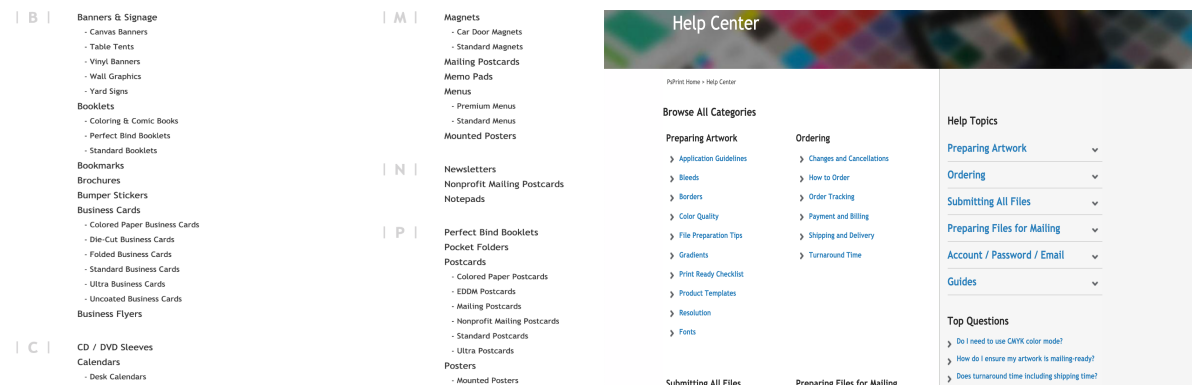
Additionally, the specific name of some products words tends to have very-low-frequency counts in any given document because page owners, especially distinct brands, often add linguistic variety to their product names, resulting in lowering TF scores. Thus, comparing the same product from different E-commerce web sites may become even more difficult.

We believe that the main problem associated with document comparison is the fact that the classification model sees each document as an individual element. Hence, one element misclassified or with features shared between classes has a huge impact on the final classification process.

To the ML most simple approaches, the NB model presented better results than the SVM. Directly comparing these approaches with the RNN we can see that in all tests the RNN presents better results. One of the reasons that we believe to be responsible for this is the multiplicity of types of E-commerce Web sites that the model was trained on. While the RNN can abstract and still perform well with multiple Web sites the NB does not, and introducing new models to train will harm the overall performance.

4.1.2 Structural

Using a different methodology for the WPC based in the structural elements of the Web pages but using the same test domain and training set it is possible to observe that the approach lacks accuracy comparably to the RNN. However, it does perform better than all the other classifiers. The misclassification may be justified by the reuse of Web pages templates, where despite presenting different subjects the templates are too similar (Figure 47), thus, the classifier has trouble distinguishing between them.



(a) Catalog page example;

(b) Spam page - misclassified as Catalog page;

Figure 47.: Misclassified examples - Structural approach

4.1.3 Visual

Analyzing Table 3 is possible to see that, considering the sizes tested, the size of the image does not have any influence on the classification accuracy. However, similar to the **Structure** approach the **Visual** falls in the same problems (Figure 48).

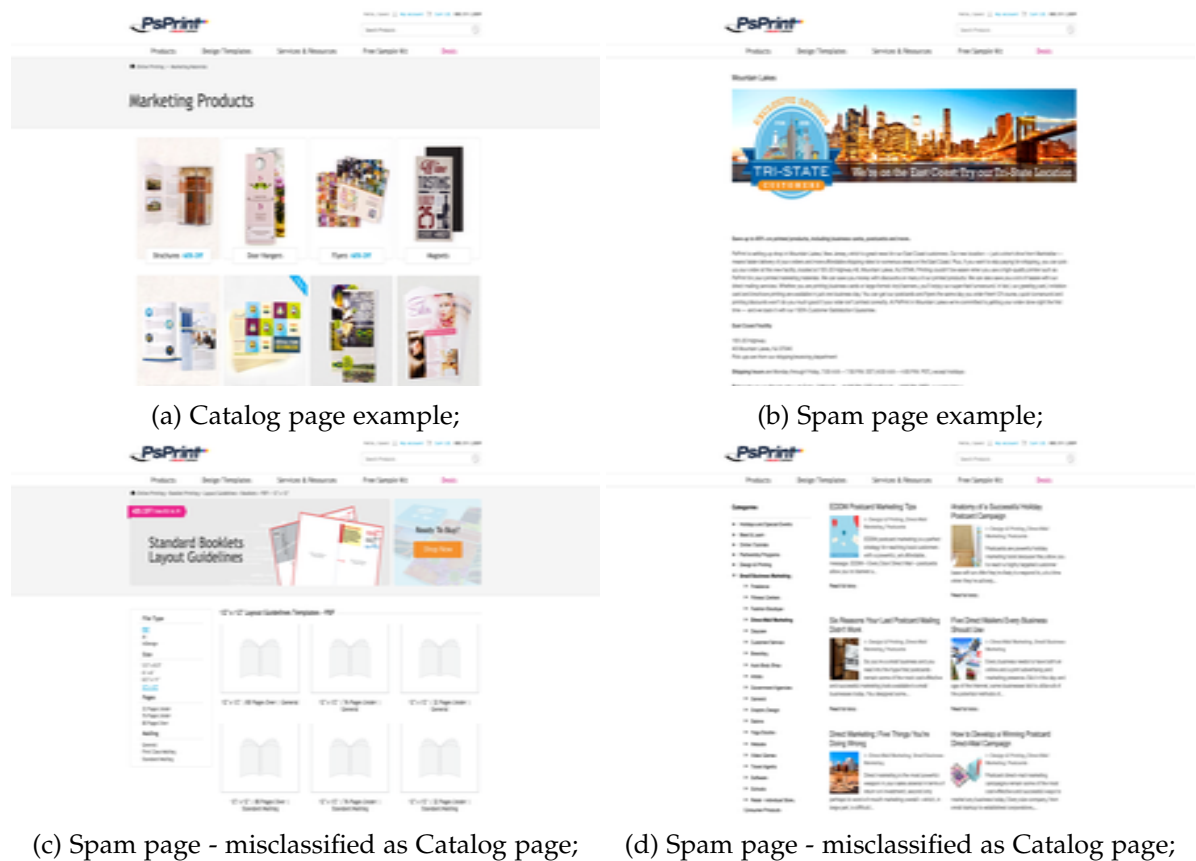


Figure 48.: Misclassified examples - Visual approach

4.1.4 Full tests report

Until now, the reports over results were from new Web pages from the E-commerce Web sites present in the training set.

