Using Data Mining Techniques to Solve the Web Classification Problem in Real Scenarios

A thesis presented by

Samuel Jesús Martínez Eguiarte

Submitted to the School of Engineering and Sciences in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Monterrey, Nuevo León, December, 2019
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey
School of Engineering and Sciences

The committee members, hereby, certify that have read the thesis presented by Samuel Jesús Martínez Eguiarte and that it is fully adequate in scope and quality as a partial requirement for the degree of Master of Science in Computer Sciences.

Dr. Hugo Terashima Marín
Tecnológico de Monterrey
Principal Advisor

Dr. Alejandro Róales Pérez
Tecnológico de Monterrey
Co-advisor

Dr. Héctor Gibrán Ceballos Cancino
Tecnológico de Monterrey
Committee Member

Dr. Miguel Ángel Álvarez Carmona
CICESE
Committee Member

Dr. Rubén Morales Menéndez
Associate Dean of Graduate Studies
School of Engineering and Sciences

Monterrey, Nuevo León, December 9th, 2019
Declaration of Authorship

I, Samuel Jesús Martínez Eguiarte, declare that this dissertation titled, “Using Data Mining Techniques to Solve the Web Classification Problem in Real Scenarios” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

______________________________
Samuel Jesús Martínez Eguiarte
Monterrey Nuevo León December 9th, 2019

@2019 by Samuel Jesús Martínez Eguiarte
All rights reserved
I would like to dedicate this thesis to my God for giving me the opportunity and privilege to work in this exciting program. Thanks to my parents for helping me emotionally and financially this last two years. I would like to also express my gratitude to my sister Cristina Martínez. Without the help and motivation of them I would have never entered this program. Thanks to all my Professors and friends who were instrumental this last two years of my life.
I would like to express my deepest gratitude to my assessors Dr. Hugo Terashima Marín and Dr. Alejandro Rosales Pérez. Thank you for guiding me in the creation of this project. As well I would like to thank the Professors of Tecnológico de Monterrey for doing an excellent job in all their teachings and making the time to solve any inquires of mine. The last acknowledgments goes to Tecnológico de Monterrey for giving me the change of studying my masters and creating this project; and to CONACYT for supporting me these last two years.
Using Data Mining Techniques to Solve the Web Classification Problem in Real Scenarios

by
Samuel Jesús Martínez Eguiarte

Abstract

The following thesis describes an investigation that aims to solve the web classification problem with real scenarios. The main motivation of the project is to be able to fully automate the testing of different types of web pages. For this reason, a total of four classes were defined as: Login, Search, Form, and Article. These four classes represent basic testing scenarios.

The main goal is to correctly identify each example as one of these classes. A dataset was initially created containing 2,000 different examples (500 examples for each class). The dataset contains the tag elements of a web page as features. These tag elements show the frequency a tag appears in a specific web page.

Since this problem can be also viewed as a text classification problem, where every tag element represents a different word and a web page represents a different document, one goal of the project is to determine if the tag dataset could be used instead of the text dataset. For this reason, another dataset was created (using the same 2000 examples), in which the plain text of the web pages was extracted in order to apply text classification techniques. The classification models selected were: Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Multinomial Naive Bayes (MNB), and a simple Neural Network (NN). These models were also defined as One-class models in order to see if the models could be trained using positive data only. In the end, three experiments were performed. One using the One-class models with the tag dataset; other using the same tag dataset but with Multi-class models; and one last one using the text dataset with both One-class and Multi-class models.

To obtain all of these 2000 scenarios, a web-scraping tool was created to obtain both the tag frequency and the plain text of a web page. As it was mentioned previously, four classes were defined with 500 examples for each class. A baseline was created in order to compare the results of the models. This baseline uses an algorithm called Web Page Classification Algorithm Based on Feature Selection (WCAFS). The results of the baseline yielded a score of 0.88.

The One-class models do not show a score similar to the one established in the baseline model, except for one. The One-class Article SVM model, with the use of the tag dataset, had a score of 0.88. This was the only model, that uses the tag dataset, that achieved the same score as the baseline. The Multi-class models only achieved the same result as the baseline when the text dataset was applied. So, the best configuration to use is a Multi-class model with a TF-IDF (Terms Frequency - Inverse Document Frequency) transformation applied to a text dataset.
List of Figures

1.1 The rendering of a web page with dynamic content. ........................... 2
1.2 Cornell web page according to the WebKB dataset. ......................... 4
1.3 Current Cornell web page. .......................................................... 5
1.4 General diagram showing the three modules of the automation solution. .. 6

2.1 The three different categories of web mining. ................................. 12
2.2 The SVM function that separates the data points using a separator with a margin. 15
2.3 Basic example of transforming a text document into a word vector. ........ 20

3.1 Final dataset used in the experiments. The content column represents the text dataset that is going to be transformed using the text transformation methods. 25
3.2 Solution proposed for the web classification problem using the original tag dataset. ................................................................. 27
3.3 Solution proposed for the web classification problem using the text dataset. 28
3.4 Architecture for the ONN. ............................................................ 29
3.5 The experimental setting used for the project. This procedure describes the separation of the train and test set for all the possible models. ............... 32
List of Tables

3.1 Basic Tag elements with their names and function in a rendered web page. . . 24
3.2 Initial results of using the WCAFS algorithm in the text dataset. .......... 33
4.1 Overall results of the One-class models for the Login class. ............... 36
4.2 Overall results of the One-class models for the Search class. ............... 37
4.3 Overall results of the One-class models for the Form class ................. 38
4.4 Overall results of the One-class models for the Article class. ............. 38
4.5 Overall results of the One-class models with feature reduction .......... 39
4.6 The best models selected for each class without counting for the PCA method filter. ................................................................. 40
5.1 Results for applying SVM in a Multi-class problem. .................. 42
5.2 Overall results of the SVM model with PCA feature reduction. ...... 42
5.3 Results for applying Multi-class KNN. ................................. 43
5.4 Overall results of the KNN model with PCA feature reduction. .... 44
5.5 Results for applying Multi-class MNB models. ......................... 44
5.6 Results for applying Multi-class NN models. ................................ 45
5.7 Overall results of the NN model with PCA feature reduction. ...... 45
5.8 The best models selected for each Multi-class configurations. The last two rows show the results of the best One-class Article model and the baseline. . 46
6.1 Overall results of the One-class models using the text dataset. ........ 49
6.2 The best One-class models compared with the best models in the tag dataset configuration. .................................................. 50
6.3 Results of the Multi-class with the text dataset. .......................... 50
## Contents

Abstract ix
List of Figures xi
List of Tables xiii

1 Introduction 1
   1.1 Context 1
   1.2 Problem Definition 2
   1.3 Motivation 3
   1.4 Classification of four different classes 4
   1.5 Objectives 6
      1.5.1 Specific Objectives 7
   1.6 Hypothesis and Research Questions 7
   1.7 Solution Overview 8
   1.8 Main Contributions 9
   1.9 Document Structure 9

2 Background 11
   2.1 Web Mining 11
      2.1.1 Web structured Mining 12
      2.1.2 Web Content Mining 13
      2.1.3 Web Usage Mining 14
   2.2 Data Mining Techniques 14
      2.2.1 Multi-class Models 14
      2.2.2 One-class Models 16
      2.2.3 Comparison with Text Classification and Web Classification 18
      2.2.4 Principal Component Analysis 21
      2.2.5 Web Page Classification Algorithm Based on Feature Selection 21
   2.3 Related Work 22
   2.4 Summary 22

3 Solution Model 23
   3.1 Data Acquisition 23
      3.1.1 List of Tag Elements 23
      3.1.2 Choosing the Class Examples 24
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 Classification Models</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1 One-class Classifiers</td>
<td>27</td>
</tr>
<tr>
<td>3.2.2 Multi-class Classifiers</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3 Text Classifiers</td>
<td>31</td>
</tr>
<tr>
<td>3.3 Experimental Setting</td>
<td>31</td>
</tr>
<tr>
<td>3.4 Baseline Model used for Comparison</td>
<td>32</td>
</tr>
<tr>
<td>3.5 Summary</td>
<td>32</td>
</tr>
<tr>
<td>4 Experiments and Results: One-Class models</td>
<td>35</td>
</tr>
<tr>
<td>4.1 Experimental Setting</td>
<td>35</td>
</tr>
<tr>
<td>4.2 Login</td>
<td>36</td>
</tr>
<tr>
<td>4.3 Search</td>
<td>36</td>
</tr>
<tr>
<td>4.4 Form</td>
<td>37</td>
</tr>
<tr>
<td>4.5 Article</td>
<td>37</td>
</tr>
<tr>
<td>4.6 Inclusion of the Principal Component Analysis</td>
<td>38</td>
</tr>
<tr>
<td>4.7 General Discussion</td>
<td>39</td>
</tr>
<tr>
<td>4.8 Summary</td>
<td>40</td>
</tr>
<tr>
<td>5 Experiments and Results: Multi-Class Models</td>
<td>41</td>
</tr>
<tr>
<td>5.1 Experimental Setting</td>
<td>41</td>
</tr>
<tr>
<td>5.2 Multi-class Support Vector Machines</td>
<td>41</td>
</tr>
<tr>
<td>5.3 Multi-class K-Nearest Neighbor</td>
<td>43</td>
</tr>
<tr>
<td>5.4 Multi-class Multinomial Naive Bayes</td>
<td>43</td>
</tr>
<tr>
<td>5.5 Multi-class Neural Network</td>
<td>44</td>
</tr>
<tr>
<td>5.6 General Discussion</td>
<td>45</td>
</tr>
<tr>
<td>5.7 Summary</td>
<td>46</td>
</tr>
<tr>
<td>6 Experiments and Results for Document Classification</td>
<td>47</td>
</tr>
<tr>
<td>6.1 Experimental Setting</td>
<td>47</td>
</tr>
<tr>
<td>6.2 One-class</td>
<td>48</td>
</tr>
<tr>
<td>6.3 Multi-class</td>
<td>48</td>
</tr>
<tr>
<td>6.4 General Discussion</td>
<td>50</td>
</tr>
<tr>
<td>6.5 Summary</td>
<td>51</td>
</tr>
<tr>
<td>7 Conclusions</td>
<td>53</td>
</tr>
<tr>
<td>7.1 Future work</td>
<td>54</td>
</tr>
<tr>
<td>Bibliography</td>
<td>58</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Context

Testing is the process in which a company team, called testers, test the functions and navigation of one or multiple web pages. The aim of this group is to detect bugs and errors that need to be fixed by the developing team. The testing process is important in order to guarantee the quality of the web page. This process can be performed before and after its release.

Testing is a term used in the field of development and programming. In quality control of software, it means the application of tests or the revision of code for the verification of correct operation. It is proven that by implementing Testing mechanisms or tests, the quality of the applications and computer systems are improved [22]. Currently, software development companies have a great focus on the preparation and execution of tests to determine if a product meets the necessary requirements in order to be launched to the market. Testing can be applied in web as a mechanism of validation before release.

The objective is to guarantee software quality. In general, workers for this team (also known as testers) do not necessarily design nor construct the software. But, there could be cases where the same person is both the programmer and the tester [5].

An automatic process of Testing enables the testing of software without the need of human intervention. Unlike a traditional model, automation has the advantage that it can be executed with little or no human supervision. In addition to that, it provides a great advantage to companies, since it is faster than manual tests. Also, it frees the programmers or testers from tedious and repetitive tasks [6].

Currently, there are several automation tools that allow us to manipulate web pages. In general, these tools work as a programming language, in which a human has to send commands that represent the interaction of a human with a page. A popular tool for performing the task is Selenium. A disadvantage of this kind of tool is that it is necessary to know the structure of the web page. As will be discussed later, there is no established pattern that applies to all web pages. Therefore, a programmer has to understand the structure of a new page in order to develop a successful test case. Although, there are other tools and add-ons that help generate these test cases faster. But, in general, the dynamic structure represents a limitation when wanting to define a testing case in a more general way.
1.2 Problem Definition

The Web is a set of unstructured documents. Extracting the web information could be beneficial in identifying different patterns [11]. A web page can be used by users with different roles and intentions. Every user may present different behavior depending on his/her needs. The web page data can be used to identify profiles between different users. Some quick examples could be the use of user profiles to correctly classify a user and identify new relationships [15]. In this scenario, the web page needs the user interaction. However, there could be a need to study web pages without knowing the user interaction. In this case, the document itself is studied and several automatic tools can be used on order to extract information. With the exponential growth in the number of web pages and their dynamic properties, it is time consuming to extract information [11].

