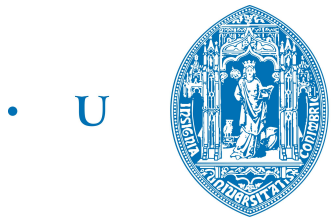


FACULTY OF SCIENCES AND TECHNOLOGY
UNIVERSITY OF COIMBRA

INTEGRATED MASTER DEGREE IN ELECTRICAL AND COMPUTERS ENGINEERING

Web Page Classification using Visual Features

Antonio Miguel Baptista Videira

A Thesis Submitted In Partial Fulfillment
of the Requirements For the Master Degree of
Electrical and Computers Engineering

Jury Members:

President: Dr. Professor Vitor Manuel Mendes da Silva
Supervisor: Dr. Professor Nuno Miguel Mendonça da Silva Gonçalves
Vogal: Dr. Professor João Pedro de Almeida Barreto

Coimbra, September 2013

Acknowledgments

After six years of ups and downs, I would like to thank many people who helped and supported me during this period of time.

To my parents, I am deeply grateful for the patience, the effort and sacrifices made to gave me the opportunity to attend university and to become what I am now. Greetings also to the rest of my family for many nice talks, which helped in compensating much of the stressful and tedious working effort during this time.

I would like to express my appreciation to Professor Nuno Gonçalves, for giving me the opportunity to write this thesis, for his attendance and availability, his rigor, support and help in all stages of this thesis.

I would like to acknowledge my immense gratitute to my friends and colleagues for the help and support in the conduct of this work. For the good moments of cheerfulness, and that since my entry in Coimbra always accompanied me. Thus providing me one of the best phases of my life.

To all my sincerest thanks and appreciation,

António Videira

Resumo

Com o constante aumento no número de utilizadores na Internet, o crescimento de websites também tem vindo a crescer proporcionalmente. Existe cada vez uma maior exigência de técnicas de classificação automática com maior precisão. Desta maneira a classificação de páginas web, tornou-se um grande tema de pesquisa nos últimos anos. Para classificar e processar páginas web automaticamente, os sistemas actuais usam o conteúdo de texto dessas páginas, que inclui o conteúdo exibido e o código HTML adjacente. No entanto, até agora, pouco trabalho de investigação tem sido desenvolvimento usando o conteúdo visual das páginas web para realizar a classificação.

Tendo em conta isso, nesta tese concentramo-nos em realizar classificação de páginas web usando o seu conteúdo visual. As páginas web podem apresentar informação visual diferente e variada dependendo do seu tópico específico. Neste trabalho foi construído um sistema de classificação para permitir a análise automática do aspecto visual da página como aparece para o utilizador.

Primeiro, um descritor correspondente a cada página é construído, através da extracção de diferentes características da própria. As características usadas são o histograma de cor e arestas, e as características de Gabor e textura. Em seguida, dois métodos de seleção de características são aplicados a esse descritor, para selecionar um subconjunto das suas características mais relevantes. O primeiro baseado no critério do Chi-Quadrado e o segundo usando a Análise dos Componentes Principais.

Outra abordagem utilizada, envolve o uso do Bag of Words (BoW) que considera as características locais SIFT extraídas de cada imagem como palavras, permitindo a construção de um dicionário. Depois é possível descrever novas imagens extraindo essas características locais e ver a correspondência com as características no dicionário que são mais próximas.

De seguida classificamos as páginas web com base no seu valor estético, o seu valor temporal (atualidade no design) e o respectivo tópico da página. Os métodos de aprendizagem de máquina utilizados são o Naïve Bayes, Support Vector Machine, Decision Tree e AdaBoost. Diferentes testes são realizados para avaliar o desempenho de cada classificador em cada experiência.

Investigando em detalhe somos capazes de tirar conclusões gerais sobre se o conteúdo visual deve ser ignorado quando realizada a classificação de páginas de web. A principal vantagem da nossa abordagem é a boa precisão em cada experiência.

Palavras-chave: ***Classificação de página web, Extração de características, Selecção de Características, Aprendizagem Máquina.***

Abstract

With the increase in the number of Internet users, the growing of websites is proportional, thereby web page classification has become a huge topic of research in the last few years. There is a constantly increasing requirement for automatic classification techniques with greater classification accuracy. To automatically classify and process web pages, the current systems use the textual content of those pages, which includes both the displayed content and the underlying HTML code. However, until now, little work has been done on using the visual content of a web page to perform classification.

On this account, in this thesis we focus on performing web page classification using their visual content. The web pages can present different and varied visual information depending on their specific topic. In this work I build a classification system to enable automatic analysis of a web page visual appearance as it appears to the user.

First a descriptor is construct, by extracting different features from each page. The features used are the simple color and edge histogram, Gabor and texture features. Then two methods of feature selection, one based on the Chi-Square criterion, the other on the Principal Components Analysis are applied to that descriptor, to select the top attributes.

Another approach involves using the Bag of Words (BoW) model to treat the SIFT local features extracted from each image as words, allowing to construct a dictionary. Then it is possible to describe new images by extracting the local features from them and matching them with features in the dictionary which are closest.

Then we classify web pages based on their aesthetic value, their recency and type of website. The machine learning methods used in this work are the Naïve Bayes, Support Vector Machine, Decision Tree and AdaBoost. Different tests are performed to evaluate the performance of each classifier in each experiment.

And by investigating our approach in detail, we are able to draw general conclusions and statements about whether or not the visual content should be ignored when performing web page classification. The main advantage of our approach is the good accuracy in each experiment.

Keywords: ***Web Page Classification, Feature Extraction, Feature Selection, Machine Learning***

”Success comes before
work only in the dictionary.”

Vince Lombardi

Contents

List of Tables

List of Figures

Nomenclature

Acronyms

WPC	*Web Page Classification*
HTML	*HyperText Markup Language*
EHD	*Edge Histogram Descriptor*
SIFT	*Scale Invariant Feature Transform*
PCA	*Principal Components Analysis*
Chi2	*Chi-Square*
SVD	*Singular Value Decomposition*
BoW	*Bag of Words*
ML	*Machine Learning*
SVM	*Support Vector Machine*
TPR	*True Positive Rate*
FPR	*False Positive Rate*

Chapter 1

Introduction

Over the last years, the world has witnessed a huge growth on the internet, with millions of web pages on every topic easily accessible through the web. A web page is a web document that is suitable for the World Wide Web, that with the help of a web browser we are able to retrieve, present and scroll that web page. When a web address is typed in the browser, what appears on the screen is called the web page, and a website is basically a collection of web pages.