Other tests were audit over the specified conditions in Subsection 3.4.3, those are:

1. New web pages from the the E-commerce Web sites present in the training set (Section 4.1);
2. New web pages from the the E-commerce Web sites present in the training set in other markets (e.g., "br", "fr");
3. New web pages of the same product domain (e.g., clothing) from different E-commerce Web sites from the training set;
4. New web pages of the different product domain (e.g., electronics) from different E-commerce Web sites from the training set;

The full overall accuracy report over the tests is shown in Figure 49.

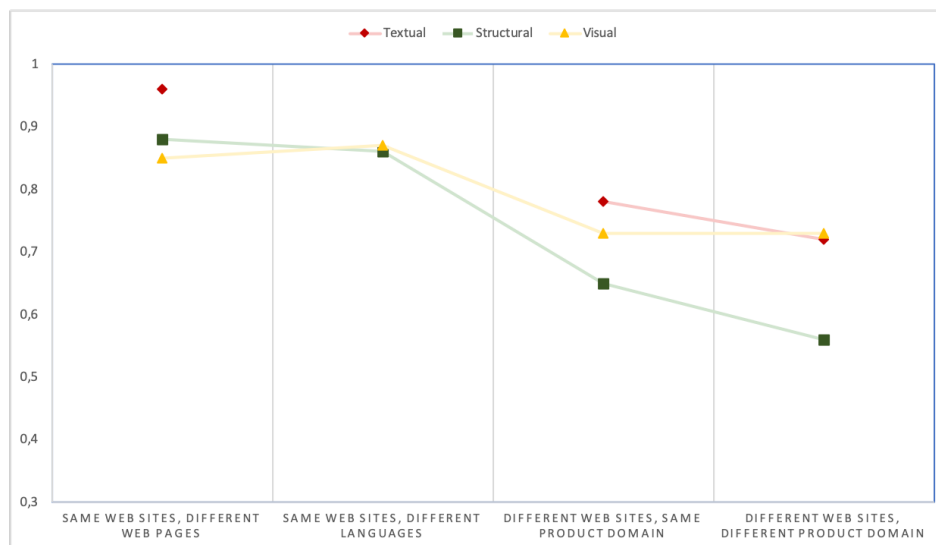
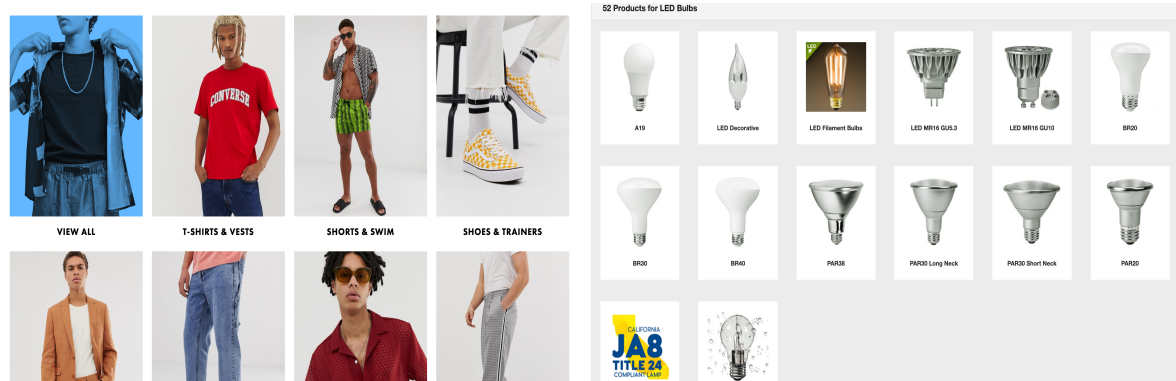


Figure 49.: Different design tests

The Web is too denormalized to get stable results for all types of E-commerce sites, although there is a superior affinity between web sites that act on the same domain of products (e.g., in our case, the big majority of the dataset is from E-commerce clothing web sites, in these cases, even when there are no functional data such as the number of prices, textual models can overcome this problem because the pages have similar information between them). However, when we enter more technical product domains (electronics, marketing)

there is a substantial growth of the terms (i.e., each E-commerce Web site gives a specific name to the product) losing the capability to generalize, an example may be seen in Figure 50.



(a) Catalog page correctly classified in all models (b) Catalog page misclassified in the RNN

Figure 50.: Tests in new Web sites in the same product domain (clothing) and in a different product domain (electronics - lamps)

Regarding the research of new languages inside the same Web site types, the usage of models that do not depend on the textual content show themselves generic enough to produce results similar to the examples of the pages that have been previously trained, i.e., the models are truly generic inside the trained E-commerce Web site and, except in cases where the change of culture involves the creation of a completely different web site, these templates are generics enough to maintain their accuracy in web pages of different languages (worst results in spam pages where each E-commerce Web site tend to have more diverse templates).

The same happens with E-commerce Web sites that were not present in the training set, some, that own highly specific construction, need to be treated as original and inserted examples on the training dataset, to approach a more reliable result and be used in a real case scenario.

The usage of structural content presents better results than those expected. Inside a Web site, Web developers tend to follow a certain creation metric, hence, pages of the same type tend to possess a high number of similarities, however, the extrapolation to other E-commerce pages proves to be less effective. For those cases, the use of the page visual content proves to be particularly efficient.

4.2 DEPLOYMENT

In this section, we review both the crawler and scraper in real-world scenarios. We dwell in the problems of the presented solution and ways to avoid it. For the crawler, we present the final classification results of the crawled web pages. For the scraper, we present a partial solution from one product page saved by the crawler.

4.2.1 Crawler

Following our proposed approach, the links that were processed and classified are saved in 3 types of documents, each one assigned to its respective genre.

In a real case scenario, giving [Upr \(accessed July 17, 2019b\)](#) as a seed page and the [RNN](#) model as the brain of the crawler, an E-commerce web site of marketing merchandise present in the training set, produce the crawl final main results.