The following thesis works with web pages and their inner code known as HTML (Hyper Text Markup Language). This language is structured with tags that describe rendered elements. A tag normally has a specific function that will be interpreted by the web browser and the display element will be rendered when the user opens the web page. Tag elements also contain plain text which is displayed as is. A web page could be constructed using a large amount of tag elements. This combination is referred to as DOM (Document Object Model) [44]. For the scope of this thesis, the tag elements to use are the ones described by W3Schools [45]. For the experiment, the number of tags and the plain text of the web page are used as datasets for the classification problem. One final thing to notice, regarding HTML, is that the use of JavaScript creates new web elements at run time. This makes it difficult to check the final rendered tag elements using only the HTML file document without using browser [34]. Figure 1.1 shows the process a HTML document follows when being rendered by the browser. In today web pages, the content generated while rendering comes from the CSS and JavaScript documents as well. So, it is difficult to detect all the possible elements by only reading the HTML document.

![Figure 1.1: The rendering of a web page with dynamic content.](image)

While it is difficult to establish a general ontology, there is a practical way where we could solve the General Testing problem. To solve this, the web classification problem has to be tackled. The web classification problem describes the ability to discern a class of a web
1.3. MOTIVATION

page using their elements. This problem can be solved by applying classification models. However, for the purposes of the established testing scenario, it is necessary that the contents of the web page are used when doing the classification.

There are not many investigations that tackle this web classification problem. One possible solution is to use vision techniques to correctly classify a web page using their rendered page viewed as an image. There is a vision-based technique called VIPS that uses the rendered image of a web page in order to generate a tree like document that can be used to find patterns [7]. While this is a possible and viable way to do classification, it establishes the rules to determine the blocks and several extensions have been made in order to tackle the dynamic web pages [2]. Currently, there is not a repository containing images of several web pages with their labeled class. The process of doing this repository may consume a large number of resources and time. Additionally, it is believed that through the code itself the same results may be obtained with less computational power.

For the web classification problem, a dataset with the name of WebKB can be used in order to train the models. However, these web pages structure do not reflect the current HTML and JavaScript; for instance, in Figures 1.2 and 1.3, we can see the difference of the same web page from the WebKB database and its current implementation. The dataset can be obtained from http://www.webkb.org. While it may be a good starting point for specific cases. The main objective of the project is to be able to solve the web classification problem with real world examples.

1.3 Motivation

An automation tool that can be used to solve this problem is Selenium. This framework gives the ability to execute multiple tests through multiple machines [4]. However, to create those functions it is necessary to design and implement them in code. The engineers, responsible of designing the code, have to be trained and gain the knowledge to understand the company processes to correctly develop the scripts. Hoffman mentions that the cost of testing include the training of personnel [16]. With this in mind, the ability to reduce the test cases using an automated process may reduce the costs of testing, in general.

A general view of the entire solution can be looked in Figure 1.4. Once a model correctly identifies a web page, it can select the corresponding script in order to find the components that will help to create an automated testing. For example, in the case of Login, the individual elements will be the username and password since these elements will contain the input of the user. These elements are within, in most cases, a great amount of tag elements. A possible solution could be to use the DOM as a tree and apply search algorithms in order to find those elements.

Lastly, the script selected is executed. With the previously found elements the automation can know execute the process as if it were a real person. With this, simple test cases could be performed by a computer without human intervention. A further project could include managing automation errors and improve the speed of the execution. While this may be a long approach the main focus of the thesis is to solve the classification problem and leave the rest of the automation for a future work.
1.4 Classification of four different classes

The solution tries to correctly identify which script is necessary for a specific web page. This initial problem can be viewed as a classification problem. The general concept is: if a web page belongs to a specific class then specific scripts can be executed in it. The definition of the classes is something that vary between company processes. For this project, four classes were defined in order to represent different web functions. Another consideration was if the number of examples were found by each class. That is, if a class did not have enough examples, then it was not considered. The Login, Search, Form and Article were the easiest examples that could be defined with updated web pages. The recollection process was of significance in the definition of the classes. A more streamlined corporate process may add more specific classes according to their business needs.

The Login class describes a page which its main focus is for the client to login to a platform or service. The most common structure is to have a text box for a username and then
1.4. CLASSIFICATION OF FOUR DIFFERENT CLASSES

Figure 1.3: Current Cornell web page.

another for a password. Not all login pages have this structure, but the main idea is to input at least two texts. When talking about testing, this example could be used to verify that the authentication process is working correctly.

The Search class describes a web page that is used by users to discover extra content from the current web page or from other sources. The most common cases of search pages are pages like Google or Bing. Other examples are institutional web pages that contain a great amount of bibliographical content and the search page is used to filter the results. Normally, search classes have an emphasis on one text box that will receive the keywords or input text that describe the content that the user is looking for. Another variation is the advanced search page where more text boxes are used to limit the results. Other important elements are the list indexes, which are normally at the bottom of the page and help when the results are too much to render in one page. If the user clicks on another index then the corresponding results are shown. In testing scenarios, one process may include the verification of these elements and the validity of the results that are showed to the user.

The Form class is a web page that obtains information from the user in order to be saved at a database. The form class is typically structured with several input elements that generally are text boxes. Some other elements include: buttons, check-boxes, radio-boxes, texts-area, etc. The most common examples are registration forms or “Contact Us” web-pages. One testing case is to verify that the form elements correctly record the users input.

The Article class is a type of web page which is different than the previous classes. The objective of this web page is to present information to the user. The most common examples are News web pages or Blog articles. The most common element found in this type of page is the paragraph elements. Another common elements are the images and hyperlinks to another article. In this case, the scenario is more of an automation scenario. A company could be interested in obtaining information from several web pages and process that information. To do this, the automation is performing web scrapping from the articles. Web scrapping is a technique where the specific information of a web page is obtained and formatted into a text document.
CHAPTER 1. INTRODUCTION

Figure 1.4: General diagram showing the three modules of the automation solution.

The definition of these classes could change depending on the goal of a company. But, these classes were defined for the project in order to experiment with the web classification problem. In a company there could be \( n \) number of different classes. The focus of this thesis is to solve the web classification problem using different known techniques.

1.5 Objectives

With the previous definition of web classification the main objective of the thesis can be established as:

To construct a classification model, with current web pages, that can predict among four different classes (Login, Search, Form and Article) using web components like tag and text elements as inputs. It is expected an accuracy of over 80% through the use of classification models like Support Vector Machines (SVM), K-Nearest Neighbor (KNN) and their One class counterparts.
1.6 Hypothesis and Research Questions

It is believed that the Data Mining (DM) techniques could improve the web automation process. Mostly because they could give the specific context of a HTML web page. In several DM techniques, instances can be declared with their given features and class label. Following these principles four web classes are defined as: Login, Search, Form, and Article classes. This in turn may lead to a web classification problem where the objective is to determine if a new instance belongs to a specific class.

With this in mind, the hypothesis is defined as: The use of DM classification methods will correctly solve the web classification problem with updated web pages (for the classes Login, Search, Form, and Article) using merely the HTML tag elements.

For this project, the One-Class, Multi-Class and Document Classification models are used to perform the classification of the web pages. This comparative may determine the best models to use in the testing scenario. The Login, Search, Form, and Article classes were chosen as it was the most common and clearly separable testing cases. However, the main idea behind the project is the ability to construct a new class and having the model determine its features given the tag elements. The results show a possible way to use these methods to determine whether one instance belongs to a specific web class.

It is believed that, depending on the classes, the number of tag elements will present a visible pattern. For example, the Login class will have present the tag element password more so than the other classes. Another example may be the class Form containing more elements of the form tag.

The following questions are expected to be answered:

1.5.1 Specific Objectives

Some particular objectives from this project are:

1. To construct a dataset containing the inner information of 2000 web pages. The inner information will consist of the number of tag elements and the web page text. The 2000 web pages will be divided into four classes each containing 500 examples.

2. To test several classification methods to obtain the best possible combination that gets the more accurate results. The main techniques to test are the One class models. More specifically the SVM, KNN, and Multinomial Naive Bayes (MNB). Additionally, Multi class counterparts will be tested and the techniques of Neural Networks (NN).

3. Filter specific tag features to obtain the most relevant elements that represent a web page. The main methods of filtering will be the K-Best features, Las Vegas Filter (LVF) and Principal Component Analysis (PCA).

4. To verify if the web page classification problem can be viewed as a simple document classification problem. To do this the dataset will be divided into two: one containing the tag elements and other containing the text information. The project will check if there is a difference between using the text or tag datasets for solving this problem. The text dataset represents a document classification problem.
1. Can the tag elements serve as a replacement dataset for the traditional text dataset?

2. What is the performance of a classification model using limited data and compared against a baseline model?

3. Is it possible to solve the web classification problem using updated web pages?

4. What is the best classification method for this particular problem and considering different scenarios?

5. Can the web classification problem be tackled using a variety of classification models?

1.7 Solution Overview

The developing team frequently uses the HTML markup to develop the web pages. However, the HTML, being a markup language, does not provide a specific structure that all web pages must follow. The markup language has more in common with natural languages, such as English or Spanish, than a programming language. In a programming language, like C or JAVA, there is an arithmetical and a logical structure that every program has to obey. HTML is more like a canvas were the programmer, or designer, has the ability to define the structure according to their own needs. This is a good thing since it makes the web pages dynamic and more creative. But, trying to find a pattern within the code of multiple web pages may be difficult. Nevertheless, there are multiple tags that are used for specific functions and it is believed that through this information it is possible to find multiple patterns in the web pages.

It is believed that web pages that have similar functions may have the same tag elements. To verify this, a dataset must be created containing the total of tag elements of multiple web pages. As said before, four categories were created: Login, Search, Form, and Article. These four categories will represent four different test cases and the problem could be viewed as a simple classification problem. To correctly classify, there will be used several classification methods that are found in the current literature.

Additionally, the web page code can be viewed as a plain text document. In this case, the web page can be processed using text classification methods. There could be specific keyword that correctly represent a single classification. To this end, some pre-processing is necessary because the HTML markup contains specific elements that represent functionality in the web browser.

Three different types of models were constructed: One Class-models, Multi-Class models and Document Classification models. The data of 2,000 different real world web pages were obtained in order to train and test the models. For the One class section, four models were trained, one for each class, and the model was tested using a portion of the dataset. The process was repeated for the Multi-Class models, while the Document classification models were trained using the plain text of all the web pages. At the end, the majority of the models presented an accuracy of over 70%. The best performing model was the One-Class model for the Article class. However, the best model that predicted the four types of class was the Document Classification model using the plain text of a web page.
1.8 Main Contributions

The main contributions of this thesis are listed below:

1. Construction of a dataset containing the information of 2,000 current web pages. This dataset includes the number of tag elements and their text information.

2. Implementation of a process to scrap the data of 2,000 web pages. This process can be extended to obtain information of additional web pages.

3. Experimentation and comparison between One class models with Multi class models. The results showing the best possible application when dealing with this problem.

4. Experimentation and comparison between using text data, like a document classification problem, and using the tag elements as a way to classify web pages. Text data is known to give good results.

5. Inclusion of a feature reduction technique to verify if the results of a model are improved.

6. Experimentation using the tag dataset. This particular dataset can be used regardless of the language of the document.

1.9 Document Structure

The following sections are divided as follows: Chapter 2 presents the background of the study, it focuses on establishing the concepts of Web DM, explaining the current models that are going to be used and the previous work done that tackles the web classification problem. Chapter 3 describes the solution models, how the data was obtained and the experimental settings used. Chapter 4, 5 and 6 show the results of the three types of models; and finally, Chapter 7 discusses the conclusions of the thesis and establishes the future work.
Chapter 2

Background

This chapter explains in more detail the technical parts of the thesis. The entirety of the process tries to solve the web classification problem, described in the previous chapter. The project does not create or improve previously known models. Instead, it is more of a comparative study to determine which model or combination of models could be used to better solve the web classification problem.

2.1 Web Mining

Web is the highest source of data in the world. It is also one of the most accessible data sources. However, it is difficult and time consuming to extract the information from the Web due to special characteristics:

1. The amount of information is extremely large. It is everyday growing and it could cover a diverse number of topics.

2. There are multiple types of data in the Web. For example: videos, images, tables, among others.

3. The data from the web is heterogeneous. This means that one simple topic could be handled by multiple authors with different terminology or keywords.

4. There is a link structure on the web. Multiple web pages are linked together and are referenced within their web pages through links.

5. The data on the web page is noisy. Because, some web pages have adds, billboards, tabs or columns from other content that compete with the main content.

6. Web is also about interaction with the user. Some information could be the button clicks or the navigation logs.

7. The content of a web page is constantly changing and evolving.
Due to these reasons, it is difficult to simply extract information from the Web. DM is used to transform the noisy information into a condensed dataset (typically in a table form). Thus, Web DM (WebDM) aims to discover useful information from web documents [29].

Web data can be classified into three different areas [40]:

1. **Content**: It refers to the real data that is inside of the web page. May include images, audio files, tables, and lists.

2. **Structure**: It is the organization of the content using the HTML and XML tags.

3. **Usage**: It describes the pattern of the users behavior while in the web.