When a website is developed, it is important to understand why people visit them in order to maximize the website objective. The internet can be accessed almost anywhere by numerous means, allowing to do research or get information. Another use of the internet is to buy something or to compare a variety of services and products. Basically a researcher or a visitor can easily become a customer of a website that makes a good case explaining why is it better and different from the other competitors. Thereby the owner of a website creates the site for his visitors, for passing the information or offering a service, in a way that makes sense even to an outsider. Using the website as a mean of communication, it is only successful when its message is received by the intended user.

Since the first websites in the early 1990's, designers have been innovating the way websites look. The visual appearance of a web page, influences the way the user will interact with this same page. The structural elements of a web page (e.g., text, tables, links, images) and characteristics (e.g., color, size) are used to determine the visual presentation and level of complexity of a page. This visual presentation is known as Look and Feel, which is one of the most important properties of a web page.

The Look and Feel of a website affects the way the user perceives it and what he/she feels just by looking at it. The visual aspects can make a website more appealing, authentic, credible, entertaining and much more. Depending on the information that a website is trying to pass to the user, specific topics require a specific design that matches the purpose of the site, or the nature of the product it tries to sell. A site belonging to creative companies, should have some extravagance in their design that makes their brand stands out, at the same time it must be authentic and trustworthy. Corporate companies, banks or insurance firms websites, tend to make an official impression. A more serious design, clean

layout, clean structure and fresh colors indicate trust and quality giving the website a more official look. The design of a website about a restaurant or an hotel must be eye-catching and inviting. Basically, the designer have in mind that the design is focus on the website topic and use his creativity to pass to the users emotions, trust and impressions about the website. Since each website should have a distinct visual aspect that depends on the information or service that is trying to provide, it is easily verified that each web page differs in their visual characteristics.

As mentioned above, with the growing of the internet, millions of web pages are easily accessible, making the web a huge repository of information and thus there is a need for categorizing web documents to facilitate the indexing, searching and retrieving of pages.

Tools like search engines help the users to locate information on the internet. Search engines have an excellent performance in finding information, but are limited in the ability of organizing the web pages. That's why users are interested in tools that help select and make a quick selection of the information that is needed. In order to achieve web's full potential as an information resource, the vast amount of content available in the internet has to be well described and organized. That's why automation of web page classification (WPC) is useful. WPC helps in focused crawling[1]*, assists in the development and expanding of web directories (for instance Yahoo), helps in the analysis of specific web link topic, in the analysis of the topical structure of the web, improves the quality of web search (e.g., categories view, ranking view), web content filtering, assisted web browsing and much more.*

By doing a web page classification we want to assign web pages to one or more predefined categories. A typical web page classification process consists in the following steps: extraction of features from training web pages, creation of a feature vector for each web page, reducing the dimensionality of the feature vectors if necessary, create a classifier, by learning over the training feature vectors and finally, classify new web pages using that classifier.

1.1 Aims and objectives

The visual appearance (Look and Feel) of each website is constructed using colors and color combinations, type fonts, images and videos, and much more. The aim of this dissertation is to enable automatic analysis of this visual appearance of web pages by using the web page as it appears to the user and evaluate the performance of different classifiers in the classification of web pages in several tasks.

The motivation behind this dissertation is based on [40], where the authors proved that by using generic visual features was possible to classify web pages for several different types of tasks. They classify web pages based on their aesthetic value, their recency and the type of website, and produce good classifications results around 70-80% of accuracy. First they extract low-level features from each page using the Lire image feature library [26] for content-based image retrieval, and construct a descriptor

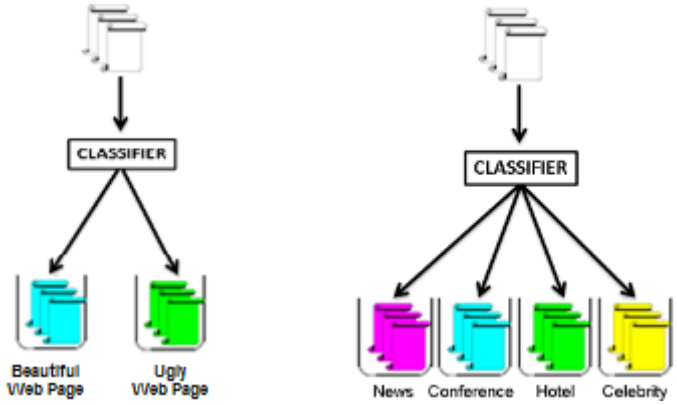
[1] The goal of a focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics.

(feature vector) that characterizes each page. Then using the chi-square criterion they select the most relevant features. And finally with the WEKA toolkit [16] two classifiers are constructed, the Naive Bayes and a decision tree learning algorithm (J48). They conclude that by using low-level features of web pages, it is possible to distinguish between several classes that vary in their Look and Feel, in particular aesthetically well designed vs. badly designed, recent vs. old and different topics.

Our objective is to use their proposed method of using visual features to classify web pages for several different types of tasks, and obtain a better rate of classification success. By extracting a visual feature descriptor from each web page, we evaluate the performance of different classifiers for each task (aesthetic value, their recency and the web page topic (category)).

Based on the number of classes in the problem, classification can be divided into binary classification and multi-class classification. In this work we have two binary classifications and one multi-class classification that deals with more than two classes. Binary classification (aesthetic value and recency), categorizes instances into exactly one of two classes (as in Figure 1.1(a)). In multi-class classification (web page topic), e.g., four-class classification, meaning that four classes are involved, say News, Conference, Hotel and Celebrity, exactly one class label can be assigned to an instance (as in Figure 1.1(b)).

The algorithms were developed in C/C++ using the OpenCV library [5], that runs under Windows, Linux and Mac OS X.



(a) Binary (b) Multi-Class

Figure 1.1: Types of classification

1.2 State-of-the-art

In web page classification also known as web page categorization, many algorithms have been adopted to classify web pages. Web pages are semi-structured text documents usually written in HTML, that contain additional information, such as HTML tags, hyperlinks and anchor texts. The features that are useful in web page classification can be classified into: on-page features, which are directly located

on the page to be classified, and features of neighbors, features that are found on the pages related in some way with the page to be classified [29].
The general problem of web page classification can be divided into multiple sub-problems:

1. *Subject classification concerns about the subject or topic of a web page (i.e., arts, business, sports, celebrity).*
2. *Functional classification refers to the role that the web page plays (e.g., personal home page, course page or admission page).*
3. *Sentiment classification focuses on the opinion that is presented in a web page, (i.e., the authors attitude about some particular topic).*
4. *Other types of classification. This include genre classification (e.g., [12]), search engine spam classification (e.g., [7]) and much more.*

The textual content that is directly located on the page, is the On-page features most used in the WPC. In order to retrieve important textual features, the web pages were preprocessed to discard the less important data. The preprocessing consists in: removing HTML tags (tags indicate the format of web pages), removing stop words, removing rare words or performing word stemming (in most cases a stemming algorithm is applied to reduce words to their basic stem[2]). After the preprocessing, a selection of features represent each web page.