However, for this case, as an **exception** from other E-commerce sites, the *sitemap.xml* (Subsection 2.1.8), already gives us the products, i.e, the Webmaster already sorted the Web pages. In their *sitemap.xml* ([Upr \(accessed July 17, 2019a\)](#)) web pages with a priority of **0.6** are **Product** pages, being presented in there **376** products. Comparing to our solution that only found **260** there is a huge gap (**31%**).

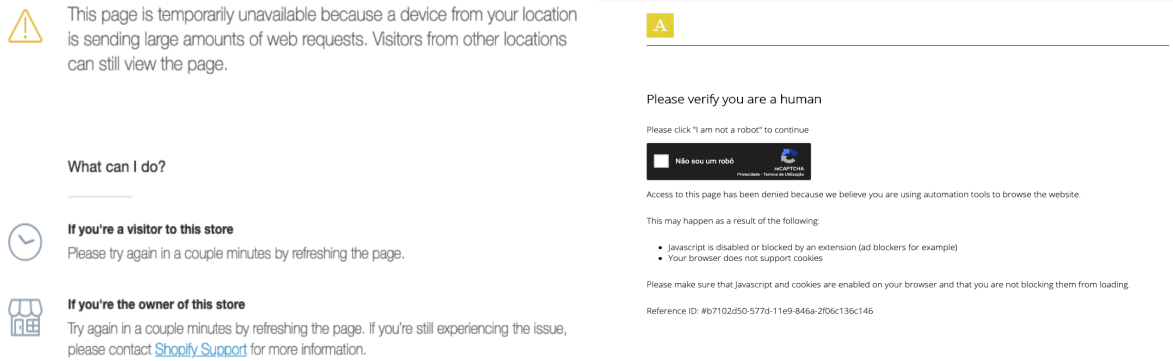
In terms of precision, the textual features model for this type of test (same web site from the training set) was 92% to the Product pages class (Figure 46). In this case, we have a better result, all product classified are in fact products (**100%**). However, other 23 product URLs were present in the main results, and **all misclassified as Spam** affecting the recall (was 96% to 92%). The Catalog class also had examples misclassified as Spam pages. Hence, it is necessary to understand these results and the problems associated with them. Initially, the crawler main results were labeled correctly, however, abruptly the crawler started to classify all pages as Spam pages.

As stated in 3.2, we follow the [CREAD](#), and that decision will have consequences for the type of application that we are developing (Figure 51).

403 Forbidden

Figure 51.: Crawl blocked - message I

As both stages (*crawler* and *scraper*) need to access information in the page a constant connection and send of requests is required, thus, a number of sites did **block** our *crawl* and *scraping* as can be seen in Figure 52.



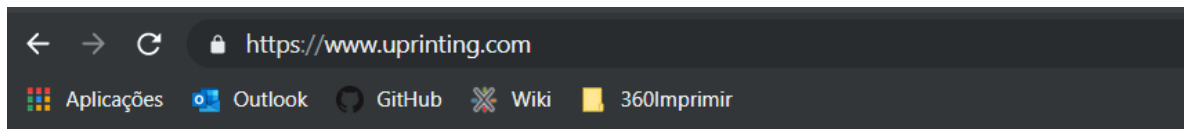
(a) Crawl blocked - message II

(b) Crawl blocked - message III

Figure 52.: Blocked crawl examples

During crawling they also blocked our extraction as can be seen in Figure 53.

The case of **Uprinting** was not an exception and that raises a very serious issue regarding the extraction of information from the web.



403 ERROR

The request could not be satisfied.

Request blocked.

Generated by cloudfront (CloudFront)
Request ID: Ee8Z0LPBVMoeJJADFW2_QB0jXQ1gqIT3biP6AM3sB4c_eb8KdN_gMg==

Figure 53.: Uprinting block

In a business perspective, this problem has a huge toll on the integrity and functionality of the application.

Using **CREAD** is **not** mandatory, hence it is possible to cheat the system by:

- Presenting themselves as a browser an not a crawler;

- Randomize time of access to pages;
- Randomize navigational patterns (change crawl itinerary);
- Ignore the Robots.txt;
- Use a [VPN](#) to camouflage our [IP](#) address

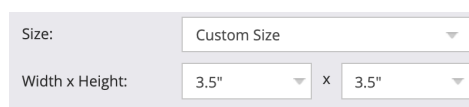
This issue is caused by a large number of page requests in a short period of time. To avoid this issue we can use a smaller scope, for example, a search for only "business cards". This filtering from the product results as originated by a simple word match between the title from the page and the selected words, in this case, "business cards". Despite being a bit archaic, for this test case, and for other tests generated using the same principle the results proven to avoid the issue in the majority of cases.

4.2.2 *Scraper*

In Subsection 3.5 we explain how we gather the attribute/value options present for each product.

The posterior test of these combinations will always be dependent on the bottleneck associated with the rendering of the page. Thus, the existence of a product with **thousands** or **millions** of combinations will prove inefficient in the creation of an application that, from a business perspective, should give real-time values. The presentation of the information to the user in a readable way shows to be also a problem.

An example of this is Figure 43 here we only use the first present attributes in the page, those, for **one** product gives us **25.201** possible combinations. The introduction of the new attributes that appear on the page after selection of certain attribute/value combination presented in Figure 54 will increase those options to **63.204.108** for only **one** product Web page.



The image shows a web form with two rows of input fields. The first row is labeled 'Size:' and has a dropdown menu currently showing 'Custom Size'. The second row is labeled 'Width x Height:' and has two dropdown menus, each showing '3.5"', separated by an 'x' character.