WebDM can be categorized into these three same applications: Web Structured mining, Web usage mining, and Web content mining. WebDM problems can be tackled using DM techniques and procedures. Figure 2.1 shows how these three categories are separated within the web mining context. In this project, several techniques will be explored. The important distinction is the fact that the data used reflects real Web scenarios.

![Figure 2.1: The three different categories of web mining.](image)

### 2.1.1 Web structured Mining

This area of Web Structured Mining focuses on the relations between web pages. Normally, this is thanks to the hyperlinks that are embedded in a single web page. Web structured mining tries to find a powerful relation between a web page and their linked pages to identify similarities. The algorithms used in this process require that the link elements of a web page are extracted and the data obtained is compared with other web pages and their relationships.

One way to look at Web structured mining is to compare it to graph theory. In this comparison, the nodes of a graph are web pages, and their connections are the hyperlinks that reference each other. Some of the current applications of web structure mining use graph theory algorithms. Web structured mining can be divided into two [31]:
1. **Hyperlink analysis**: The identification of patterns between the connections made of web pages through hyperlinks.

2. **Document structure**: The analysis of the tree document structure form the web page and their relationship with the tag elements.

Some common applications are the search engines that use this hyperlink structure to obtain similarities among web pages. The PageRank algorithm is one of the main focus when trying to improve the web page similarity through links. Other combinations have been made to improve the relevancy of the search results [37].

Recommendation systems for social network analysis are also a common use of Web structure mining. This system is used for identifying groups between people, companies, computers, and other connection entities [19].

Web structure mining follows four different steps [40]:

1. **Data collection**: This means the collection of hyperlinks and web documents.

2. **Pre-processing**: It validates the links and transforms the data.

3. **Knowledge discovery**: It applies DM techniques to the processed data.

4. **Knowledge analysis**: Identifies the patterns and displays the information to the final users.

### 2.1.2 Web Content Mining

Web Content Mining is the area that uses the elements inside of the web page. For example, the topics talked on a web page, the text used or the comments a user makes while visiting a web page [29]. In the majority of the cases, this content is unstructured and the need to extract this information is useful in order to obtain the context of the web page. The use of these techniques has been applied to several pattern discovery problems like e-commerce [43]. Web content was also used to analyze the activities of research and development departments from different companies [13].

Web content mining deals primarily with Information Retrieval (IR) from the web pages. As stipulated above, the web pages have a semi-structured form. This makes it difficult to apply traditional DM techniques since the majority deal with relational datasets. The area of web content includes the integration and extraction of relevant information and its processing to formats that are understood to DM techniques. For this reason, some applications of web content mining include: categorization, clustering, user modeling, and finding patterns [26].

Like Web structured mining, web content mining can be divided into two types [28]:

1. **Text mining**: The use of text model techniques in order to find patterns, extract information, and in general obtain knowledge from unstructured or semi-structured web pages.

2. **Multi-media mining**: This area focuses on the extraction of media content within the web page in order to process it and analyze it.
2.1.3 Web Usage Mining

This area of Web usage mining tries to find patterns in the end user while it navigates the web. For example, the recordings of the clicks made by the user are considered to fall in web usage mining [29]. The main objective is to represent, in a readable way, the behavioral patterns that the user has while navigating or using the web page. The way to obtain this information is through three sources [18]:

1. **Server logs**: This refers to the collection of data that is stored in the servers. This data may include IP addresses, time logs, web pages visited, etc.

2. **Client data**: It is collected from the host that access the websites. The way to obtain the information is through JavaScript implementations. They generally describe the users navigational history.

3. **Intermediate data**: When the user request a web page through an intermediate devices such like proxy servers. The information gain could vary depending on the request of the user.

One application of web usage mining is to provide better content depending on the user behavior. In a study, raw log files are used to provide the user with better products [30].

This thesis can be viewed as a combination of Web structure and Web content mining. Since the inner code is used, one could identify it as a Web structure. However, the application at run time and the use of the plain text of web pages can classify this problem as Web content. In reality, one problem can belong to different classification problems. This problem has the distinction that uses inner HTML code in order to classify contextual information of the web page.

2.2 Data Mining Techniques

The idea of this project is to incorporate several DM models to solve the web classification problem. The models can be divided into two sections: Multi-class models and One-class models. Both types try to solve the problem of identifying the class of a new instance. For both types there are considered the following methods: SVM, KNN, and MNB.

2.2.1 Multi-class Models

Multi-class models have the distinction of having more than two target classes. If a model only has two categories, for example, positive and negative; this problem can be viewed as a binary classification problem. As said before, Multi-class have $N$ number of classes, and the objective is to find the set of rules that correctly classifies a new instance.

**Support Vector Machines**

Support Vector Machines (SVM) are useful when trying to find non-linear classification problems. SVM transforms a dataset into a higher dimension (which does not have a linear representation). The main objective is to be able to define the optimal separator in the hyperplane.
that could help with the separation of the classes in a classification problem [27]. A normal linear function, for a set of $x$, may be classified as follows:

$$F(x) = w^T x - b$$

(2.1)

where $w$ is a weight vector and $b$ is a bias calculated during training. A margin is presented with $\frac{1}{|w|}$ which represents the margin between the closest data point of each class. The objective is to be able to reduce this margin while maintaining the classification [47]. Figure 2.2 shows a representation of how the margin describes the line that correctly classifies the data points.

![Figure 2.2: The SVM function that separates the data points using a separator with a margin.](image)

For non-linear problems, SVM transform the data points into a higher dimensional vectors. While these vectors are in a high dimension, the linear function is discovered using this high dimension plane. However, the original input space appears to be a non-linear function [47]. The decision boundary is represented as:

$$F(x) = w^* \theta(x) - b$$

(2.2)

Where $\theta(x)$ represents the transformation of $x$ into a higher dimension plane. Generally, SVM are most commonly used with binary problems. However, one could interpret a multi-class problem as a combination of multiple binary-class problems. SVM do this by creating multiple models that correctly tries to classify two classes of the original classes. At the end, a voting mechanism is applied to identify the correct class of the instance. This method can be known as one-vs-one (OVO).

**K-nearest Neighbors**

KNN works by using a set of instances that can be represented in a $m$-dimensional space. Thus, forming a pattern that can be seen from this $m$-dimensional space. When a new instance is presented, it establishes its class by looking at the $k$ neighbors that are near its data point in the $m$-dimensional space. An Euclidean distance is used to determine that a point is close or near another [35]. The Euclidean distance is measured using the following formula:
CHAPTER 2. BACKGROUND

\[ d(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \]  

(2.3)

The new instance is assigned the most prominent class of its k neighbors. KNN is one of the simplest methods to implement since the class is assigned as a voting mechanism between the k neighbors. Generally, k has a low value and an odd number is assigned in order to prevent ties [35].

**Multinomial Naive Bayes**

MNB assigns the class of an instance based on its likelihood. It uses the Bayesian theorem that describes the probability of an event based on prior knowledge of that event [36]. The Bayes’ rule specifies that given an event E and a class c. The probability of E being class c is:

\[ p(c|E) = \frac{p(E|c)p(c)}{p(E)} \]  

(2.4)

The most probable class is the one assigned to the new instance. The simple form of MNB dictates that all the attributes from the instance are independent from each other. Although, such independence is very hard to accomplish. MNB has had a popularity with text classification problems [48]. In this problem, event E is considered to be a document D with a vocabulary of words W. The recollection of all different words W from all documents D are generally called the dictionary. For this problem the NB is considered as a MNB. In MNB, the word count for each document is treated separately in the calculations of \( p(d|C) \). That is, the prior knowledge of \( P(d|C) \) is calculated using the probability of the words in the document in class C \( p(w|C) \). This technique is being mainly used for text classification problem where there is a great amount of documents used for training [33].

**Neural Networks**

NN consist of the connection between multiple layers called neurons. Each neuron produces a sequence of real value activation. Generally, neuron receive their inputs from their environment or from other neurons after a weighted calculation is applied. Learning comes by altering these weighted values in order to find the desiring behavior from the NN [39].

NN are used in multiple problems, one of them is the classification of text documents. NN are beneficial in this kind of problems due to their ability to process multiple features. Although, this may come to a high computational price [23].

For the text classification problem there has been a raising popularity with Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN). Both have reached solutions that reach over 90% accuracy over text classification problems that include a great amount of corpus [9].

**2.2.2 One-class Models**

The previous multi-class models try to classify one instance into a predefined list of categories. However, there is a big problem when there is not enough data that can be used to train the
2.2. DATA MINING TECHNIQUES

model. In the real world, there is a lot of information about one class (target class), but there is not enough information, or any, about an unknown or outlier class. To solve this problem one-class models are used [24].

One-class Classification Models (OCC) are classification models that are trained using data for the target class and none, or few, data from the unknown class. Generally, the target class is labeled as a positive and the outlier is labeled as a negative. The main objective is to create a boundary, using the positive examples, in order to accept future positive values while reducing the number of negative examples accepted [24].

OCC can be tackled using several algorithms and methodologies. Some multi-class algorithms have been translated to solve the OCC problem. Three models are studied in this project: One-Class Support Vector Machines (OSVM), One-Class K-Nearest Neighbors (OKNN), and One-Class multinomial Naive Bayes (OMNB).

**One-Class Support Vector Machines**

Instead of using a hyper-plane to distinguish between two classes, a hyper-sphere is created using the positive examples. These positive examples represent the positive data points in the plane and a minimum radius is used around them to generate the final boundary. This method is also known as the Support Vector Data Description (SVDD) [24]. Kernel functions can be used in order to improve the boundary and reduce the number of positive examples labeled as negative. Generally, the Polynomial or Gaussian kernel are good fits for the OCC problem. However, due to the absence of negative training data, OCC problems with SVM generally require a high amount of positive examples [24].

**One-Class K-Nearest Neighbors**

Like regular KNN, this method uses the nearest neighbors in order to identify if the instance belongs to the positive class. The method is generally used with $k = 1$. For a new instance $z$, it is calculated the distance to the first nearest neighbor. The neighbors are the training set that contains the positive values. Then, this nearest neighbor is used to calculate its distance to its nearest neighbor in the training set. The division between the two distances is used to determine the class [10].

$$NN(z) = \frac{||z - NN^{\text{tr}}(z)||}{||NN^{\text{tr}}(z) - NN^{\text{tr}}(NN^{\text{tr}}(z))||}$$ (2.5)

Where $NN^{\text{tr}}(z)$ is the nearest neighbor of $z$ from the training dataset. A threshold $th$ can be used to determine the final calculated distance and evaluate if $z$ belongs to the positive or negative class.

For applying this method with $k > 1$. The following steps are proposed:

1. Having the number of neighbors $k$, a threshold $th$, and a training dataset $D_{tr}$ with size $n$. Create a matrix of $n \times 1$.

2. For every $i \in D_{tr}$, calculate the Euclidean distance of its $k$ nearest neighbors. Using the mean of these distances populate the $n \times 1$ matrix.
3. Obtain the nearest neighbor of \( z \) and obtain its euclidean distance \( D_1 \).

4. Using nearest neighbor. Obtain the \( k \) nearest neighbors distances from the matrix \( nx1 \) and obtain the mean \( D_2 \).

5. Calculate the distance \( D(z) \) as the division of \( D_1 \) from \( D_2 \).

6. If \( D(z) < th \). The class of \( z \) is positive. Otherwise the class is negative.

The value of the threshold \( th \) can be calculated using a harmonic mean of the euclidean distances from the \( nx1 \) matrix [1].

**One-Class Multinomial Naive Bayes**

Since the dataset used to train the model is going to be positive values. The process of OMNB is a little different than its regular multi-class form. In this scenario a threshold \( th \) is used to determine the class of the new instance. If \( P(z|C) \) is smaller than \( th \) then we can label that instance as a negative class [42].

The following procedure goes as follows:

1. For every \( d \) in \( D_{tr} \) the probability \( P(d|C) \) is calculated.

2. The probability of \( P(d|C) \) is represented as the product of the probabilities \( P(w|C) \) for all the words that appear in document \( d \).

3. Given a new instance \( z \) the probability \( P(z|C) \) is calculated. If the value is less than \( \delta th \) we label the instance as negative.

The value of \( \delta \) can be established as the minimum distance, of \( P(d|C) \), from of all examples in class \( C \) [32].

**2.2.3 Comparison with Text Classification and Web Classification**

As said in the previous chapter, the web classification problem can be viewed as a way to correctly classify a web page given their contents. Several models and techniques are used to solve this particular problem. However, another problem can present similar characteristics. The text classification problem or document classification problem can possibly share elements of the web classification problem. This also implies that algorithms applied to the text classification problem can be used to solve the web classification problem. In the following sections, it will be described how these patterns can be used.