Using information derived from HTML tags can boost the classifiers performance. Golub and Ardö [15] determine how significance indicators assigned to different web page elements (internal metadata, title, headings and main text) influence automated classification. They showed that the best results are achieved from a well-turned linear combination of those four elements. Kwon and Lee [21] proposed a web page classifier based on an adaptation of k-Nearest Neighbor (k-NN) approach with a feature selection method and a term-weighting scheme, where terms within different tags are given different weights.The intuition is that frequently co-occurring terms constrain the semantic concept of each other. The more co-occurred terms two documents have in common, the stronger the relationship between the two documents.

For having a good quality document summarization, Shen et al. [34] proposed an approach to classify web pages topic through web page summarization algorithms for extracting the most relevant features from web pages. They were able to show an improvement in the accuracy as compared with content based classifiers.

Kan and Thi [19] and Min-Yen Kan [18] perform classification using the URL of the web page, without the necessity of downloading the page, an approach faster than the typical web page classifiers, that is useful when the page content is not available.

Holden and Freitas [17] using a classification algorithm (Ant-Miner), i.e., the first Ant Colony Optimisation (ACO) algorithm for discovering classification rules, allowed them to use that set of rules to classify web pages based on their subject.

[2] It is the base part of a word that has prefixes or suffixes.

Qi and Davison [30] were able to assign a web page to one or more predefined category labels, and point out that the appropriate use of textual and visual features that reside directly on the page can improve classification performance. Feature selection and the combination of multiple techniques can bring further improvement.

A web page classification method presented by Selamat and Omatu [33] used a neural network with inputs based on the Principal Component Analysis and class profile-based features. By selecting the most regular words in each class and weighted them, and with several methods of classification, they were able to demonstrate an acceptable accuracy. Chen and Hsieh [9] proposed a WPC method using a SVM based on a weighted voting scheme. This method uses Latent semantic analysis to find relations between keywords and documents, and text features extracted from the web page content.Those two features are then sent to the SVM model for training and testing respectively.Then based on the SVM output, a voting scheme is used to determine the category of the web page.

The approach presented by Fiol-Roig et al. [13] shows that by using data mining techniques to construct the classifiers, allows to managed classifications about the subject or topic of a web page. Showing that a web page classification based on web page features available in the HTML code pages is possible.

Jan Ulrich [39] developed a classification scheme for both single-class and multi-class. The classifiers were constructed without using the negative examples, making it more practical. For the single-class, the PEBL algorithm was used to build the classifier, using only the positive examples. The multi-class classifier, demonstrates better results using Logitboost rather than SVM.

The authors Wibowo and Williams [42] proposed a method of hierarchical classification of web pages. The approach was based on the assumption that a summary is present at the beginning of each document. By using small numbers of features from pre-categorized training documents, categorization accuracy can be substantially improved.

All the approaches above describe the documents on the web based on data that is extracted from the HTML code. Instead of text representation written in HTML, a web page is also represented by the visual representation rendered by the web browser. This is the approach that is more related to our project, but there are not many studies of WPC using the visual information of a web page, because that information is usually ignored.

A WPC approach based on the visual information was implemented by Asirvatham et al. [3], where a number of visual features, as well as textual features, were used in the web page classification. They proposed a method for automatic categorization of web pages into a few broad categories based on the structure of the web documents and the images presented in it. Another approach was proposed by Kovacevic et al. [20], where a page is represented as a hierarchical structure – Visual Adjacency Multigraph, in which nodes represent simple HTML objects, texts and images while directed edges reflect spatial relations on the browser screen. Also using the visual information contained in the multigraph, one is able to define heuristics rules, that are applied to recognize multiple logical areas, which correspond to different parts of the page. These studies and [40] prove that the visual representation is useful and can be used in web page classification.

1.3 Contribution

The first contribution is the construction of algorithms of feature extraction and features selection in C/C++ using the OpenCV library. The extraction of well-known low-level features like the Edge Histogram, Gabor Descriptor and Tamura Features. Two methods of feature selection are constructed to select the features more relevants, one based on the Chi-Squared criterion and another on the Principal Components Analysis.

Another major contribution is a study that is made to learn which methods and features, are more suitable for each classification task. Different classifiers are evaluated, by performing different tests using the different methods to conclude which has better accuracy and is more indicate in each task.

1.4 Thesis Outline

This thesis is organized as follows: chapter 2 presents the entire theoretical basis behind the classification process, clarifies how the different descriptors and the feature selection algorithms are built, as well as an introduction to machine learning is given, supervised learning and the algorithms used to perform classification. Chapter 3 explains the main aspects and properties of the web pages selected for each class in each classification task, and how those web pages are extracted. Chapter 4 displays a measure of the classification performance of each classifier in the different tasks, and discuss the performance based on the confusion matrix. And finally a conclusion and discussion about the results obtained and ideas for future works, are given in chapter 5.

Chapter 2

Classification Process

This chapter presents the work methodology used to fulfill the objectives proposed. Namely how the process of classification of new web pages is done. In Figure 2.1 it is possible to see the steps necessary to do the prediction of classification of new web pages.

First we present an explanation of the methods used to extract feature from the images, and the construction of the respective feature descriptors. We will explained in detail the techniques used to perform feature selection. Finally an introduction to the topic of machine learning, and a discussion about the algorithms used to classify new web pages.