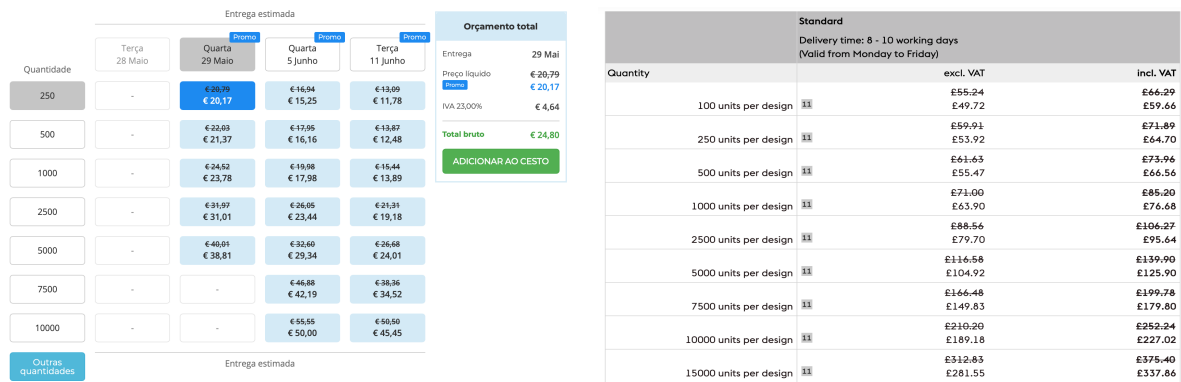
Figure 54.: Extra attributes from Figure 43 Web page - Non default

In these cases, we decided to not add these options to the result list and use only the default values present on the page. However, the selection of the attribute/value may be directly dependent on the last attribute/value option chosen. For those cases, custom integration of the options needs to be designed.

After the extraction of the options information present on the page, the actions for the gathering the **price information** are started. We used **Selenium** to perform actions over the

Web page (i.e., click, select, write input) and automating the extraction process as shown in Figure 36.

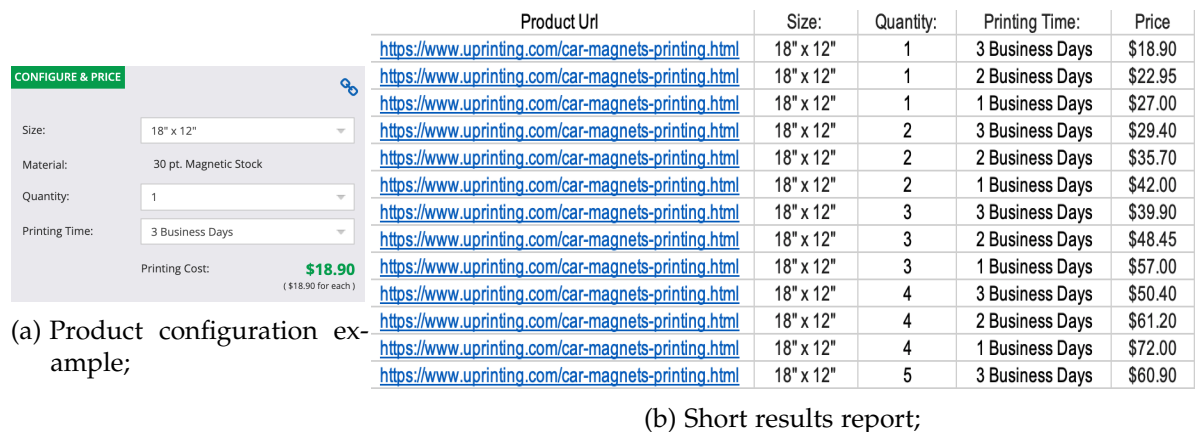
The final extraction of the price value is performed by the use of a **regex** over tags that present price information (usually *Span* tag as state in Subsection 3.3.1). When more than one result is presented over one Web page a pattern matching approach is done with that specific Web site in mind (Subsection 2.3.1), in Figure 55 we may see examples where this action was required.



(a) Product page price representation example I (b) Product page price representation example II

Figure 55.: Product page price representation

In Figure 56 we have an example of the gathering of prices in an E-commerce Web site from a product (Upr (accessed July 25, 2019)), the total number of prices gathered for this product were 600 values excluding the "Custom Size" options (i.e., using the default values). Extra information may be added to each Web site solution (e.g., unit price).



(a) Product configuration example;

(b) Short results report;

Figure 56.: Product scrap results

For the cases where the number of options is too high the most interest attribute/values options from a specific product should be preselected by the user. This way, the under-

standing of the values should be easier and the time required to do the selection of those values and extraction of the price information from a specific product should decrease.

4.3 SOLUTION OVERVIEW

This chapter is divided into two stages, the study of the tests designed in Section 3.1, and the validation of these results (crawler and scraper) in a real-world scenario.

In the first stage, tests over multiple approaches based on the textual, structural and visual features of the Web pages. In particular, on the textual content of a Web page, various approaches to its classification were tested, being the use of a RNN the one that had best outcomes.

The results show that the presence of textual elements is crucial for the proper functioning of the WPC. However, for completely unknown product domains the visual elements of the page prove to be a valuable asset.

In the second stage, we performed a real case study, *UPrinting*, one of the E-commerce Web sites used in our training set.

This case shows that following the CREAD has its consequences on the solution, being the control in the Web site side, however, that control can be circumvented. On the pages that weren't blocked the system maintained its accuracy percentage, solidifying the solution. For the scraper, the actions automation using a test framework over the page provided the intended results. Static data provides stale values. In a real-world scenario, dynamic results must be integrated for having complete records about products, hence, viable information for decision making.

However, issues about the time spent in the crawler (in Web sites with thousands of products) and the scraper (in Web pages with thousands of options) and even the display of the results is a concern to the functionality of the application.

CONCLUSION

5.1 SUMMARY AND DISCUSSION

In this project, we presented an approach for the automatic web page classification by genre in three types (i.e., Catalog, Product and Spam) using the combination of structural, text and structure, and visual features. We also described an approach to extract information of interest (i.e., product description and prices) from product web pages. This integration shows that it is possible to create a generic framework for E-commerce Product pages, however, the quality and viability of the approach depends on the Product domain and E-commerce Web site dimension.

The automatic genre classification was primarily focused on extracting textual features from Web pages. These features were studied in multiple approaches (i.e, [BoW](#), document similarity ([TF-IDF](#) and [LSI](#)) and machine learning algorithms ([NB](#), [SVM](#) and [ANN](#))).

In other approaches, we studied the [HTML](#) and visual features of Web pages that capture the layout of the genres and tested the [WPC](#) process using this information separately.

The usage of textual content proved particularly useful in the classification between Product/Catalog classes against the Spam class. We believe that the textual content for this type of "subject-based" classification is the more suited, as other approaches are too generic.

The analysis of textual data is a needed feature to guarantee the good performance of the classifier in known and unknown E-commerce Web sites.