**Text Classification**

In its simple form, the text classification can be viewed as the ability to assign predefined classes to text documents. It can be defined as a set of documents \( D = d_1, d_2, ..., d_n \) where each document \( d_i \) is assigned a class \( l_j \) from a set \( L = l_1, l_2, ..., l_m \). The task is thus to find \( f \) such that [3]:
2.2. DATA MINING TECHNIQUES

\[ f : D \rightarrow L \quad f(d) \rightarrow l \] (2.6)

For the majority of the text classification cases, the measurements used are precision, recall, and F-measure (or F-score). These measures determine the effectiveness of the classification model. The precision and recall can be defined as [14]:

1. **Precision**: It is the probability that an instance is relevant given the total of predictions made by the model. The following equation can be used:

   \[ \text{precision} = \frac{TP}{TP + FP} \]

   Where \( TP \) is the total of true positives and \( FP \) is the total of false positives.

2. **Recall**: It is the probability that an instance is relevant given the total of real instances in the test. The following equation can be used:

   \[ \text{precision} = \frac{TP}{TP + FN} \]

   Where \( TP \) is the total of true positives and \( FN \) is the total of false negatives.

3. **F-score**: It can be calculated as the harmonic mean of the precision and recall. The following equation can be used:

   \[ F\text{score} = 2 \cdot \frac{PR}{P + R} \]

   Normally the F-score is a better measurement to calculate the accuracy of a model. Most of the previous work, described in a future section, uses the F-score.

Another term used in text classification is the words frequency and vocabulary. Given the set of documents \( D \) let \( V \) be a set of collections of words \( V = w_1, w_2, ..., w_n \) each word \( w \) establishes a different term in the collection. The frequency \( f_d(w) \) describes the number of repetitions a word \( w_i \) appears on a document \( d_j \) [3]. A vector can be defined as:

\[ \vec{t}_d = f_d(w_1), f_d(w_2), ..., f_d(w_n) \]

(2.7)

This vector is commonly used to describe the features required by the classification model. A short example can be viewed in Figure 2.3. While this is a valid way to create numerical instances of documents. There is a technique called Term Frequency–Inverse Document Frequency (TF-IDF) that helps with the optimization of the classifier.

**Term Frequency - Inverse Document Frequency**

TF-IDF is a numerical statistic that is meant to show how important a word in a document given a collection of documents. The goal of the technique is to augment the value of rare term given a number of similar documents [38]. The result is that those words that do not repeat frequently enough in all the documents are given an elite status which helps to discriminates
Web page classification is similar to text classification. But, the web document may be different than a text document.

The method is divided into two techniques: TF and IDF. The TF is the number of repetitions a word appear in a document. IDF determines whether a word is common or rare between a collection of documents. The value can be obtained by the logarithmic of the total number of documents divided by the total number of documents containing the word $w$ [38]. The following equations can be used:

$$tf(w, d) = frequency(w, d)$$  \hspace{1cm} (2.8)

$$idf(w, D) = log\frac{N}{|\{d \in D : w \in d\}|}$$  \hspace{1cm} (2.9)

With these definitions the TF-IDF can be calculated as:

$$tfidf(w, d, D) = tf(w, d)idf(w, D)$$  \hspace{1cm} (2.10)

The TF-IDF is used when a vector of frequencies are used. In the web classification problem, the frequencies of the tag elements can be used instead of a word frequency. With the goal of obtaining a better context, the TF-IDF is used in the experiment in order to see if the frequency of tag elements can be improved using this method. The normal TF-IDF method is also used with text documents of the web pages.

<table>
<thead>
<tr>
<th>Terms (Frequency)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web (1)</td>
<td></td>
</tr>
<tr>
<td>page (1)</td>
<td></td>
</tr>
<tr>
<td>classification (2)</td>
<td></td>
</tr>
<tr>
<td>text (2)</td>
<td></td>
</tr>
<tr>
<td>web (1)</td>
<td></td>
</tr>
<tr>
<td>document (2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$d = \{1, 1, 2, 2, 1, 2, ...\}$

Figure 2.3: Basic example of transforming a text document into a word vector.
2.2. Principal Component Analysis

Principal Component Analysis (PCA) is a linear unsupervised classification technology. It has the ability to create interrelated variables uncorrelated through singular value decomposition (SVD) or eigenvalue decomposition. PCA is used when trying to reduce the dimensions of a dataset and it has proven to be successful in multi-class classification [20].

The process of PCA is generally described as follows [21]:

1. Given a set $X = x_1, x_2...x_n$ calculate the mean $m$.
2. Re-scale every $xinX$ to have unit variance. This to ensure that all the attributes are treated on the same scale.
3. The eigenvalue is calculated for every position. The highest $k$ values are chosen.

2.2.5 Web Page Classification Algorithm Based on Feature Selection

The Web Page Classification Algorithm Based on Feature Selection (WCAFS) was proposed by Zhou, et al [50]. The algorithm is based on the TF-IDF transformation. The difference, is that the WCAFS algorithms calculates the weight of the word based on three aspects: category, Inter-class Deviation (ID) and Intra-class Distribution Ratio (IDR).

The category is determined depending on the location of the term. If a term is found in the title of the document, then the category is set as 4. If the term is found in the description of the web page, then the category is set as 2. The category is set as 1 if the term is not found in any of those locations.

Inter-class Deviation (ID) means that a term may appear in certain classes and not in others. The calculation of ID is as follows:

$$ID(w, C_j) = \frac{N(w, C_j)}{\sum_{x=1}^{m} N(w, C_x)}$$  \hspace{1cm} (2.11)

Where $N(w, C_j)$ is the number of documents that contain the term $w$ in class $C_j$, $\sum_{x=1}^{m} N(w, C_x)$ is the sum of all the documents that contain the term $w$ in all the classes, and $m$ is the number of classes.

Intra-class Distribution Ratio (IDR) is the probability that the term appears in all the documents in a class. The calculation of IDR is as follows:

$$IDR(w, C_j) = \frac{M(w, C_j)}{M(C_j)}$$  \hspace{1cm} (2.12)

Where $M(w, C_j)$ is the frequency that the term $w$ appears in class $C_j$, and $M(w, C)$ is the total number that all the terms appear in class $C_j$.

The final WCAFS transformation uses the following formula:

$$W(w, D_i) = \frac{f(w, D_i) xlog(\frac{N(D)}{N(w, D)} + 0.01)}{\sqrt{\sum_{k=1}^{n} (f(w_k, D_i)^2 x (log(\frac{N(D)}{N(w, D)} + 0.01))^2}}} x ID(w, C_j) x IDR(w, C_j)$$  \hspace{1cm} (2.13)
Where \( f(w, D_i) \) represents the category of term \( w \) in document \( D_i \). \( N(D) \) is the number of documents in the collection \( D \), \( N(w, D) \) is the document frequency of term \( w \) in the collection \( D \). The variable \( n \) is the number of terms in document \( D_i \). \( ID(w, C_j) \) is the Inter-class Deviation of term \( w \) and \( IDR(w, C_j) \) is the Intra-class Distribution Ratio of the term \( w \). The class \( C_j \) is the class that the term \( w \) belongs in the document \( D_i \) [50].

**2.3 Related Work**

Several applications are being made to solve the classification of web elements. One of the most prominent experiments is the work done by Gkantouna and Tzimas [12]. They established that the hypertext level of a web page can be used to determine the specific patterns of their elements. In their experiment they use the Joomla framework to determine the different elements or attributes that one instance has.

Another experiment, performed by Yeongsu and Seungwoo, uses specific tag features such as title, date and paragraph to perform an SVM model in order to do the classification of web content [25]. Their work relates to a specific area of DM called WebDM. They concur in the use of web tags to determine the category of a web page. Although, unlike Gkantouna, they do not use the Joomla framework to establish their classes.

Experiments made by Sun et al. [41] have gotten a SVM classification model using the WebKB dataset. Their problem is to classify between different types of university web pages. Applying different elements to the SVM they got an F score of 0.575. They use different types of context elements such as: text, title and anchor words. This article does show that an SVM model could be used to classify web page documents.

Chen and Hsieh [8] have created another SVM model which consists of two SVM models: one using latent semantic analysis and other uses text features. The final model is a voting system that consists of determining whether one instance belongs to a class or not. The results show that the voting schema is better than previously know methods with a precision of over 90.

Another experiment using text documents was made by Zhou et al [50]. This experiment proposes a new algorithm that outperforms the TF-IDF transformation. With this experiment a precision of over 0.9 is found. The TF-IDF transformation has a precision of 0.8.

**2.4 Summary**

In this chapter, the WebDM was defined as three possibles categories: Web structure mining, Web content mining, and Web usage mining. For the purposes of the project, the web classification problem will fall into a combination of Web content mining and Web structure mining. This is because the tag dataset is going to be used to correctly classify web pages without the need of user information. Different DM techniques were described in this chapter. In order to solve the web classification problem, the models chosen were the SVM, KNN, MNB, and NN. Additionally, the problem can be viewed as a text classification problem and the TF-IDF transformation can be used to improve the overall results of the models. Finally, the One-class models are described since it presents an opportunity to be able to train the models with limited data.
Chapter 3

Solution Model

The following chapter explains the different models that were used in the final sets of experiments. The aim of the project is to compare different configurations of models in order to determine the best solution for the web classification problem. This chapter will describe the procedure followed when recollecting the data and the construction of the final models.

3.1 Data Acquisition

Gathering the data is difficult in this project. There is not a real dataset that contains the information that it is needed. For solving this problem, the information of real web pages have to be extracted. To help with the creation of the dataset, a web crawler was made using Selenium. The program was made using C# and the full code can be obtained in https://github.com/samuelistico/web-examples-repository/. The general extraction problem follows these steps:

1. A list of 2,000 web pages is made containing the URL of all the different web pages.

2. For every URL the crawler opens a Google Chrome browser and navigates to the following page.

3. Once the page is fully loaded, a JavaScript command is executed in order to obtain the count of all the elements specified below. The count is zero if a web page does not contain the specific element.

4. The plain text of the web page is then extracted using a library in C# (called HtmlAgility).

5. The data is saved as a json file and then it is transformed into a csv file for each class. The final result is four files corresponding to the different classes.

3.1.1 List of Tag Elements

The W3Schools specifies a list of 119 different tag elements [45]. All of those elements will be used as features for the dataset. However, there are tags that are going to be removed, transformed or added.
The elements that are going to be removed are those tag elements that do not provide additional information for the problem or that are common among all web pages. For example, the tags: BODY, HTML, DOCTYPE and BODY will be removed since the number of frequency in all web pages is 1.

The `a` tag elements, which represents a hyperlink, are going to be divided into two different types: `a` and `a.href` elements. The `a.href` element is an `a` element that contains the attribute `href`. The rest of the elements will be considered simple `a` elements. This distinction is made to identify `a` elements that function as links or action objects. Links generally send the user to another web page while action objects may reflect an interaction with the user.

The INPUT tag elements are going to be transformed into 22 different elements. These elements are the ones found in the W3Schools web page [46]. The reason to do this is because every INPUT element has a distinct function regarding their type. The most common example is text and password. Both of these elements are INPUT elements but with different functions. For the purposes of the web classification, these elements are going to be added in the final dataset.

With these changes a total of 141 features are obtained. However, other elements are going to be removed from the dataset. There are 21 elements that have a value of 0 in every instance. Since they do not present relevant information, these features will be removed. Features with a frequency less than 50 will be also removed. The information gained using these features is not much. The total of features with less than 50 repetitions is 30. One last set of tag elements are removed, these elements were chosen due to being elements that contribute little with the final rendered objects of the DOM. The number of these elements are 7 and were chosen manually.

The final dataset contains a total of 83 different features. That is, of the previously 119 elements, 36 elements were removed in order to obtain the real tags that are believed to be helpful in the web classification problem. Table 3.1 presents a short list of the most representative elements and their function. Figure 3.1 shows a representation of the final dataset.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.HREF</td>
<td>Represents a hyperlink in a web page.</td>
</tr>
<tr>
<td>A</td>
<td>Represents a A element that does not necessary functions as a hyperlink.</td>
</tr>
<tr>
<td>INPUT.TEXT</td>
<td>Represents a text box where a user can introduce text values.</td>
</tr>
<tr>
<td>INPUT.PASSWORD</td>
<td>Similar to the previous element. Except the value typed it is hidden.</td>
</tr>
<tr>
<td>BUTTON</td>
<td>Represents an element that can be clicked by the user triggering an action.</td>
</tr>
</tbody>
</table>

Table 3.1: Basic Tag elements with their names and function in a rendered web page.

### 3.1.2 Choosing the Class Examples

The 2,000 examples are divided into four classes with 500 examples each. Each instance had to be manually chosen from the Internet. Generally, the examples were found using special keywords on popular search engines like Google or Bing. Certain rules had to be met in order to be considered as an instance of a class. These rules are defined for each class.
3.1. DATA ACQUISITION

Figure 3.1: Final dataset used in the experiments. The content column represents the text dataset that is going to be transformed using the text transformation methods.

**Rules for Login**

For the Login class, an authentication form must be heavily present in the web page. An authentication form typically consist of three elements: the username text box, password, and an action button. The majority of the examples had the login form in the center of the web page. Other examples had the form with extra content. Some special cases had only the username text box and, after filling the information, the password text box would appear later. These cases were also considered as instances of the Login class.