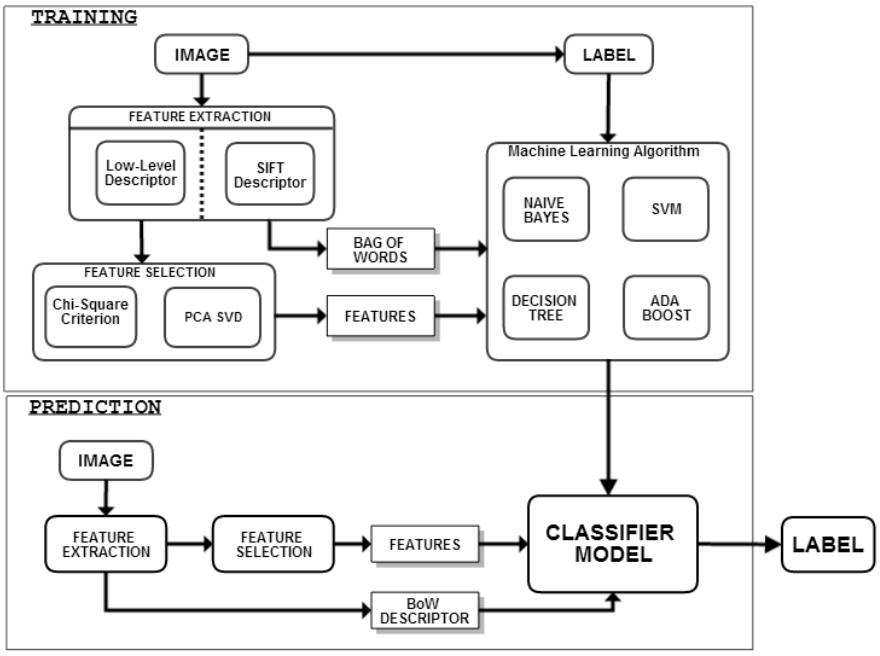


Figure 2.1: Classification Process.

2.1 Feature Extraction

The concept of feature in computer vision and image processing refers to a piece of information which is relevant and distinctive. For each web page different feature descriptors (feature vector) are computed. This section describes how a descriptor of low level features which contains 166 attributes that characterize the page is obtained and how the SIFT descriptor using Bag of Words model is built.

2.1.1 Low Level Descriptor

Visual descriptors are descriptions of visual features of the content of a image. These descriptors describe elementary characteristics such as shape, color, texture, motion, among others. To built this descriptor the following features were extracted from each image: color histogram, edge histogram, tamura features and gabor features. Next an explanation of the features extracted used to construct this descriptor is presented.

2.1.1.1 Color Histogram

The color histogram is a representation of the distribution of colors in an image. It can be built in any color space, but the ones used in this work are the RGB or HSV color space. In the RGB color space all the different colors and shades are derived from varying combinations of red, green and blue. An image pixel tonality for the RGB color is expressed between 0 and 255 assigned to each color channel, where 0 equals to pure black and 255 to pure white. If an image has very bright colors, this will result in a histogram graph with the majority of the pixels located to the right of the center. While an image with dark and shadows areas will produce a histogram with the pixels concentrated in the left part of the graph.

Figure 2.2 shows the histogram for each color channel of the RGB color space. Each bin of the histogram, shown as bars, represents the frequency of the observations in that interval (bin), and the height of the each rectangle is equal to the frequency density of the interval. Since the number of bins used in this work for the color histogram are 32, the interval of each bin is multiple of 8.

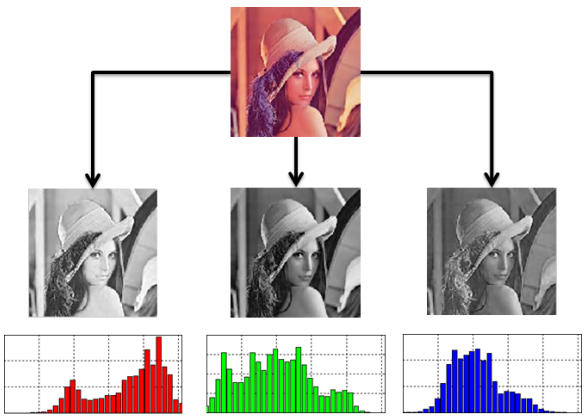


Figure 2.2: Red, green and blue channels of the image, and their respective histograms

The HSV color space capture the components on the way a person perceive color. It was selected because it reflects human vision quite accurately and because it mainly uses only one of its components (Hue) to describe the main properties of color in the image. The other two components (i.e., saturation and value) are significant only when describing black, white, gray and the various shades of colors.

The Hue histogram represented in Figure 2.3 is constructed by discretization of the colors in the image into bins. Each bin will represent an intensity spectrum. This means that a histogram provides a compact summarization of the distribution of data in an image.

Figure 2.3: Hue channel of the image, and the respective histogram

2.1.1.2 Edge Histogram

An edge is a significant local change of intensity, that typically occurs on the boundary between two different regions in an image. In computer vision the goal of edge detection is to produce a line drawing of a scene from an image of that scene. Some important features can be extracted from this edges of an image (e.g., corners, lines, curves), used later by higher-level computer vision algorithms. This changes of intensity can happen by various events. Geometric events, like discontinuity in depth or discontinuity in surface orientation. Non-geometric events, as specularity, shadows or inter-reflections. One way to represent this edge features is using an histogram. An edge histogram will represent the frequency and directionality of the brightness changes in the image.

The Edge Histogram Descriptor in [8] describes the edge distribution in an image. It is a descriptor that expresses only the local edge distribution in the image, describing the distribution of non-directional edges and non-edge cases, as well as four directional edges, and keeps the size of the descriptor as compact as possible for an efficient storage of the metadata.

To extract the EHD, the image is divided into a fixed number of sub-images and the local edge distribution for each sub image is represent by a histogram. The edge extraction scheme is based on a image block rather than on the pixel, i.e., each sub-image space is divided into small square blocks.

As shown in Figure 2.4, the image is divided into 4x4 non-overlapping blocks defined as sub-images, and to characterize a sub-image a histogram of edge distribution is generated. Each histogram represents the distribution of 5 types of edges in each sub-image, each histogram contains 5 bins. The edge types are arranged in the following order: vertical, horizontal, 45-degree diagonal, 135-degree diagonal and non-directional.

For each image block it is determined which edge is predominant, i.e., the image block is classified into one of the 5 types of edge or a non edge block. Since there are 16 sub images in the image, the final histogram is construct by 16x5 = 80 bins.

The non edge blocks do not enter in any histogram bins, although all the histogram bins are normalized by the number of image blocks including the non edge blocks, implying that the summation of all

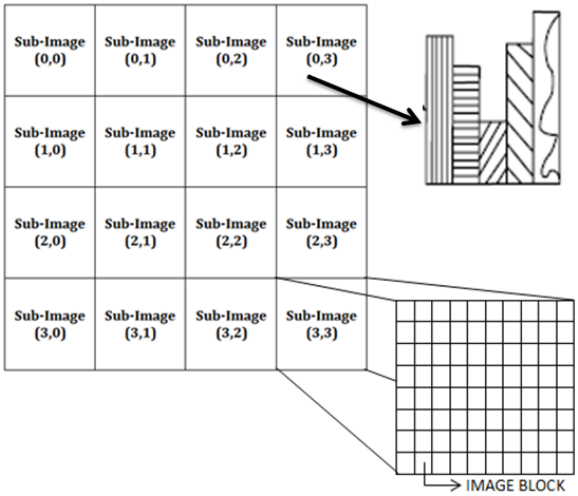


Figure 2.4: Representation of the image space divided into 4x4 non-overlapping blocks, definition of image block and the five types of edge bins for each sub image

the bin values of the histogram is less than or equal to 1.