Our approach categorizes the web page by genre based on text features from the Web page [URL](#) merged with specific tags and their enclosed text. From the preselected tags not all share the same relevance, we concluded in [Table 3](#) that the usage of Anchor and Span tags is essential for the quality of results. Adding further information did not provide any improvement. The introduction of new E-commerce web sites from different product domains harms the classification process, i.e, more product domains tend to decrease the accuracy. In particular, for approaches like document comparison, [NB](#) and [SVM](#), the increase in the number of terms presented for classification show to have a negative impact on the classification process. The only one that proved to be able to deal with the high term count was the [RNN](#).

Directly comparing the structural and visual models to the textual model (Figure 49) we can see that the presence of text features are relevant, as the design of pages tries to transmit information to the user by various document formatting (e.g., listings) the information present in those formats is not always relevant and this approaches seem to lack the capacity to understand that. Thus, textual content information about the topic/subject is necessary for the good performance of the WPC classifier and particularly for E-commerce Web sites in the same type of product domain.

Merely thinking of the integrity of the solution, some considerations are necessary. Even though the application has good behavior in most tested cases, the reality is that there are web sites that have the lazy loading of the pages very present in their design (i.e., content is only loaded when it exists some kind of action over the page). In these cases, the analysis and integration of specific steps for the information loading in the page are necessary, something that defies the application generality purpose.

Another problem is the crawler/scrapper duration time, from a business perspective, it is necessary to always be updated. For a medium-sized site considering only the crawler, it can take over a full day, which, in different situations (e.g., flash sales, Christmas sales) could prove to be a real problem.

The same is applied on the scraper, as we can see in Section 3.5 a simple product page, depending on the type of the product, could generate millions of combinations. Although for business, all this information may have value, both the presentation of results and the extraction of knowledge are issues. The analysis of a table where the results are being inserted is impractical and the collection of results itself for a product with millions of combinations, in an E-commerce Web Site which may have thousands of products, to a market with different competitors is not something that can be easily achieved with low computing resources.

5.2 FUTURE WORK

There is still an optimization problem to be solved regarding the process of selection of the best classifier to guide the crawler. The necessity to fully pass through a catalog page should not be the only criterion because it may be too narrow, and misclassification of a catalog limits the scraping of multiple products.

Another possible improvement is the design of an ensemble of classifiers, using all the information present in the Web page (textual, structural and visual) so that every classifier model can use its strong points and help each other; or a type of decision tree that first evaluates the information present on the page and that designs a classifier model to it.

Both the crawler and the scraper should pass from a sequential application to a parallel (multiple agents) application to reduce the time of crawl/scrape.

It is also crucial to explore methods of presenting the information to the user. With potentially millions of results, this seems to be an interesting data visualization problem that needs to be addressed for the system to achieve its full potential.

BIBLIOGRAPHY

- Ansam A AbdulHussien. Comparison of machine learning algorithms to classify web pages. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(11):205–209, 2017.
- Myriam Abramson and David Aha. What’s in a url? genre classification from urls, 2012. URL <https://www.aaai.org/ocs/index.php/WS/AAAIW12/paper/view/5252>.
- Akiko Aizawa. An information theoretic perspective of tf-idf measures. *Information Processing & Management*, 39:45–65, 01 2003. doi: 10.1016/S0306-4573(02)00021-3.
- Ioannis Anagnostopoulos, G Kouzas, Christos-Nikolaos Anagnostopoulos, T Kotsilieris, S Kalogeropoulos, Vassili Loumos, and E Kayafas. Classification of e-commerce web pages using statistical descriptor vectors. *Rivista di Statistica Applicata*, 14:91–107, 01 2002.
- Dr.M. Kannan and. COMPARISON OF SOFTWARE TESTING TOOLS WITH RESPECT TO TOOLS AND TECHNICAL RELATED PARAMETERS. *International Journal of Advanced Research in Computer Science*, 8(9):70–75, September 2017. doi: 10.26483/ijarcs.v8i9.4908. URL <https://doi.org/10.26483/ijarcs.v8i9.4908>.
- Jo ao Mário Gonçalves da Costa. *Web Page Classification using Text and Visual Features*. PhD thesis, University of Coimbra, 2014.
- J. Belfort. *The Wolf of Wall Street*. Number vol. 1 in *The Wolf of Wall Street*. Random House Publishing Group, 2007. ISBN 9780553904246. URL <https://books.google.pt/books?id=GLLCyNj-IUoC>.
- Catherine Benincasa, Adena Calden, E B Hanlon, Matthew Kindzerske, K. Law, E Lam, J. Rhoades, Ishani Roy, M. Leonardo Satz, E Gerald Valentine, and Nathaniel Whitaker. Page rank algorithm, 2006.
- James R. Bettman and C. Whan Park. Effects of prior knowledge and experience and phase of the choice process on consumer decision processes: A protocol analysis. *Journal of Consumer Research*, 7(3):234–248, 1980.
- Christopher Brooks. Web crawling as an ai project, 01 2008.
- Term frequency and weighting*. Cambridge University Press, accessed July 16, 2019. URL <https://nlp.stanford.edu/IR-book/html/htmledition/term-frequency-and-weighting-1.html#sec:secbagofwords>.

- Antanas Cenys and Tomas Grigalis. Unsupervised structured data extraction from template-generated web pages. *JOURNAL OF UNIVERSAL COMPUTER SCIENCE*, 20, 02 2014.
- P.P Chakrabarti and S Ghose. A general best first search algorithm in and/or graphs. *Journal of Algorithms*, 13(2):177 – 187, 1992. ISSN 0196-6774. doi: [https://doi.org/10.1016/0196-6774\(92\)90014-4](https://doi.org/10.1016/0196-6774(92)90014-4). URL <http://www.sciencedirect.com/science/article/pii/0196677492900144>.
- Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31:1623–1640, 04 2000. doi: 10.1016/S1389-1286(99)00052-3.
- Suriti Chakrabarti, Joonmyun Cho, and Hector Garcia-Molina. Analyzing different web crawling methods. *International Journal of Computer Applications*, 107(5), 2014.
- Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000. URL <https://maestria-datamining-2010.googlecode.com/svn-history/r282/trunk/dmct-teorica/tp1/CRISPWP-0800.pdf>.
- B. Choi and Z. Yao. *Web Page Classification**, pages 221–274. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-32393-8. doi: 10.1007/11362197-9. URL https://doi.org/10.1007/11362197_9.
- CamFind*. Cloudsight, Inc., accessed December 20, 2018. URL <https://itunes.apple.com/us/app/camfind/id595857716?mt=8>.
- Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Web mining: information and pattern discovery on the world wide web. *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, pages 558–567, 1997. ISSN 1082-3409. doi: 10.1109/TAI.1997.632303.
- Siddharth Das. Cnn architectures: Lenet, alexnet, vgg, googlenet, resnet and more ..., accessed July 17, 2019. URL <https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>.
- Classifying Websites with Neural Networks*. Datafiniti, accessed January 16, 2019). URL <https://blog.datafiniti.co/classifying-websites-with-neural-networks-39123a464055>.
- Datafiniti*. Datafiniti, accessed June 20, 2019. URL <https://datafiniti.co>.