**Rules for Search**

For the case of the Search class, the search text box was the element that had to appear and be very prominent in the web page. Normally, like Google, the text box is large and is either in the middle or in the top of the web page. Other variations could include having the text box on the top right corner and having the rest of the web page be filled with static results. Another variation is the advanced search web page, this page contains multiple text boxes that represent a more specific search. The majority of the web pages used were universities databases.

**Rules for Form**

The Form login was obtained mostly by fill up forms or "Contact Us" web pages. The key factor was the number of input elements. Since a login could be considered a form; the web pages were picked if they had at least 3 input elements that were different from those of the Login class. For example, a form with three text boxes could be considered an instance of the Form class. However, there could be a possible overlap with the advanced search case from the Search class. For this reason, other web pages were obtained were the elements were different than text boxes.


Rules for Article

The article class was the easiest of all the classes. The examples were articles that contain a big amount of paragraph elements. For this reason, popular news and entertainment web pages were used to obtain these examples. Other elements that could appear in the article were images and headers.

Construction of the Dataset

Once the list of the web pages was created, the C# program was executed to obtain the information of all 83 tags and the plain text of each web page. This dataset was then divided into two: the tag dataset and the text dataset. In the end, both datasets represent a term frequency vector of a single web page.

3.2 Classification Models

As said before, the models can be divided into two categories depending on the dataset used for the problem. These categories are: tag models and text models. In both cases the SVM, KNN, MNB, and NN models are used. However, both categories handle the data preprocessing in different ways. These differences will be explained in the Experimental Settings section. Figure 3.2 shows, in a general way, the process followed when using the original tag dataset. The first observation is the selection of the parameters. With the one class models, the full train and test sets have to be used to determine the best parameters. This is mainly because the training set generally has only positive values. In this instance, the test set is used to determine the best model parameters. For the Multi-class models, the full dataset is used. This means that all the 2,000 examples are used, with their labels, to determine the best parameters. Both sets of models return the accuracy measurements, the parameters of the highest accuracy model are used for the experiment. One last thing to notice is that, for the tag dataset, the process will be executed several times. The dataset will be transformed using the PCA method. However, as it will be explained in the next chapter, the PCA method will not be applied in all models. The total models will be of 13. There will be eight with the original dataset, and five with the PCA method. This whole process will be repeated again but the tag dataset will be transformed using a TF-IDF method. In total, 26 models will be executed and compared.

For the text models, a similar process will be performed with a dataset containing 1000 different frequency terms. The process will transform the words frequency into a vector dataset. Then, the TF-IDF method will be performed. This final dataset will then be used to create and evaluate the eight different models. Table 3.3 shows the process of the Text models. In total, 44 models will be used and analyzed. The main objective will be to determine what is the best combination that solves the web classification problem. The Precision, Recall and F-score will be used in order to determine the highest performing model.
3.2. CLASSIFICATION MODELS

3.2.1 One-class Classifiers

In total, there were 13 different One-class classifiers. However, the parameters can be fine tuned in order to improve the model. For the majority of the parameters, a guide was used to determine the range values of the parameters [17]. Basically, the parameters were introduced with an initial range of values. For example, some attributes had a range from zero to one with increments of 0.25. The Grid Search was performed and the best model was recorded. If the final values were in the limit of the range (for example the value returned one) the range of that specific parameter was extended until the search returned a model with a lower score. For example, if the highest performing model had a parameter of 1, the grid search was performed again but with a range of one to two. Again, this was done until the best performing model was not in the limit or the accuracy was lower. In general, the parameters were bound to the specifics of the library used or the search was concluded when the model did not perform any better.

One-class Support Vector Machines

For OSVM, the *sklearn* library was used. In this classifier, three parameters were used: $\gamma$, $\nu$ and the *kernel*. To fully obtain the best possible classifier a grid search is perform with a range of these parameters. The values used were the following:

$$\text{kernel} = ["\text{poly"}, "\text{rbf"}, "\text{sigmoid"}, "\text{linear"}]$$

$$\gamma = [2^{-27}, 2^{-25}, ..., 2^{-9}, 2^{-7}]$$

$$\nu = [0.25, 0.5, 0.75, 1.0]$$

In this case, the $\gamma$ parameter was determined by the example performed in the previously mentioned guide [17]. The range of this parameter began with $2^{-15}$, $2^{-13}$, ..., $2^{-9}$, $2^{-7}$. After
the first run, the best parameter was $2^{-15}$. The second range became $2^{-23}, 2^{-21}, ..., 2^{-17}, 2^{-15}$. This process continued one more time until $2^{-27}$ became the best model since no improvements were obtained by reducing the parameter further.

The kernel parameter had only those four values that are also considered in the guide [17]. Another parameter was $C$ but the sklearn library did not consider it a parameter in the One-class version of SVM. However, $\nu$ is a specific parameter for this model. In the library documentation, the range of $\nu$ goes from (0, 1]. But, after a few executions the best parameter found was one. For this reason, no further searches were performed for the $\nu$ parameter.

**One-class K-Nearest Neighbor**

For OKNN, a python class was created. This class receives the $k$ parameter and calculates the Euclidean distance of all the training data points. Then a threshold $th$ is defined in order to determine the margin difference that will be used to determine whether a new instance is part of the class or not. The implementation of the class can be found on https://github.com/samuelistico/web-examples-repository/blob/master/OneClassModels.ipynb. For this problem, the parameter of $k$ will be the only parameter that is going to be fine tuned. This parameter represents the number of neighbors that the model will consider while assigning the label classes.

$$k = [1, 3, 5, 7, 9, 11, 13, 15, 17]$$

In this case, the search of the $k$ value was halted after 17. This is because increasing the $k$ value did not improve the overall accuracy of the model. Values like 19 to 25 returned the same accuracy as the one obtained in 17. For this reason, the final parameters for all the models were established as [1,17].
One-class Multinomial Naive Bayes

For the One-class Multinomial Naive Bayes (OMNB), a python implementation was created according to the specifications established in Chapter 2. For this class, the probabilities of each word or vector will be calculated for the entirely dataset. Since all the training data is from one class, the entire training dataset is going to be used for the calculations. The total of words are going to be fixed for the tag elements. The only parameter used is the threshold $th$ with the following values:

$$0.5, 1, 1.5, 2$$

Like the OKNN, changing the value of $th$ did not returned better results. However, this model was unaffected by increasing or decreasing the value. Overall, this model was the worst model of all the execution regardless of the parameter used.

One-class Neural Network

For the One-class Neural Network (ONN), the keras library was used to construct the NN classifier. For this case, a simple NN architecture was used. The simplest found was one called FastText [9]. Figure 3.4 shows the basic architecture of this model. However, some modifications had to be made in order to be executed correctly. The most notable one was the application of the ReLu activation function with a final softmax function. The sigmoid function was considered in order to perform a One-class model with only positive classes. However, the final results always calculated the classes as positive examples with a very low difference between instances. In the end, it was decided to use the previously mentioned architecture with low examples of negative instances in the training data. This resulted with the best possible model. Nevertheless, in the following chapters, it will be noticed this change since the previous One-class models do not have the advantage of having negative classes in their training sets.

![Figure 3.4: Architecture for the ONN.](image)
3.2.2 Multi-class Classifiers

Other 22 models were created of the Multi-class type. In general, the parameters are almost the same as their One-class counterparts. However, some other attributes were added depending on the library used. Again, this parameters were searched until the model reach an optimal.

Multi-class Support Vector Machines

For the Multi-class Support Vector Machines (MSVM), the sklearn library is used. In this classifier three parameters were used: $\alpha$, $C$ and the kernel. To fully obtain the best possible classifier a Grid Search was performed with these parameters. The values used are:

$$kernel = ["poly", "rbf", "sigmoid", "linear"]$$

$$gamma = [2^{-27}, 2^{-25}, ..., 2^{-9}, 2^{-7}]$$

$$C = [2^{-5}, 2^{-3}, ..., 2^{11}, 2^{13}]$$

For this parameters the values of $gamma$ and $kernel$ will be the same as the One-class counterpart. The $C$ value was chosen similarly as the gamma attribute. An initial range of $[2^{-5}, 2^{-3}, ..., 2^{1}, 2^{3}]$ was used. The values were modified depending on the best performing model until the value $2^{13}$ was found to be the value with the highest accuracy. It has to be pointed out that the library uses a OVO approach for the classification of this model.

Multi-class K-Nearest Neighbor

For MKNN, the sklearn library was used. Unlike the One-class counterpart, this model has additional parameters that can be used. The most time consuming was the number of neighbors. Two additional parameters were chosen in order to improve the overall results of the model without sacrificing too much memory while searching all the possible combinations. However, the best possible value of the number of neighbors was equal to one. With this there was no need for updating the values. In the case of $p$ and algorithm, these attributes are specified by the library. These values are fixed and do not require to be changed further.

$$numberofneighbours = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21]$$

$$algorithm = ["ball_tree", "kd_tree", "brute"]$$

$$p = [1, 2]$$

Multi-class Multinomial Naive Bayes

For MMNB, the sklearn library was used. In this classifier, the $\alpha$ attribute is considered. This attribute is different than the one attribute found in the One-class model. With several runs it was found that the lower the $\alpha$ value the model obtained a higher accuracy. For this reason a range was created using the following values:

$$[1.0, 0.1, 0.01, 0.001, ..., 1e - 98, 1e - 99]$$

With this setting it was quickly found that the best $\alpha$ was 1e-10.
Multi-class Neural Network

For MNN, the *keras* library was used to construct the NN classifier. The same *FastText* architecture, that was used in the One-class model, was followed. Unlike the One-class model, MNN has the advantage of having all four classes in the training set. For this reason, the only change was the modification of the hidden units in the last layer from two to four.

### 3.2.3 Text classifiers

For the Document classifiers, the plain text feature was used. The techniques used were the same as the One-class and Multi-class sections. But, the main difference is that the dataset is now transformed using a Vector method with the TF-IDF procedure. Both of these methods can be found using the sklearn library. However, the Vector method can be improved by using a collection of stop-words. These stop-words describe terms that are repetitive but do not add much context to the document. Terms found in this collection include: "The", "a", "is", etc. The *nltk* library can be used to download the collection of stop-words. For this experiment, the English and Spanish collection are used.

In this section, only 481 examples are used. This is due to the fact that some of the instances had empty text when processed. Since the empty cases vary between all four cases it was decided to limit to 481 since it was maximum number found between all four classes. With this, it was ensured that the models were trained with balanced data.

### 3.3 Experimental Setting

For the separation of the training and testing sets the following procedure was used:

1. There were four different datasets containing the four different classes and their examples.

2. These four datasets were shuffled. Then these shuffled datasets were used to create a universal dataset containing all 2000 shuffled examples (after combining, another shuffle procedure was used to ensure that the examples were disperse). This dataset was used for the Grid Search procedure on the Multi-class models.

3. With the four shuffled datasets, four shuffled training and testing sets were created. These sets were divided 70:30 having four training sets with 350 examples and four different testing sets with 150 examples. These training sets were used for training the One-class classifiers. In the case of the NN model, 36 negative classes were added to the training set.

4. For every class, the testing set (containing 150 examples) was combined with 50 examples from the other three testing sets. The final testing sets had 300 examples (being 150 from the class and 150 outliers). These final testing sets were shuffled and each one was applied in the testing phase for the One-class models.
5. A multi-class training and testing set were created by combining the four 350 training sets and the four 150 testing sets. These Multi-class sets consisted of one shuffled training set of 1400 and one shuffled testing set of 600.

Figure 3.5 shows the procedure followed for the experimental setting. This procedure was also used with the text classifiers dataset, with the variation of having 481 examples instead of 500. When applying transformations the universal set was used. After the transformation the set was divided into the four categories and the procedure was performed again. The only exception was the dataset used for the ONN. In this case, 36 extra negative classes were added to the 336 positive classes giving a total of 372 instances.

Figure 3.5: The experimental setting used for the project. This procedure describes the separation of the train and test set for all the possible models.

### 3.4 Baseline Model used for Comparison

As explained in Chapter 2, the WCAFS algorithm is used as a baseline to compare the use of the tag and text datasets. As it was described, the WCAFS algorithm returns a weighted vector that is used in a KNN model. Table 3.2 shows the initial results of using the WCAFS transformation with a simple KNN model with the number of neighbors set as 1. For the purposes of the experiment, the SVM and KNN models will be used as a baseline for each configuration. The number of attributes, or terms, used for the training was of 70000. This number was used because it yielded the best results possible for this algorithm.