By applying digital filters in the spatial domain they were able to extract an edge feature in the image block. For make it happen the image block (i, j)th is divided into four sub-blocks. For each sub-block a label is assigned and the average of gray level a_k *where* $k = 0, ..., 3$ *is computed for the four sub-blocks at the (i, j)th image block, as shown in Figure 2.5. Now for each (i, j)th image block the respective vertical, horizontal, 45-degree diagonal, 135-degree diagonal and non-directional edge magnitude are obtained as follows:*

$$m_v(i,j) = |\sum_{k=0}^{3} a_k(i,j) \times f_v(k)| \tag{2.1}$$

$$m_h(i,j) = |\sum_{k=0}^{3} a_k(i,j) \times f_h(k)| \tag{2.2}$$

$$m_{d-45}(i,j) = |\sum_{k=0}^{3} a_k(i,j) \times f_{d-45}(k)| \tag{2.3}$$

$$m_{d-135}(i,j) = |\sum_{k=0}^{3} a_k(i,j) \times f_{d-135}(k)| \tag{2.4}$$

$$m_{nd}(i,j) = |\sum_{k=0}^{3} a_k(i,j) \times f_{nd}(k)| \tag{2.5}$$

where $f_v(k)$, $f_h(k)$, $f_{d-45}(k)$, $f_{d-135}(k)$ *and* $f_{nd}(k)$ *are the filter coefficients for the 5 types of edges, where* $k = 0, ..., 3$ *represents the location of the sub-blocks (Figure 2.6).*

The corresponding edge in the image block is obtained if equations 2.1 to 2.5 are greater than a threshold (T_{edge}*) as in equation 2.6. Then, the histogram values of the corresponding edge bin is increases by one, otherwise the image block contains no edge.*

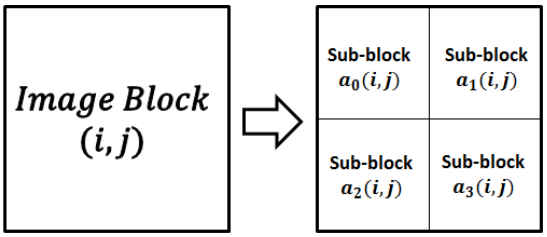


Figure 2.5: Image block divided into sub-blocks, and the respective average of gray levels

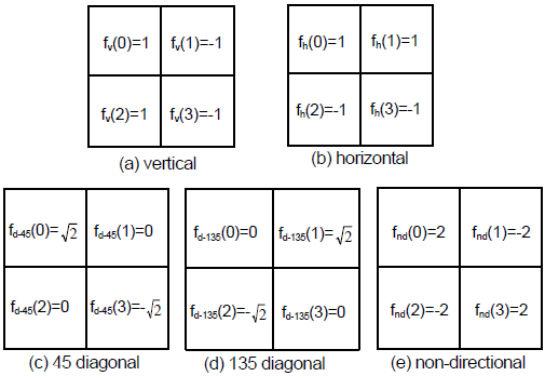


Figure 2.6: Filter coefficients for the 5 types of edges used for edge detection

$$max\{m_v(k), m_h(k), m_{d-45}(k), m_{d-135}(k), m_{nd}(k)\} > T_{edge} \tag{2.6}$$

The size of the image block is assumed to be a multiple of 2. Equations 2.7 and 2.8 show how the size of the image block is obtained for an image.

$$x = \sqrt{\frac{image\_width \times image\_height}{number\_of\_blocks}} \tag{2.7}$$

$$block\_size = |\frac{x}{2}| \times 2 \tag{2.8}$$

where $image\_width$ *and* $image\_height$ *represent the horizontal and vertical sizes of the image, respectively. Each sub-image is divided into a fixed number of image blocks (number_of_blocks), to cope with different sizes (resolutions) of the images.*

2.1.1.3 Tamura Features

Texture is a very general notion that can be attributed to almost everything in nature. Image textures are defined as images of natural textured surfaces and artificially created visual patterns, which approximate, within certain limits, these natural objects.

Tamura et al. [38] on the basis of psychological experiments, propose six features corresponding to human visual perception: coarseness, contrast, directionality, line-likeness, regularity and roughness. After testing the features, the first three attained very successful results and they concluded those were the most significant features corresponding to human visual perception. The definition of these three features in [11] shows the preprocessing that is applied to the images and the steps necessary to extract those three features. The coarseness and contrast are scalar values, and the directionality is histogramized into a histogram of 16 bins. Below we present an overview of how this three features can be extracted from an image.

Coarseness *This feature gives information about the size of the texture elements. Has a direct relationship to scale, and an image contain texture at several scales. The essence of calculating the*

coarseness value is to use operators of various sizes. Aims to identify the largest size at which a texture exists, even where a smaller micro texture exists. The coarseness measure is calculated in the next 4 steps.

1. *First compute the averages at every point* (n_0, n_1) *of the image* I *over neighborhoods (equation 2.9). The size of the neighborhoods are powers of two (1 x 1, 2 x 2, ... , 32 x 32), i.e.,* $k = 0, ..., 5$.

$$A_k(n_0, n_1) = \frac{1}{2^{2k}} \sum_{i=1}^{2^{2k}} \sum_{j=1}^{2^{2k}} I(n_0 - 2^k - 1 + i, n_1 - 2^k - 1 + j) \tag{2.9}$$

2. *At every point compute absolute differences* $E_k(n_0, n_1)$ *between the non-overlapping neighborhoods on opposite sides in the horizontal and vertical directions, equations 2.10 and 2.11.*

$$E_k^h(n_0, n_1) = |A_k(n_0 + 2^{k-1}, n_1) - A_k(n_0 - 2^{k-1}, n_1)| \tag{2.10}$$

$$E_k^v(n_0, n_1) = |A_k(n_0, n_1 + 2^{k-1}) - A_k(n_0, n_1 - 2^{k-1})| \tag{2.11}$$