- Viktor de Boer, Maarten van Someren, and Tiberiu Lupascu. Web page classification using image analysis features. In *Web Information Systems and Technologies - 6th International Conference, WEBIST 2010, Valencia, Spain, April 7-10, 2010, Revised Selected Papers*, pages 272–285, 2010. doi: 10.1007/978-3-642-22810-0_20. URL https://doi.org/10.1007/978-3-642-22810-0_20.
- Knowledge Graph, AI Web Data Extraction and Crawling*. DIFFBOT, accessed June 25, 2019. URL <https://www.diffbot.com/>.
- Bardia Doosti, David J. Crandall, and Norman Makoto Su. A deep study into the history of web design. In *Proceedings of the 2017 ACM on Web Science Conference, WebSci '17*, pages 329–338, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4896-6. doi: 10.1145/3091478.3091503. URL <http://doi.acm.org/10.1145/3091478.3091503>.
- Google Shopping*. Google, accessed December 23, 2018. URL <https://www.google.com/shopping?hl=en>.
- Peter Gurský, Vladimír Chabal, Robert Novotny, Michal Vasko, and Milan Verescák. Extracting product data from e-shops. In *ITAT*, 2014.
- Taher H. Haveliwala, Aristides Gionis, Dan Klein, and Piotr Indyk. Evaluating strategies for similarity search on the web. *Proceedings of the 11th International Conference on World Wide Web, WWW '02*, 04 2003. doi: 10.1145/511446.511502.
- A. Horch, H. Kett, and A. Weisbecker. Mining e-commerce data from e-shop websites. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 2, pages 153–160, Aug 2015. doi: 10.1109/Trustcom.2015.575.
- Mohammad Hossin and Sulaiman M.N. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5: 01–11, 03 2015. doi: 10.5121/ijdkp.2015.5201.
- Paul Huntington, David Nicholas, and Hamid R. Jamali. Web robot detection in the scholarly information environment. *Journal of Information Science*, 34(5):726–741, 2008. doi: 10.1177/0165551507087237. URL <https://doi.org/10.1177/0165551507087237>.
- Import.IO*. Import.io, accessed June 20, 2019. URL <https://www.import.io>.
- Total number of Websites*. Internet Live Stats, accessed March 22, 2019a. URL <http://www.internetlivestats.com/total-number-of-websites/>.
- Why External Web Data Is Getting Vastly More Valuable*. Internet Live Stats, accessed March 22, 2019b. URL <http://www.internetlivestats.com/internet-users/>.

- Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112. Association for Computational Linguistics, 2015. doi: 10.3115/v1/N15-1011. URL <http://aclweb.org/anthology/N15-1011>.
- Min-Yen Kan and Hoang Oanh Nguyen Thi. Fast webpage classification using url features. In *CIKM*, 2005.
- B.W. Kernighan and D.M. Ritchie. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2001.
- C. Kleissner. Data mining for the enterprise. *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, 1998a.
- Charly Kleissner. Data mining for the enterprise. In *HICSS*, 1998b.
- Ron Kohavi and Foster Provost. Applications of data mining to electronic commerce. *Data Min. Knowl. Discov.*, 5(1-2):5–10, January 2001. ISSN 1384-5810. doi: 10.1023/A:1009840925866. URL <https://doi.org/10.1023/A:1009840925866>.
- Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. *SIGKDD Explor. Newsl.*, 2(1):1–15, June 2000a. ISSN 1931-0145. doi: 10.1145/360402.360406. URL <http://doi.acm.org/10.1145/360402.360406>.
- Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. *SIGKDD Explor. Newsl.*, 2(1):1–15, June 2000b. ISSN 1931-0145. doi: 10.1145/360402.360406. URL <http://doi.acm.org/10.1145/360402.360406>.
- Martijn Koster. A standard for robot exclusion, 1994. URL "<http://nersp.nerdc.ufl.edu/~nemnm/infoseek/norobots.html>". accessed December 30, 2018.
- Milos Kovacevic, Michelangelo Diligenti, Marco Gori, and Veljko M. Milutinovic. Recognition of common areas in a web page using visual information: a possible application in a page classification. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 250–257, 2002. doi: 10.1109/ICDM.2002.1183910. URL <https://doi.org/10.1109/ICDM.2002.1183910>.
- Milos Kovacevic, Michelangelo Diligenti, Marco Gori, and Veljko Milutinovic. Visual adjacency multigraphs—a novel approach for a web page classification. In *Proceedings of SAWMo4 workshop, ECML2004*, 2004.
- Milos Kudelka, Vaclav Snasel, Zdenek Horak, and Ajith Abraham. Web site description based on genres and web design patterns. In *Proceedings of the 2009 International*