### 3.5 Summary

In this chapter, the experimental setting was defined. The experiments were divided into One-class and Multi-class experiments. For the One-class, the models are going to be trained using
the positive examples of each class and they will be tested using several positive and negative examples obtained from the other classes examples. For the Multi-class experiment, all the four classes will be used for the training and testing sets. These two experiments will include the use of filter methods. An additional dataset with the TF-IDF transformation will be used as well. A final experiment is going to be applied. This last experiment will use a text dataset that will be transformed using a vector and TF-IDF transformation. The One-class and Multi-class models will use that transformed dataset as well. The precision, recall and F-score will be obtained and analyzed to determine the best combination that solves the web classification problem.

Table 3.2: Initial results of using the WCAFS algorithm in the text dataset.

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>0.94</td>
<td>0.79</td>
<td>0.86</td>
</tr>
<tr>
<td>Search</td>
<td>0.91</td>
<td>0.79</td>
<td>0.85</td>
</tr>
<tr>
<td>Form</td>
<td>0.76</td>
<td>0.96</td>
<td>0.85</td>
</tr>
<tr>
<td>Article</td>
<td>0.97</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>Average</td>
<td>0.89</td>
<td>0.88</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Chapter 4

Experiments and Results: One-Class models

This chapter describes the results of the One-Class models. The precision, recall and F-score were calculated from all four models. The filter and feature reduction methods are also shown. The best results are described by class.

4.1 Experimental Setting

For this experiment, four models were used: SVM, KNN, MNB and NN. The precision, recall and F-score are measured to determine the best fit model. This section is separated by the four classes: Login, Search, Form, and Article. This is because the models are independent from other One-class models. The dataset used to train the Login class is different than the dataset used to train the Search class. The train set consists of 350 positive examples. The test set consists of 300 examples where 150 are positive classes and the other 150 are negative. As said before, a Grid Search method was used in order to find the best possible parameters for each model. For these models, the Grid Search used the testing set in order to determine the highest accuracy. This is because the training set has only positives examples and there is no other way to measure accuracy without the use of negative classes.

In the Grid Search method the following parameters were found:

1. **SVM**: kernel = linear, gamma = 0.0078125, and nu = 1.
2. **KNN**: The number of neighbors $k$ was found to be 11.
3. **MNB**: The threshold value of $th$ had a best value of 1.
4. **NN**: No parameters are needed for this model. However, this model uses a training set with 36 negative classes in order to function as a binary model. The low classes represent less than the 10% of the total set of 372. This is to correctly represent a One-class problem were the classes are imbalanced.

The measurements of the models indicate how well the model is in classifying positive classes, negative classes and the average between both cases. In the majority, the average
is going to be used to determine the best model. However, certain positive and negative predictions will be highlighted. One last thing to remark is that the experiment was used with the TF-IDF transformation of the original dataset. Although, for the most part, the results do not improve. Additionally, the PCA method will be applied only in the NN model. This will be explained further in the PCA section.

4.2 Login

Table 4.1 shows the results of all the models applied. The best score is the NN model with 0.67. Using the TF-IDF transformation gives similar results. The KNN and MNB models do not seem to yield high results (this will be a common trend in all the experiments) and the SVM model present a lower score than the NN model. In the previous Chapter, it was established that using the WCAFS algorithm gives a F-score of 0.88. With this in mind, the results showed in this configuration do not reach the score presented in previous work.

One consideration is that the WCAFS method works in a Multi-class configuration. The One-class Login models have the limitation of working with only positive instances. The NN model does present the highest score. But, it has the advantage of being trained with limited negative instances.

<table>
<thead>
<tr>
<th>Login</th>
<th>Average</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Precision</td>
<td>Recall</td>
<td>F1-score</td>
<td></td>
</tr>
<tr>
<td>Using tag original</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.59</td>
<td>0.59</td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>0.63</td>
<td>0.51</td>
<td>0.37</td>
<td></td>
</tr>
<tr>
<td>MNB</td>
<td>0.25</td>
<td>0.50</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>0.78</td>
<td>0.69</td>
<td>0.67</td>
<td></td>
</tr>
<tr>
<td>Using tag TF-IDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.58</td>
<td>0.57</td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>0.67</td>
<td>0.51</td>
<td>0.37</td>
<td></td>
</tr>
<tr>
<td>MNB</td>
<td>0.25</td>
<td>0.50</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>0.76</td>
<td>0.68</td>
<td>0.66</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Overall results of the One-class models for the Login class.

4.3 Search

Table 4.2 shows the results of all the Search One-class models. Like the previous section, the results show that the NN model has the highest score. The main difference is that the use of tag TF-IDF yields a worse result. The KNN and MNB models are also the lowest scoring models. The SVM models have similar results with a score around 0.50.

The same pattern can be seen in this configuration. The only main difference is that the use of tag TF-IDF is worse for the NN model. In the Login class, this transformation did not affect the overall results. However, in the case of the NN model, the transformation lowers
the score from 0.60 to 0.50. This could mean that key features are lost when transforming the dataset. Some tag attributes are given a more important weight than actual key tag elements. Nevertheless, the best scoring model does not achieve the 0.88 F-score established in the baseline.

<table>
<thead>
<tr>
<th>Model</th>
<th>Using tag original</th>
<th>Using tag TF-IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>0.55          0.55</td>
<td>0.54 0.53</td>
</tr>
<tr>
<td>KNN</td>
<td>0.75 0.51</td>
<td>0.75 0.51</td>
</tr>
<tr>
<td>MNB</td>
<td>0.25 0.49</td>
<td>0.25 0.49</td>
</tr>
<tr>
<td>NN</td>
<td>0.76 0.65 0.60</td>
<td>0.52 0.52 0.50</td>
</tr>
</tbody>
</table>

Table 4.2: Overall results of the One-class models for the Search class.

### 4.4 Form

Table 4.3 shows the results of the Form models. Again, the best performing model is the NN model. However, it has the lowest scoring models so far. The SVM model performs just like the KNN and MNB models. Like the Login class, the use of the TF-IDF transformation did not change the overall results.

This is the worst performing configuration so far. The NN model presented a high score of just 0.56. This does is considerably lower than the baseline. One possibly explanation could be that the tag elements of the Form class are not descriptive enough. The Login class may have the advantage of having unique elements like `input.password`. The Form class share similar elements that the Search class. So, the model could have a difficult time finding a pattern given only positive data.

### 4.5 Article

Table 4.4 shows the results of all the Article models applied. The Article configuration is the first in having the SVM model as the highest scoring model with 0.88. The NN model has a similar score of 0.86. However, when using the TF-IDF transformation, the score is considerably lower with 0.18. The KNN and MNB models present the same scores as the previous configurations.

This is the first model that achieves similar results as the baseline. In the Article class, the best model to use is the SVM. However, all the previous classes had a lower score in the SVM models. One possibly explanation is that the tag features are considerably different in
CHAPTER 4. EXPERIMENTS AND RESULTS: ONE-CLASS MODELS

<table>
<thead>
<tr>
<th>Form</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using tag original</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.25</td>
<td>0.50</td>
<td>0.33</td>
</tr>
<tr>
<td>KNN</td>
<td>0.75</td>
<td>0.51</td>
<td>0.36</td>
</tr>
<tr>
<td>MNB</td>
<td>0.25</td>
<td>0.50</td>
<td>0.33</td>
</tr>
<tr>
<td>NN</td>
<td>0.67</td>
<td>0.60</td>
<td>0.55</td>
</tr>
<tr>
<td>Using tag TF-IDF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.25</td>
<td>0.50</td>
<td>0.33</td>
</tr>
<tr>
<td>KNN</td>
<td>0.75</td>
<td>0.51</td>
<td>0.36</td>
</tr>
<tr>
<td>MNB</td>
<td>0.25</td>
<td>0.50</td>
<td>0.33</td>
</tr>
<tr>
<td>NN</td>
<td>0.60</td>
<td>0.58</td>
<td>0.56</td>
</tr>
</tbody>
</table>

Table 4.3: Overall results of the One-class models for the Form class

the Article class than any other classes. The Article web pages present elements that are more prominent than the other three classes. One example could be the \textit{input} elements. These elements are heavily present in the Login, Search and Form classes. But, in the Article class, the \textit{input} elements are not present. Nevertheless, this class does present a good solution that reaches the score achieved in similar studies.

<table>
<thead>
<tr>
<th>Article</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Precision</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Using tag original</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.88</td>
</tr>
<tr>
<td>KNN</td>
<td>0.75</td>
</tr>
<tr>
<td>MNB</td>
<td>0.75</td>
</tr>
<tr>
<td>NN</td>
<td>0.89</td>
</tr>
<tr>
<td>Using tag TF-IDF</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.88</td>
</tr>
<tr>
<td>KNN</td>
<td>0.75</td>
</tr>
<tr>
<td>MNB</td>
<td>0.75</td>
</tr>
<tr>
<td>NN</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Table 4.4: Overall results of the One-class models for the Article class.

4.6 Inclusion of the Principal Component Analysis

For the Principal Component Analysis (PCA) method, a different procedure was followed. The PCA library can be set to transform the dimensions of a dataset at any number. Of course, the goal is to reduce the 83 existing tag attributes. For this reason, 82 datasets were created with different dimensions. For each dataset, the models had to be trained and measured. Initial runs where executed using all four models. The KNN and MNB models were not changed by the PCA method. The SVM model did have an effect. But, for the most part, the results had a score lower than 0.35 or it presented the same results than the previous models. The NN
model presented the best results when combined with the PCA method. For these reasons, it was decided to do the 82 searches, of the One-class section, only performing the NN model. The results can be viewed in Table 4.5.

By looking at the results, it can be concluded that the PCA model improves the scores of the NN models. All of the average scores are higher than their original non-PCA models. In fact, the results obtained beat the overall high scores of any class configuration. The positive and negative predictions have also been improved. The best result is a positive prediction score of 0.90 for the Article class. The best features used are very disperse in all four classes. It could be noted that, in the Article class, the number of features is considerably low compared than the others. The Search and Form models require a higher number of features. The first search model require 81 features which is not a considerable reduction than the original 83 attributes. In this setting, the best dataset to use is the tag TF-IDF since the results are the same but the number of features are less (with the exception of the Login class). This could be due to the fact that the TF-IDF method increases the value of the most considerable tag elements in the whole dataset. So, the number of key attributes can be found easier than using the original dataset. Nevertheless, the overall results are almost the same for both datasets. The PCA it is a powerful method to use. But, for this setting, it only improves with the use of NN models.

<table>
<thead>
<tr>
<th></th>
<th>Features</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Using PCA with original</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>23</td>
<td>0.79</td>
<td>0.71</td>
<td>0.69</td>
</tr>
<tr>
<td>Search</td>
<td>81</td>
<td>0.73</td>
<td>0.67</td>
<td>0.64</td>
</tr>
<tr>
<td>Form</td>
<td>76</td>
<td>0.76</td>
<td>0.69</td>
<td>0.66</td>
</tr>
<tr>
<td>Article</td>
<td>14</td>
<td>0.91</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td><strong>Using PCA with Tag TFIDF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>32</td>
<td>0.77</td>
<td>0.71</td>
<td>0.69</td>
</tr>
<tr>
<td>Search</td>
<td>40</td>
<td>0.73</td>
<td>0.66</td>
<td>0.64</td>
</tr>
<tr>
<td>Form</td>
<td>40</td>
<td>0.77</td>
<td>0.69</td>
<td>0.66</td>
</tr>
<tr>
<td>Article</td>
<td>7</td>
<td>0.91</td>
<td>0.89</td>
<td>0.89</td>
</tr>
</tbody>
</table>

Table 4.5: Overall results of the One-class models with feature reduction

### 4.7 General Discussion

In this scenario, the best results were obtained by using the Article class with scores above 0.80. Only in this configuration, the One-class models can be used for classifying web pages. But, in general, the One-class models do not provide a viable solution when compared with previous studies. The use of the TF-IDF method did not improve, in the most cases, the results. A case can be used for applying the PCA method with the NN model. This may be because the PCA method returns a new vector dataset with more significant values. Further experiments will evaluate the use of Multi-class and the text document classifications. But, for this scenario, the use of One-class heavily depends on the distinction in the classes. A quick
reminder, some elements were divided into separate features (like the \( a \) and \( input \) elements). This could be further divided to improve the overall scores of the Search and Form classes. Table 4.6 shows the selection of the best models for each class. The PCA method does provide with better results but by only using the NN model. The overall best models, for the One-class, are the NN models. The literature highly remarks the use of OSVM against non-OSVM. But, only one scenario provides a viable solution.

<table>
<thead>
<tr>
<th>Class</th>
<th>Model</th>
<th>Dataset</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>NN</td>
<td>Original</td>
<td>0.78</td>
<td>0.69</td>
<td>0.67</td>
</tr>
<tr>
<td>Search</td>
<td>NN</td>
<td>Original</td>
<td>0.76</td>
<td>0.65</td>
<td>0.60</td>
</tr>
<tr>
<td>Form</td>
<td>NN</td>
<td>tag-TF-IDF</td>
<td>0.60</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>Article</td>
<td>SVM</td>
<td>Original</td>
<td>0.88</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>Baseline</td>
<td>KNN</td>
<td>WCAFS</td>
<td>0.89</td>
<td>0.88</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 4.6: The best models selected for each class without counting for the PCA method filter.