3. *At every point* (n_0, n_1)*, find the value of* k *that maximizes the difference* $E_k(n_0, n_1)$ *in either direction, equation 2.12*

$$s(n_0, n_1) = \arg\max_{k,d} E_k^d(n_0, n_1) \tag{2.12}$$

4. *Then the coarseness measure (equation 2.13) is the average over* 2^s *for the entire image.*

$$F_{coars} = \frac{1}{N_0 N_1} \sum_{n_0=1}^{N_0} \sum_{n_1=1}^{N_1} 2^{s(n_0, n_1)} \tag{2.13}$$

Contrast *This measure aims to captures the dynamic range of gray levels in the image, together with the polarization of the distribution of black and white. The second-order and normalized fourth-order central moments of the gray level histogram, that is, the variance* σ^2*, and kurtosis*[1] α_4 *are used to define the contrast (equation 2.14).*

$$F_{con} = \frac{\sigma}{(\alpha_4)^n} \text{ where } \alpha_4 = \frac{\mu_4}{\sigma^4} \tag{2.14}$$

where μ_4 *is the fourth moment about the mean. The value* $n = \frac{1}{4}$ *is recommended by Tamura, as the best value for discriminating the textures. Experimentally, it gives the closest agreement to human measurements. This is the value used in the calculation of this measure.*

Directionality *This feature is a global property over a region. Does not aim to differentiate between different orientations or patterns, but measures the total degree of directionality. This is measured using the frequency distribution of oriented local edges against their directional angles. Two simple masks*

[1] Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. Higher values indicate a higher, sharper peak, and lower values indicate a lower, less distinct peak.

(Figure 2.7) are used to convolving the image to calculate the horizontal and vertical derivatives $\triangle_H$ and $\triangle_V$. And for every position (n_0, n_1) the angle (Equation 2.15) is calculated.

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

Figure 2.7: Kernels used for calculate the horizontal and vertical derivatives

$$\theta = \frac{\Pi}{2} \arctan^{-1} \frac{\triangle_V(n_0, n_1)}{\triangle_H(n_0, n_1)} \tag{2.15}$$

These values are then histogramized into a histogram edge probabilities of 16 bins. The histogram will reflect the degree of directionality.

2.1.1.4 Gabor Features

The interest about the Gabor functions is that it acts as low-level oriented edge and texture discriminators, sensitive to different frequencies and scales, which motivated researchers to extensively exploit the properties of the Gabor functions. The Gabor filters have been shown to posses optimal properties in both spatial and frequency domain, and for this reason is well suited for texture segmentation problems.

The Gabor descriptor is computed by passing the image thought a filter bank of Gabor filters. Gabor filters are a group of wavelets, were each wavelet captures energy at a given frequency an direction, and texture features can then be extracted from this group of energy distributions. The scale (frequency) and orientation properties of the Gabor filter are useful when texture analysis is required. Therefore a filter bank consist in a group of Gabor filters with different frequencies and orientations.

Zhang et al. [43] present an image retrieval method based on Gabor filter, where the texture features were found by computing the mean and variation of the Gabor filtered image.

Follows a description about the fundamentals of 2-D Gabor filters, and the texture representation and retrieval based on the output of Gabor filters to create the Gabor descriptor.

GABOR FILTER

Gabor filter is a linear band-pass filter whose impulse response is defined as a Gaussian function modulated with a complex sinusoid called the mother wavelet (equation 2.16).

$$g(x, y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) \exp\left(-\frac{1}{2}\left(\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2} + 2\pi j f x\right)\right) \tag{2.16}$$

where f is called the modulation frequency of the Gabor function, σ_x and σ_y determine its bandwidth and $j = \sqrt{-1}$.

The impulse responses of the filters in a Gabor filter bank are scaled and rotated version of (2.16)*. Information about the texture is obtained through statistics of the output of those filters.*

For a given image $I(x, y)$, $(x, y) \in \Psi$ (Ψ is the set of image point), its discrete Gabor wavelet transform is given by a convolution:

$$W_{mn}(x,y) = \iint_{\Psi} I(x_1,y_1) g^*_{mn}(x-x_1)(y-y_1)\mathrm{d}x_1\mathrm{d}y_1 \tag{2.17}$$

where g^*_{mn} are the complex conjugate Gabor wavelets at scale m and orientation n of equation 2.16.

GABOR DESCRIPTOR

The result of the convolution of the Gabor filter with the image, is an array of magnitudes at each scale $m = 0, 1, ..., M-1$ and orientation $n = 0, 1, ..., N-1$, where M and N are the number of scales and orientation, respectively. That array represents the energy content at different scale and orientation of that image. Assuming that image regions have homogeneous texture, the mean μ_{mn} and standard deviation σ_{mn} of the magnitude are calculated, and used to represent the homogeneous texture feature of the region.

$$\mu_{mn} = \iint_{\Psi} |W_{mn}(x,y)|\mathrm{d}x\mathrm{d}y \tag{2.18}$$

$$\sigma_{mn} = \sqrt{\iint_{\Psi} \left(|W_{mn}(x,y)| - \mu_{mn}\right)^2 \mathrm{d}x\mathrm{d}y} \tag{2.19}$$

The final descriptor (feature vector) is formed as vector of means (equation 2.18) and standard deviations (2.19)*, in each scale and orientation of the filter responses as (equation 2.20).*

$$Gabor_{descriptor} = [\mu_{00}\,\sigma_{00}\ \ \mu_{01}\,\sigma_{01}\ \cdots\ \mu_{(M-1)(N-1)}\,\sigma_{(M-1)(N-1)}] \tag{2.20}$$

2.1.2 SIFT Descriptor using Bag of Words model

In computer vision, keypoints-based image features are getting more attention. Keypoints are salient image patches that contain rich local information of an image. The Scale invariant feature transform was developed in 1999 by David Lowe. The SIFT features are one of the most popular local image feature for general images, and was later refined and widely described in [25].

This approach transforms image data into scale-invariant coordinates relative to local features. The SIFT detector has four main stages of computation used to generate the set of image features:

1. ***Scale-space extrema detection*** *This stage of computation searches over all scales and image locations. It is implemented efficiently by using a different-of-Gaussian function to identify potential interest points that are invariant to scale and orientation. The scale-space is then defined by the funcion:*

   $$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

   where $L(x,y,\sigma)$ is the convolution of the variable scale Gaussian $G(x,y,\sigma)$ with the input image $I(x,y)$.