- Workshop on Social Informatics, SOCINFO '09*, pages 68–73, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3706-1. doi: 10.1109/SocInfo.2009.15. URL <http://dx.doi.org/10.1109/SocInfo.2009.15>.
- Rahul kumar, Anurag Jain, and Chetan Agrawal. Survey of web crawling algorithms. *Advances in Vision Computing: An International Journal*, 3(3):17, Sep 2016. doi: 10.5121/avc.2016.3301. URL <http://dx.doi.org/10.5121/avc.2016.3301>.
- R. Levering, M. Cutler, and L. Yu. Using visual features for fine-grained genre classification of web pages. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pages 131–131, Jan 2008. doi: 10.1109/HICSS.2008.488.
- P. Li, F. Zhao, Y. Li, and Z. Zhu. Law text classification using semi-supervised convolutional neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 309–313, June 2018. doi: 10.1109/CCDC.2018.8407150.
- SnapTell*. Lifewire, accessed December 21, 2018. URL <https://www.lifewire.com/snaptell-iphone-app-review-2000036>.
- Yuhui Lin. *RNN-Enhanced Deep Residual Neural Networks for Web Page Classification*. PhD thesis, CALGARY, ALBERTA, 2016.
- B. Liu, R. Grossman, and Yanhong Zhai. Mining web pages for data records. *IEEE Intelligent Systems*, 19(6):49–55, Nov 2004. ISSN 1541-1672. doi: 10.1109/MIS.2004.68.
- Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer Publishing Company, Incorporated, 2nd edition, 2011. ISBN 3642194591, 9783642194597.
- Daniel López-Sánchez, Juan M Corchado, and Angélica González Arrieta. A cbr system for image-based webpage classification: case representation with convolutional neural networks. In *The Thirtieth International Flairs Conference*, 2017.
- Hersovici M., Jacovi M., Maarek Y., Pelleg D., Shtalhaim M., and Ur S. The shark-search algorithm, 1998. URL <http://www.cs.cmu.edu/~dpelleg/bin/360.html>.
- Kate Maddox. Media power 50: Top b2b media properties keep innovating in a tough climate, accessed December 28, 2018. URL <https://adage.com/article/btob/media-power-50-top-b2b-media-properties-innovating-a-tough-climate/289548/>.
- Ng W.K. Madria S.K., Bhowmick S.S. and Lim E.P. Research issues in web data mining. *Data Warehousing and Knowledge Discovery DaWaK 1999. Lecture Notes in Computer Science*, 1676, 1999.

- Filippo Menczer. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology*, 00:378–419, 12 2004.
- George Moiseev. Classification of e-commerce websites by product categories. In *AIST*, 2016.
- Momondo. Momondo, accessed June 20, 2019. URL <https://www.momondo.pt>.
- Ajinkya More. Attribute extraction from product titles in ecommerce, 2016.
- mySimon. Comparison shopping - unique gift ideas - compare prices, accessed December 23, 2018. URL <https://www.mysimon.com/>.
- Aviral Nigam. Web crawling algorithms. *International Journal of Computer Science and Artificial Intelligence*, 4(3):63, 2014.
- W. Petprasit and S. Jaiyen. E-commerce web page classification based on automatic content extraction. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 74–77, July 2015. doi: 10.1109/JCSSE.2015.7219773.
- Paulo Pimenta. *KuantoKusta*, accessed December 23, 2018. URL <https://www.kuantokusta.pt/public/quem-somos->.
- Printi. *Do seu jeito: Mais que uma Gráfica Online*. Printi, accessed July 17, 2019. URL <https://www.printi.com.br/>.
- Osama Rababah and Fawaz Masoud. Key factors for developing a successful e-commerce website. *Communications of the IBIMA*, page 19, Jan 2010. doi: 10.5171/2010.763461. URL <http://dx.doi.org/10.5171/2010.763461>.
- Daniele Riboni. Feature selection for web page classification, 2002.
- Dmitri Roussinov, Kevin Crowston, Michael Nilan, Barbara Kwasnik, Jin Cai, and Xiaoyong Liu. Genre based navigation on the web. *Hawaii International Conference on System Sciences*, 4:4013, 01 2001. doi: 10.1109/HICSS.2001.926478.
- Sara-Meshkizadeh and Amir Masoud Rahmani. Webpage classification based on compound of using html features & url features and features of sibling pages. *Int. J. Adv. Comp. Techn.*, 2:36–46, 2010.
- A. M. Sarhan, G. M. Hamissa, and H. E. Elbehiry. Feature selection algorithms based on html tags importance. In *2015 Tenth International Conference on Computer Engineering Systems (ICCES)*, pages 185–190, Dec 2015. doi: 10.1109/ICCES.2015.7393043.
- Scrapinghub. Scrapinghub, accessed June 20, 2019. URL <https://scrapinghub.com/>.

- Browser Automation*. SeleniumHQ, accessed January 13, 2019. URL <https://www.seleniumhq.org>.
- Slyce*. Slyce, accessed December 21, 2018. URL <https://slyce.it>.
- T. Srivastava, P. Desikan, and V. Kumar. *Web Mining – Concepts, Applications and Research Directions*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-32393-8. doi: 10.1007/11362197_10. URL https://doi.org/10.1007/11362197_10.
- Search Engine Market Share Worldwide*. StatCounter Global Stats, accessed December 26, 2018. URL <http://gs.statcounter.com/search-engine-market-share>.
- Ali Stouky, Btissam Jaoujane, Rachid Daoudi, and Habiba Chaoui. Test automation tools: A comparison between selenium, jenkins and codeception. *International Journal of Engineering & Technology*, 7(3.7):367–370, 2018. ISSN 2227-524X. doi: 10.14419/ijet.v7i3.7.18880. URL <https://www.sciencepubco.com/index.php/ijet/article/view/18880>.
- Pang-Ning Tan and Vipin Kumar. Discovery of web robot sessions based on their navigation patterns. *Data Mining and Knowledge Discovery*, 6(1):935, 2002. doi: 10.1023/a:1013228602957. URL <http://dx.doi.org/10.1023/A:1013228602957>.
- Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. In *Data Classification: Algorithms and Applications*, 2014.
- Trivago*. Trivago, accessed June 20, 2019. URL <https://www.trivago.pt>.
- Trupti V. Udupure, Ravindra D. Kale, and Rajesh C. Dharmik. Study of web crawler and its different types. *IOSR Journal of Computer Engineering*, 16(1):0105, 2014. doi: 10.9790/0661-16160105. URL <http://dx.doi.org/10.9790/0661-16160105>.
- Uprinting sitemap*. Uprinting, accessed July 17, 2019a. URL <https://www.uprinting.com/sitemap.xml>.
- Uprinting*. Uprinting, accessed July 17, 2019b. URL www.uprinting.com.
- Car Magnets*. Uprinting, accessed July 25, 2019. URL <https://www.uprinting.com/car-magnets-printing.html>.
- Antonio Miguel Baptista Videira. *Web Page Classification using Visual Features*. PhD thesis, University of Coimbra, 2013.
- Dan Woods. Why external web data is getting vastly more valuable, accessed March 22, 2019. URL <https://www.forbes.com/sites/danwoods/2015/03/06/why-external-web-data-is-getting-vastly-more-valuable/#1c65e68312d3>.