4.8 Summary

In this Chapter, several models were created to correctly classify all four classes. For each model, the best parameters were found using a GridSearch method with four different datasets. For the Login, Search and Form classes, it was found that the best solution was the NN model. However, the results are lower when compared with the established baseline. In the end, only the Article class model was found to surpass the baseline established.
Chapter 5

Experiments and Results: Multi-Class Models

This chapter describes the results of the Multi-Class models. The precision, recall and f-score were calculated for the four models. The filter and feature reduction methods are also shown. The best results are described by each model. The results shown will be primarily compared with the results found in the One-class models.

5.1 Experimental Setting

This section, unlike the One-class chapter, the results are separated by model instead of class. Since all the classes are present in training, it is better to display the results by model instead of class. This experiment will also use the tag and text dataset. The Grid Search method will be used to determine the better parameters to use for each individual model. In this configuration, the full 2000 examples are going to be used instead in the search instead of only the testing set. Once the best parameters are determined, the dataset is divided into the training and testing sets. The PCA method is going to be display only in the models using only the original tag dataset.

5.2 Multi-class Support Vector Machines

For the Multi-class Support Vector Machines (MSVM), the best models parameters were found using the Grid Search procedure. The best parameter values are:

\[
\text{kernel} = \text{linear} \\
\text{gamma} = 7.4505e - 09 \\
C = 8192.0
\]

Similarly to the OSVM models, the best \textit{kernel} to use is the \textit{linearkernel}. The \textit{gamma} also is better with a low value. But, the penalty attribute \textit{C} throws better results with a high value. A quick reminder, the range of the \textit{C} value went from \(2^{-5}\) to \(2^{13}\) and the best value
was found to be $2^{13}$ (increasing this value did not throw better results). Table 5.1 describes the overall results of the MSVM models. The best model yields a score of 0.76. By looking at the PCA results in Table 5.2 it can be seen that the average score, by reducing the features to 62, is not improved.

The same pattern, that was found in the One-class models, can be found in the Table 5.1. This pattern consists of the Article class being the one with the highest score (0.96) and the Form class is the lowest (0.63). This concurs that the data points in the Article class are more distinct than the other classes.

The class scores presented are from the same model. That is, the same original MSVM model (presented in Table 5.1) has a score of 0.96 when predicting Article classes and a 0.63 score when predicting Form classes. But, overall the model has a 0.76 average score between all four classes. Since the same test set is used in the Baseline, the average score is the one used to determine the effectiveness of the model.

When comparing the results of the original with the TF-IDF, it can be seen that there is no significant difference. The original has a score of 0.76 and the score of the TF-IDF model is 0.74. Additionally, the PCA results of Table 5.2 have not different results. One possible explanation is that the transformation do not improves the weight of the key elements. So, a specific element has the same, or lower value, than the original set. Nevertheless, the average score of the best possible model does not reach the 0.88 score of the baseline.

<table>
<thead>
<tr>
<th>MSVM</th>
<th>Original</th>
<th>TF-IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Login</td>
<td>0.75</td>
<td>0.76</td>
</tr>
<tr>
<td>Search</td>
<td>0.68</td>
<td>0.73</td>
</tr>
<tr>
<td>Form</td>
<td>0.69</td>
<td>0.59</td>
</tr>
<tr>
<td>Article</td>
<td>0.93</td>
<td>0.99</td>
</tr>
<tr>
<td>Average</td>
<td>0.76</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Table 5.1: Results for applying SVM in a Multi-class problem.

<table>
<thead>
<tr>
<th>SVM</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Precision</td>
</tr>
<tr>
<td>Features = 62</td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>0.63</td>
</tr>
<tr>
<td>Search</td>
<td>0.70</td>
</tr>
<tr>
<td>Form</td>
<td>0.76</td>
</tr>
<tr>
<td>Article</td>
<td>0.98</td>
</tr>
<tr>
<td>Average</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Table 5.2: Overall results of the SVM model with PCA feature reduction.
5.3 Multi-class K-Nearest Neighbor

For the Multi-class K-Nearest Neighbor (MKNN), the Grid Search procedure returned the following values:

\[ \text{neighbors} = 1 \]
\[ \text{algorithm} = "\text{balltree}" \]
\[ P = 1 \]

Unlike the One-Class KNN, the number of neighbors is set to 1 as the best possible value. Table 5.3 describes the overall results of the models. The scores, on this configuration, are worse than the MSVM models. By looking at the PCA results in Table 5.4 it can be seen that the score is improved from 0.66 to 0.74. Additionally, the features are reduced from 83 to 49.

Like in previous models, the Article class is the highest scoring class. Although, in this configuration the other classes have worse scores. In this case, the PCA method has better results than the original and TF-IDF models. This may be because the KNN algorithm works with Euclidean distances. More attributes may lead to a higher distance. Thus, the model wrongly classifies the instance.

This is an important experiment because the baseline works with the same KNN model. The main difference of course is that the baseline uses the WCAFS algorithm while this uses the tag dataset. Even though the transformation and feature reduction is used, the baseline still outperforms this configuration. This could heavily indicate that the tag dataset is not a suitable replacement for the text dataset.

<table>
<thead>
<tr>
<th>MKNN</th>
<th>Original Precision</th>
<th>Original Recall</th>
<th>Original F1-score</th>
<th>TF-IDF Precision</th>
<th>TF-IDF Recall</th>
<th>TF-IDF F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>0.57</td>
<td>0.61</td>
<td>0.59</td>
<td>0.65</td>
<td>0.69</td>
<td>0.67</td>
</tr>
<tr>
<td>Search</td>
<td>0.60</td>
<td>0.51</td>
<td>0.55</td>
<td>0.64</td>
<td>0.54</td>
<td>0.58</td>
</tr>
<tr>
<td>Form</td>
<td>0.64</td>
<td>0.55</td>
<td>0.59</td>
<td>0.73</td>
<td>0.48</td>
<td>0.58</td>
</tr>
<tr>
<td>Article</td>
<td>0.81</td>
<td>1.00</td>
<td>0.90</td>
<td>0.70</td>
<td>1.00</td>
<td>0.82</td>
</tr>
<tr>
<td>Average</td>
<td>0.66</td>
<td>0.67</td>
<td><strong>0.66</strong></td>
<td>0.68</td>
<td>0.68</td>
<td><strong>0.66</strong></td>
</tr>
</tbody>
</table>

Table 5.3: Results for applying Multi-class KNN.

5.4 Multi-class Multinomial Naive Bayes

For the Multi-class Multinomial Naive Bayes (MMNB), the Grid Search procedure returned an \( \alpha \) value of 1e-10. The PCA method could not be executed in this experiment due to the fact that the library did not accept negative values. However, as seen in the previous sections, the use of filter methods do not always generate a better result for the MNB model. But, for this part, the comparison of the model will be limited to the non-PCA results of the other models. Table 5.5 describes the overall results of the MMNB models. These results have the
Table 5.4: Overall results of the KNN model with PCA feature reduction.

The lowest scores compared with the other Multi-class models (even without considering the PCA results). The best possible score is 0.63 with the original dataset.

A different pattern is present in these results. The lowest scoring class is the Search class with 0.52 and 0.49. In the previous models, the Article class was able to achieve a score close to 0.90 or higher. However, in this section, the Article score is below 0.80. Thus, the MMNB is the worst Multi-class configuration so far. However, as it will be seen in the next chapter, the MNB may be useful when using the text dataset.

One possibly explanation for the low scores could be that the tag dataset is not well suited for this particular model. To explain further, the MNB model works with the probability of a class given the probability of its attributes. In the text dataset, there could be more than 1000 different attributes or terms. Some classes may have the advantage of a specific term to only appear in a specific class. However, the tag dataset always has 83 attributes among all the classes. The calculation of the probabilities may not be significant because all the classes may have some frequency among all terms. However, as it will be seen in Chapter 6, this model is significantly improved in the text dataset.

Table 5.5: Results for applying Multi-class MNB models.

5.5 Multi-class Neural Network

Table 5.6 describes the overall results for the Multi-class Neural Network (MNN) using the original and tag TF-IDF dataset. The overall training procedure took around 200 epochs with a validation split of 0.2 for the training set. In this experiment, several executions were made in order to improve the overall accuracy of the model. However, by using the test dataset it was found that the best possible score was around 76%. The training accuracy reached
over 90% but the validation score was similar to the testing score with 77%. Looking at the overall results it can be seen that the model presents the same behavior as previous models (the Article class has the highest score, while the Form class is the lowest). The one thing to notice, is that the Article class has the highest score so far for the Multi-class models with a value of 0.97. By looking at the PCA results in Table 5.7, it can be seen that the average score is slightly improved from 0.74 to 0.77.

This model presents the best possible score by using the PCA method. Without counting the score of the One-class Article models, the score of this PCA MNN model has the highest value than any other model so far. The combination of PCA is very well suited for this scenario (like its ONN counterpart). For this reason, it can be concluded that the PCA method can significantly improve the results of the NN models. The MSVM model present a score of 0.76, which is almost the same as this particular model. For that reason, the use of SVM for Multi-class problems should not be discarded. Although, the baseline score is still not reached with this configuration.

<table>
<thead>
<tr>
<th>MNN</th>
<th>Original</th>
<th>TF-IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Login</td>
<td>0.70</td>
<td>0.67</td>
</tr>
<tr>
<td>Search</td>
<td>0.69</td>
<td>0.71</td>
</tr>
<tr>
<td>Form</td>
<td>0.62</td>
<td>0.60</td>
</tr>
<tr>
<td>Article</td>
<td>0.94</td>
<td>0.99</td>
</tr>
<tr>
<td>Average</td>
<td>0.74</td>
<td>0.74</td>
</tr>
</tbody>
</table>

Table 5.6: Results for applying Multi-class NN models.

<table>
<thead>
<tr>
<th>NN</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Precision</td>
</tr>
<tr>
<td>Features = 71</td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>0.79</td>
</tr>
<tr>
<td>Search</td>
<td>0.70</td>
</tr>
<tr>
<td>Form</td>
<td>0.64</td>
</tr>
<tr>
<td>Article</td>
<td>0.94</td>
</tr>
<tr>
<td>Average</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Table 5.7: Overall results of the NN model with PCA feature reduction.

5.6 General Discussion

The first thing to notice, is the fact that the Multi-class models have the advantage of being trained with a balanced dataset with all four classes. The results overall surpass the scores of the previously mentioned One-class models except for the Article class. Table 5.8 shows the summary of the results of all the configurations. As it can be seen, the best models use the Original tag dataset, without the TF-IDF transformation, only two instances are improved by
the use of the PCA method. None of the models surpass the Article One-class model. It also has to be pointed out that none of the models reach the score established at the baseline. The Multi-class models were expected to have a higher accuracy and be the closest to the baseline score. However, these scores do not prove that the tag dataset can be used as an optimal solution for the web classification problem.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>Reduction</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSVM</td>
<td>Original</td>
<td>None</td>
<td>0.76</td>
<td>0.77</td>
<td>0.76</td>
</tr>
<tr>
<td>MKNN</td>
<td>Original</td>
<td>PCA(49)</td>
<td>0.74</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td>MMNB</td>
<td>Original</td>
<td>None</td>
<td>0.65</td>
<td>0.65</td>
<td>0.63</td>
</tr>
<tr>
<td>MNN</td>
<td>Original</td>
<td>PCA(71)</td>
<td>0.77</td>
<td>0.77</td>
<td>0.77</td>
</tr>
<tr>
<td>One-class Article</td>
<td>SVM</td>
<td>Original</td>
<td>0.88</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>Baseline</td>
<td>KNN</td>
<td>WCAFS</td>
<td>0.89</td>
<td>0.88</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 5.8: The best models selected for each Multi-class configurations. The last two rows show the results of the best One-class Article model and the baseline.

5.7 Summary

In this Chapter, the Multi-class models were evaluated. Unlike the One-class models, the results shown in this Chapter were displayed by model and not by class. Again, the best parameters were found by using the GridSearch method with all the 2000 instances. The results of these Multi-class models do not achieve the baseline score established. Thus, it was concluded that the tag dataset is not a suitable alternative when trying to solve the web classification problem.
Chapter 6

Experiments and Results for Document Classification

This chapter describes the results of the use of the text dataset using the TF-IDF method with the One-class and Multi-class models. For this section the same procedure of the previous chapters were applied. However, the main difference is the transformation of the text dataset.