2. ***Keypoint localization*** *As scale-space extrema detection produces too many keypoints candi-*

dates, in this stage for each candidate keypoint the interpolated location of the extremum is calculated, which substantially improves matching and stability. The interpolation is done using the quadratic Taylor expansion of the Difference-of-Gaussian scale-space function, $D(x, y, \sigma)$ with the candidate keypoint as the origin. This Taylor expansion is given by:

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x$$

where D and its derivatives are evaluated at the candidate keypoint and $x = (x, y, \sigma)$ is the offset from this point. The location of the extremum, is given by

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}$$

If the function value at $\hat{x}$ is below a threshold value then this point is excluded. This eliminates points that have low contrast or are poorly localized on an edge.

3. ***Orientation assignment*** *One or more orientations are assigned to each keypoint location based on local image gradient directions. This step achieve invariance to rotation as the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. For the smoothed image L at scale σ, the gradient magnitude $m(x, y)$, and orientation $\theta(x, y)$, are precomputed using pixel differences as:*

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}$$

$$\theta(x, y) = arctan2(L(x, y + 1) - L(x, y - 1), L(x + 1, y) - L(x - 1, y))$$

An orientation histogram with 36 bins is formed. Each sample in the neighboring window added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint.

4. ***Keypoint descriptor*** *Finally a descriptor vector is computed for each keypoint. Keypoint descriptors typically uses a set of 16 histograms, aligned in a 4x4 grid, each with 8 orientation bins, one for each of the main compass directions and one for each of the mid-points of these directions. This results in a feature vector containing 128 elements.*

The bag-of-words (BoW) model [23] can be defined as follows. Given a training dataset D, that contains n images, where $D = \{d_1, d_2, ..., d_n\}$, where d is the extracted features, a specific algorithm, is used to group D based on a fixed number of visual words W represented by $W = \{w_1, w_2, ..., w_v\}$, where V is the cluster number. Then, it is possible to summarize the data in a $N \times V$ co occurrence table of counts $N_{ij} = n(w_i, d_j)$, where $n(w_i, d_j)$ denotes how often the word w_i occurred in an image d_j.

Figure 2.8 shows how to extract the BoW feature from images. The following steps are required: i) detect the SIFT keypoints, ii) compute the local descriptors over those keypoints, iii) quantize the descrip-

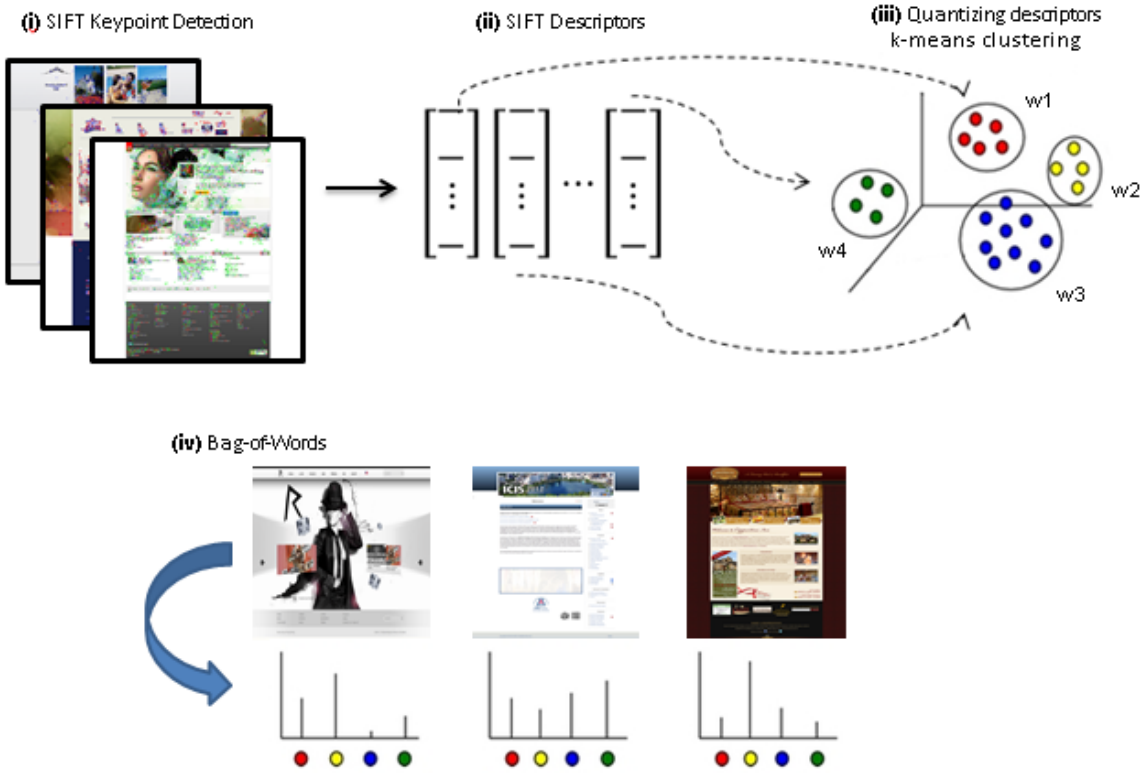


Figure 2.8: Steps for constructing the bag-of-words for image representation.

tors into words to form the visual vocabulary, and iv) to retrieve the BoW feature, find the occurrences in the image of each specific word in the vocabulary.

Using the SIFT image feature detector and descriptor implemented in OpenCV, each image is abstracted by several local keypoints.These vectors are called feature descriptors and as explained above the SIFT converts this keypoints into a 128-dimensional vector. The BoW model is mainly designed for the local descriptors of images which describe regions around the keypoints detected in the images and one image can have a bunch of salient patches around keypoints. Local descriptors (128-dimensional SIFT vector) is demonstrated to be a good way to represent the characteristics of these patches. But once we extract such local descriptors for each image, the total number of them would most likely be of overwhelming size. In that case, BoW solve this problem by quantizing descriptors into "visual words", which decreases the descriptors amount dramatically. This is done by k-means clustering, an iterative algorithm for finding clusters in data. This will allow to find a limited number of feature vectors that represent the feature space, allowing to construct the dictionary.

To use the BoW model to represent images, it is necessary to construct a dictionary. The OpenCV class used to construct the dictionary is BOWKMeansTrainer. The C++ constructor is given by:

```
BOWKMeansTrainer::BOWKMeansTrainer( int clusterCount, const TermCriteria
                &termcrit=TermCriteria(), int attempts, int flags=KMEANS_PP_CENTERS );
```

This class is a wrapper around the K-means algorithm, and the parameters of this class determine the number of cluster centers (dictionary size), the termination criterion for the algorithm and the K-Means version used, that by default is $KMEANS\_PP\_CENTERS$*. By doing this we choose a number of*

cluster centers where a cost function based on the distances of data points in a cluster to the respective cluster is minimized. After the BOWKMeansTrainer object is called, the cluster method to run K-Means is applied on the descriptors and obtain the cluster centers.