Guandong Xu, Yanchun Zhang, and Lin Li. *Web Content Mining*, pages 71–87. Springer US, Boston, MA, 2011. ISBN 978-1-4419-7735-9. doi: 10.1007/978-1-4419-7735-9_4. URL https://doi.org/10.1007/978-1-4419-7735-9_4.

Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli: using GUI screenshots for search and automation. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, Victoria, BC, Canada, October 4-7, 2009*, pages 183–192, 2009. doi: 10.1145/1622176.1622213. URL <https://doi.org/10.1145/1622176.1622213>.

SPAM PAGES EXAMPLES

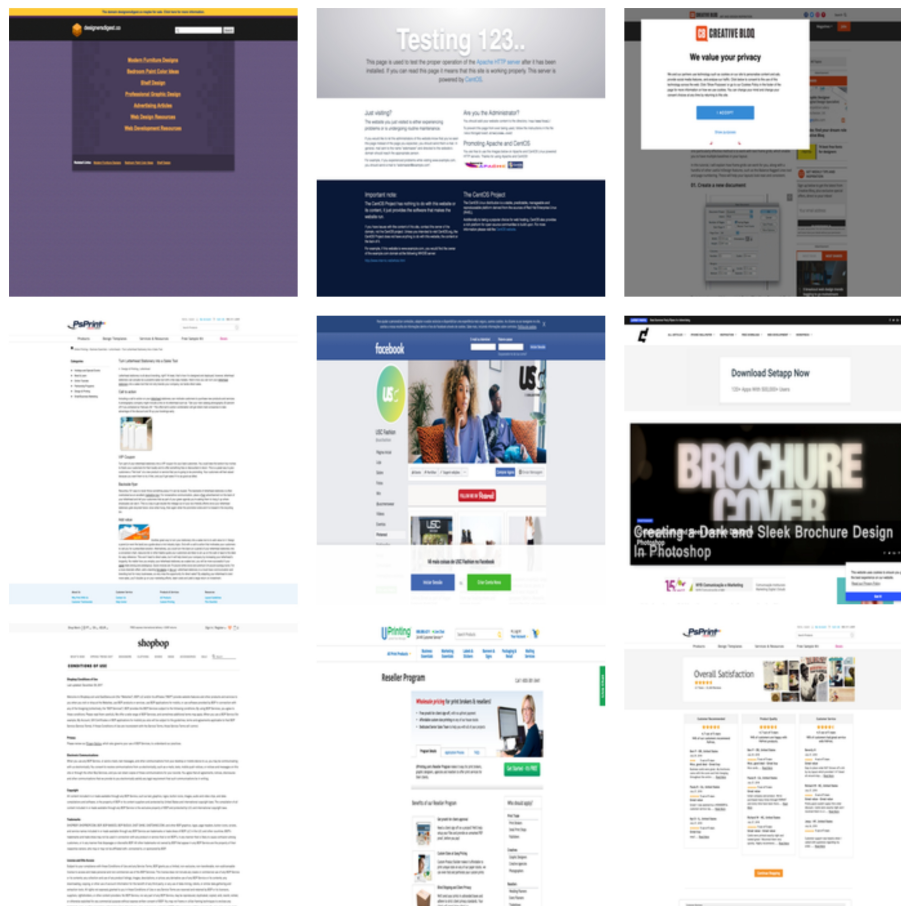


Figure 57.: Spam page examples.

B

NEURAL NETWORK ARCHITECTURE EXAMPLES

Layer (type)	Output Shape	Param #
inputs (InputLayer)	(None, 5500)	0
embedding_1 (Embedding)	(None, 5500, 50)	3000000
lstm_1 (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation_1 (Activation)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 3)	771
activation_2 (Activation)	(None, 3)	0

=====
Total params: 3,046,851
Trainable params: 3,046,851
Non-trainable params: 0

Figure 58.: Neural Network for textual evaluation

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1024)	43008
dropout_1 (Dropout)	(None, 1024)	0
batch_normalization_1 (Batch Normalization)	(None, 1024)	4096
dense_2 (Dense)	(None, 512)	524800
batch_normalization_2 (Batch Normalization)	(None, 512)	2048
dropout_2 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 32)	16416
batch_normalization_3 (Batch Normalization)	(None, 32)	128
dropout_3 (Dropout)	(None, 32)	0
dense_4 (Dense)	(None, 3)	99
=====		
Total params: 590,595		
Trainable params: 587,459		
Non-trainable params: 3,136		

Figure 59.: Neural Network for structural evaluation

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
conv2d_2 (Conv2D)	(None, 62, 62, 32)	9248
max_pooling2d_1 (MaxPooling2)	(None, 31, 31, 32)	0
dropout_1 (Dropout)	(None, 31, 31, 32)	0
conv2d_3 (Conv2D)	(None, 31, 31, 64)	18496
conv2d_4 (Conv2D)	(None, 29, 29, 64)	36928
max_pooling2d_2 (MaxPooling2)	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
conv2d_5 (Conv2D)	(None, 14, 14, 64)	36928
conv2d_6 (Conv2D)	(None, 12, 12, 64)	36928
max_pooling2d_3 (MaxPooling2)	(None, 6, 6, 64)	0
dropout_3 (Dropout)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 512)	1180160
dropout_4 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 3)	1539
=====		
Total params: 1,321,123		
Trainable params: 1,321,123		
Non-trainable params: 0		

Figure 60.: Neural Network for visual evaluation

SAME WEB SITES, DIFFERENT WEB PAGES TESTSET ANALYSIS

	Spam	Catalog	Product	Total
Spam	100	0	0	100
Catalog	0	96	4	100
Product	0	8	92	100
	100	104	96	300

Table 4.: Textual (RNN) model assessment - Test 1

	Spam	Catalog	Product	Total
Spam	76	8	16	100
Catalog	4	93	3	100
Product	5	0	95	100
	85	101	114	300

Table 5.: Structural (dense layers) model assessment - Test 1

	Spam	Catalog	Product	Total
Spam	82	15	3	100
Catalog	7	93	0	100
Product	12	2	86	100
	101	110	89	300

Table 6.: Visual CNN model assessment - Test 1

Dissertation elaborated on an internship under the orientation and funding of 360Imprimir in Braga Portugal.