6.1 Experimental Setting

As said in chapter 3, the text dataset is used for this experiment. This dataset consist of the plain text found in the 2000 web pages. However, there are cases where the plain text is empty. For this reason, not all 2000 examples were used. By checking the four classes it was found that the maximum amount of examples that could balance all classes was 481 for each class. That is, for every class there are 481 examples giving a total of 1924 in all four. When dividing these datasets in training and testing the same 70:30 proportion was used. For the case of One-class, the training set consisted of 336 positive examples. The test set consisted of 145 positive examples and 150 negative examples (50 for the rest of the three models) giving a total of 295 test examples. in the case of Multi-class, the total of training size was of 1344 and the testing size was 580. Except for the split difference on the dataset, the rest of the procedure was followed like it was described in Chapter 3.

When pre-processing the dataset, it was necessary to transform the text document into numerical vectors. The `sklearn` library of Vectorizer was used in order to do so. This library took into consideration a collection of stop-words that will reduce the total terms of the vector in order to increase accuracy. The collection was gathered via a `nltk` library. The English and Spanish stop-word collections were downloaded because the majority of the examples were in those languages. When transforming the dataset into term vectors, over 90,000 different attributes were found for all the 1344 examples. To reduce the training time only, the most frequent 1000 terms were used. The number of attributes will not affect much the results since the previously models only worked with 83 attributes. As it will be seen, the results will not vary much with the previously known One-class models and it will give insight into how to improve those other models. Finally, the TF-IDF transformation was used to give more importance to specific words in the documents.
The following sections will be divided into the One-class and Multi-class categories. In this scenario, filter methods are unused since the Vectorizer creates the number of attributes based on the most frequent ones. If a feature reduction is needed, then the number of terms are applied in the Vectorizer library. Again, for each model a Grid Search procedure was used in order to find the best parameters.

### 6.2 One-class

For the first three models the Grid Search found these best possible values:

1. **OSVM**: The *kernel* was set as linear (like the previously models). Also, *nu* was set with a value of 1. Lastly, the gamma attribute was set to 0.0078125.

2. **OKNN**: The number of neighbors *k* was set as 11.

3. **OMNB**: The threshold value of *th* was set as 1.

The parameters values are the same as the ones found in the original dataset experiment in Chapter 4. Table 6.1 shows the results of applying the One-class models to the text dataset. The same pattern that was discovered in the previous chapters can be found in the Login class. The better model is NN closely followed by SVM; and the worst model is still being MNB.

The rest of the classes present a different pattern. In the other three, the best performing model is SVM followed by the KNN model. In the last configuration, the NN model performs worse than the MNB model. Overall, the best score was with the SVM model with a score of 0.82 for the Article and Login class.

With the increment of attributes it could be expected that these models perform better than the others in the original dataset experiment. However, the NN model seem to be worse with these amount of attributes. The Article class, which supposedly is the easiest to identify, has a score of 0.30 for the use of the NN model. In Table 6.2, the best results among all the models are presented. If these results are compared with the results of the One-class configuration, it can be seen a general improvement. However, none of the text dataset models achieve a score of 0.88 like the baseline. The score of the Article class is lower than the tag dataset with 0.82. But, this is the only instance, the other three classes have better scores than the ones with the tag dataset. Wit this, it is difficult to conclude that the tag dataset is a suitable alternative than the traditional text dataset.

### 6.3 Multi-class

For the first three models the Grid Search found the best possible values:

1. **MSVM**: The *kernel* was set of *linear* (like the previously models). The *gamma* attribute was set to $7.4505e-09$ (again like the previous models). The *C* parameter was set to 2048 (this is the only value that differs).
### 6.3. MULTI-CLASS

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Login Class</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.83</td>
<td>0.82</td>
<td><strong>0.82</strong></td>
</tr>
<tr>
<td>KNN</td>
<td>0.67</td>
<td>0.60</td>
<td>0.55</td>
</tr>
<tr>
<td>MNB</td>
<td>0.24</td>
<td>0.49</td>
<td>0.32</td>
</tr>
<tr>
<td>NN</td>
<td>0.86</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td><strong>Search Class</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.79</td>
<td>0.78</td>
<td><strong>0.78</strong></td>
</tr>
<tr>
<td>KNN</td>
<td>0.66</td>
<td>0.61</td>
<td>0.57</td>
</tr>
<tr>
<td>MNB</td>
<td>0.24</td>
<td>0.49</td>
<td>0.32</td>
</tr>
<tr>
<td>NN</td>
<td>0.40</td>
<td>0.40</td>
<td>0.40</td>
</tr>
<tr>
<td><strong>Form Class</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.78</td>
<td>0.75</td>
<td><strong>0.74</strong></td>
</tr>
<tr>
<td>KNN</td>
<td>0.71</td>
<td>0.60</td>
<td>0.53</td>
</tr>
<tr>
<td>MNB</td>
<td>0.24</td>
<td>0.49</td>
<td>0.32</td>
</tr>
<tr>
<td>NN</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td><strong>Article Class</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.87</td>
<td>0.83</td>
<td><strong>0.82</strong></td>
</tr>
<tr>
<td>KNN</td>
<td>0.77</td>
<td>0.57</td>
<td>0.46</td>
</tr>
<tr>
<td>MNB</td>
<td>0.24</td>
<td>0.49</td>
<td>0.32</td>
</tr>
<tr>
<td>NN</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
</tr>
</tbody>
</table>

Table 6.1: Overall results of the One-class models using the text dataset.

2. **MKNN**: The number of *neighbors* was set as 21. The *algorithm* as *balltree*. Finally, the *p* value was set as 2. With the exception of the algorithm parameter, the variables differ from the ones found in Chapter 5.

3. **MMNB**: The *alpha* value was set to 0.001 being different than the one found previously.

Table 6.3 shows the results of applying all the models, like the tables shown in Chapter 5, the results are displayed by class. The considerable result to notice, is that the NN model has the lowest score of 0.37. The other three models have considerably good results, the model with the biggest score is the MNB with 0.91. Closely followed is the SVM model with 0.90. Even the KNN has a high score with 0.85 (compared with previous KNN results). With this it can be concluded that the best way to classify these web pages is with the use of the text dataset.

This conclusion it is known by looking at the literature. However, one aspect to notice is the fact that the baseline uses 70000 words for the training of the model. All of the models presented in Table 6.3 are trained using only 1000 terms. One could give the argument that applying the WCAFS algorithm is not necessary in this scenario. Two models (SVM and MNB) surpass the baseline score and the KNN model has a close score of 0.85. The only model that do not achieve a high score is the NN model. This can be solved by making the model more complex, the previous model worked better with lower attributes. Probably with a more complex network and training time this model would achieve the scores the other
models have. However, is not necessary since the SVM and MNB models are easier to train and achieve a score beyond the baseline.

This configuration was expected to perform better. Looking at the results of the tag dataset it can be seen that the F-scores are considerably different. With this, it is difficult to conclude that the tag dataset is an alternate solution for the we classification problem.

<table>
<thead>
<tr>
<th>Class</th>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>SVM</td>
<td>0.83</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>Search</td>
<td>SVM</td>
<td>0.79</td>
<td>0.78</td>
<td>0.78</td>
</tr>
<tr>
<td>Form</td>
<td>SVM</td>
<td>0.78</td>
<td>0.75</td>
<td>0.74</td>
</tr>
<tr>
<td>Article</td>
<td>SVM</td>
<td>0.87</td>
<td>0.83</td>
<td><strong>0.82</strong></td>
</tr>
<tr>
<td>Tag dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>NN</td>
<td>0.78</td>
<td>0.69</td>
<td>0.67</td>
</tr>
<tr>
<td>Search</td>
<td>NN</td>
<td>0.76</td>
<td>0.65</td>
<td>0.60</td>
</tr>
<tr>
<td>Form</td>
<td>NN</td>
<td>0.60</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>Article</td>
<td>SVM</td>
<td>0.88</td>
<td>0.88</td>
<td><strong>0.88</strong></td>
</tr>
<tr>
<td>Baseline</td>
<td>KNN</td>
<td>0.89</td>
<td>0.88</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 6.2: The best One-class models compared with the best models in the tag dataset configuration.

### 6.4 General Discussion

The results of this experiment show what was found in the literature. Having a text dataset, a transformation with the TF-IDF can be used in conjunction with the SVM and MNB models in order to correctly classify all the classes. The frequency of the words are better suited for giving context of a document. However, there are some considerations to be made. First, the better performing models could be attributed to the increment in total attributes in the dataset.
Second, it is known that these models could be improved by using other reduction procedures like removing the stop-words. Probably, with the use of techniques to improve and split the tag dataset, the scores found in the original dataset can be improved as well. Nevertheless, if a testing team has the ability to generate a plain text dataset from web pages, it could lead to better performing results than using the tag based dataset. However, the NN model should be changed, because it was the only model that performed better in the previous chapters.

6.5 Summary

In this Chapter, the One-class and Multi-class models were tested using the text dataset with a vector of 1000 unique words with their frequency. These models were expected to perform better than the other models. This was the case for the Multi-class models that surpassed the baseline score. It was concluded that the text dataset is still the best solution when trying to classify these four classes.
Chapter 7

Conclusions

In this thesis, the goal was to solve the web classification problem with the motivation of being applied in testing scenarios. To do this, a dataset of 2000 examples were created and four basic classes were defined each containing 500 different examples. For this purpose, the classes were defined as the most basic options of a testing scenario.

The known literature had the classification techniques of SVM as the best known method to solve this particular scenario. However, the One-class models were chosen due to their ability to be trained with limited positive data. For comparison, the Multi-class models were also applied in order to determine if the DM techniques could solve the web classification problem or at least determine how well it performed. The previous related work used an algorithm called WCAFS with a F-score of 0.88.

Another particular element for this thesis was the use of tag elements similarly to word vectors in a text classification problem. The plain text of the 2000 examples were also obtained in order to compare if the tag dataset could be used instead of a text dataset. The final models to be used were: SVM, KNN, MNB and NN (with their One-class and Multi-class configurations). The PCA feature reduction method was applied to see if the overall scores were improved. Finally, this whole process was also applied with the tag dataset but with a TF-IDF transformation in order to see if there was an improvement.

For the first experiment, the One-class models were applied with the tag dataset. In this configuration the tag TF-IDF transformation did not return favorable results. The best performing model was ONN. A pattern was found in all the models applied. The Article class was almost always the best performing class in all the models while the lowest was the Form class. This could be attributed to the fact that the Search and Form classes have similar important attributes. The Article class was very different than the other three. So, it could be expected to be the best performing class. In this configuration, the only model that achieved the baseline score was the One-class Article with 0.88.

For the second experiment, the Multi-class models were applied with the tag dataset. Again, the tag TF-IDF transformation was of no use. Since these models are trained with all the four classes, the final results were divided by each model. Overall, the best performing model was the MNN with the use of PCA transformations. However, none of the models achieved the baseline score of 0.88. This could be attributed to the low attributes in the tag dataset. One model so far, the One-class Article, had achieved the baseline score. With these results it could not be concluded that the tag dataset could be used in the web
CHAPTER 7. CONCLUSIONS

classification problem.

For the last experiment, the One-class and Multi-class models were applied with the text dataset. A vector and TF-IDF transformation was used in order to create a training and testing set with 1000 attributes. This was a considerably change since the previous experiments only had 83 attributes. In this scenario, the One-class and Multi-class models had better scores than their counterparts in the other experiments. However, the NN models had a worst score. Finally, it was concluded that the best way to solve the web classification problem is to use the text dataset with a Multi-class model. Only the MSVM and MMNB achieved a score that surpassed the baseline. Additionally, these models were only trained with 1000 terms instead of 70000 that were used for the baseline experiment.

In conclusion, there is no sufficient evidence to support the use of the tag dataset instead of the text dataset. In a testing scenario, as the one described in Chapter 1, the best possible solution is to determine all the classes and create a dataset with the plain text of the web pages. If the classes are well defined and the number of examples are balanced, then the MSVM, MMNB and MKNN models could be used to correctly classify the instances. The only model that achieved a good score was the One-class Article class. However, the rest of the models (both One-class and Multi-class) did not reach a score equal to or greater than the one established previously.

7.1 Future work

This experiment should be replicated with web pages who have a similarity between them. One possible way to do this is by applying these models on corporate web pages with a similarity of over 90%. This could determine if the tag dataset works better with web pages who are in the same domain. It is believed that the use of the tag elements could have been improved if the known attributes were separated into different feature elements. One way to do it is by looking at the \texttt{INPUT} element, which was separated into multiple elements depending on a specific value (for example the type). In the text classification problem, the stop-words collection and the vector method played an important part in the experiment. So, a similar procedure could be followed with the tag elements. The most common tag elements could be removed and similar elements could be united. The tag elements could be further separated using their inner attributes and it is believed that more important attributes could be added to the dataset. A project could be started to try to find the best possible combination of tag elements to improve the models described in this thesis. However, the best model to follow is the SVM model since it performed well in all the experiments. Overall, the web classification problem can be further improved based on how the dataset is created and how the attributes and classes are defined.
Bibliography


Master student was born in Monterrey, México, on January 4, 1993. He earned the Technologies Engineering degree from Universidad de Monterrey in December 2015. He was accepted in the graduate program in Computer Sciences in January 2018.