Once the dictionary is constructed, it is possible to describe new images by extracting features from them and matching them with features in the dictionary which are closest. That is, the dictionary is ready to be used to encode images. This task is done by the following three interfaces:

- *Feature Detector - identifies the keypoints.*
- *Feature Extractor - extracts features from the keypoints found by the detector.*
- *Descriptor Matcher - matches those features to the features in the dictionary to construct the BoW representation of the image.*

As explained above the SIFT image feature detector and descriptor was the one used. The Descriptor Matcher used was the FlannBased, an interface that performs a quick and efficient matching using the FLANN (Fast Approximate Nearest Neighbor Search Library). The constructors are given by:

```
//Create a nearest neighbor matcher
Ptr<DescriptorMatcher>   matcher   = DescriptorMatcher::create("FlannBased");

//Create Sift descriptor extractor
Ptr<DescriptorExtractor> extractor = DescriptorExtractor::create("SIFT");

//Create Sift feature keypoint extractor
Ptr<FeatureDetector>      detector  = FeatureDetector::create("SIFT");
```

And the The BOWImgDescriptorExtractor class ties it all together:

```
//Create BoW descriptor extractor
BOWImgDescriptorExtractor BoWDescriptor(extractor, matcher);

//Set the dictionary we created in the first step
BoWDescriptor.setVocabulary(dictionary);
```

In the implementation of this algorithm, different sizes of the dictionary (i.e., the number of cluster centers) were used, to see the difference of performance in the classifiers.

2.2 Feature Selection

An important component of both supervised and unsupervised classification problem is feature selection. A technique that selects a subset of the original attributes by selecting a number of relevant features. By choosing a better feature space, a number of problems can be solved, e.g., avoid overfitting and achieve better generalization ability, reduce the storage requirement and training time and allows us to better understand the domain.

Relevant features are those that best characterize each class, features that are critical for performance. The features discarded, are either partially or completely irrelevant/redundant in the data set,

and they can lead to a reduction of the classification accuracy and to an unnecessary increase of computational set. Redundant features are those which provide no more information than the currently selected features, and irrelevant features provide no useful information in any context.

The NxM Matrix in table 2.1 shows how the features extracted from each image(sample) are stored in a matrix.

Table 2.1: Example of a Matrix NxM storing the features extracted from each image

	Feature#1	Feature#2	$\cdots$	Feature#N
Sample#1	a_1	a_1	$\cdots$	a_1
Sample#2	a_2	a_2	$\cdots$	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sample#M	a_M	a_M	$\cdots$	a_M

The next subsections provides a basis for understanding how the two algorithms for applying feature selection in table 2.1 are built. One is based on the Chi-Square Criterion, the other using the Principal Components Analysis. In both methods a different number M corresponding to the most relevant features is selected. The different values of M used in this work are 1%, 2%, 5%, 10%, 20% and 50% of the total features.

2.2.1 Chi-Square Criterion

Feature Selection via chi square (X^2) *test is a very commonly used method [22]. Chi-squared attribute evaluation evaluates the worth of a feature by computing the value of the chi-squared statistic with respect to the class. This test is applied for determining the correlation between the decision classes and the features. The initial hypothesis* H_0 *is the assumption that the two features are unrelated, and it is tested by chi-squared formula:*

$$X^2 = \sum_{i=1}^{rows} \sum_{j=1}^{cols} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

where O_{ij} *is the observed frequency and* E_{ij} *is the expected (theoretical) frequency, asserted by the null hypothesis. The greater the value of* X^2*, the greater the evidence against the hypothesis* H_0 *is. Then, the original data is modified to include the features that are statistically more relevant, that is, only the top M attributes are then selected. The Feature Selection method using the Chi-Squared criterion is represented in algorithm 1.*

2.2.2 Principal Component Analysis using Singular Value Decomposition

PCA was invented in 1901 by Karl Pearson as an analogue of the principal axes theorem in mechanics. This algorithm is based on [37], that proposed a method using PCA to perform feature selection. PCA is a powerful data representation method, where most of the applications of PCA is to transform samples into a new space and to use lower-dimensional representation from the new space to denote

Algorithm 1 Feature Selection using Chi-Square Criterion

Input: Data Matrix (M×N) ▷ M represents the number of samples, and N the number of features
Input: Number of classes C.
Output: Top M features

1: For each feature and class
Find the mean value corresponding to each feature.

2: Then for each feature
Compute the mean value of the classes mean values.

3: For each feature
Compute the Expected and Observed Frequencies for all features
- Expected Frequency is equal to the size of samples in each class.
- Observed Frequency is the number of frequencies obtained for each sample of each class.

4: For each feature
Compute the chi-squared value for each feature.

$Chi2 = \frac{(ExpectedFreq - ObservedFreq)^2}{ExpectedFreq}$;

5: Sort the chi-squared values and choose the M features with the smallest sum of all values.

the sample. They achieved feature selection by using the PCA transform from a viewpoint of numerical analysis, allowing to select a number M of features components from all the features components of the original samples.

We can use the Singular Value Decomposition (SVD) to perform PCA. The SVD technique allows to reduce dimensionality by obtaining a more compact representation of the most significant elements of the data set, and this enable to express the data set more compactly.

If X is the data matrix, the PCA viewpoint requires to compute the eigenvalues and eigenvectors of the covariance matrix, which is the product XX^T. Since the covariance matrix is symmetric, it can be diagonalized by a matrix of its normalized eigenvectors, resulting in equation 2.21, where D is a diagonal matrix and E is a matrix of eigenvectors of X.

$$XX^T = EDE^T \tag{2.21}$$

On the other hand, when applying SVD decomposition to the data matrix X (equation 2.22) it results in an orthogonal matrix U, a diagonal matrix S and another orthogonal matrix V. If we try to construct the covariance matrix from the SVD decomposition, it will result in equation 2.23.

$$X = USV^T \tag{2.22}$$

$$XX^T = (USV^T)(USV^T)^T = (USV^T)(VSU^T) \tag{2.23}$$

Since V is an orthogonal matrix ($V^TV = I$), resulting in $XX^T = US^2U^T$. Comparing with equation 2.21, it is concluded that the square roots of the eigenvalues of XX^T are the singular values of X, and

that the columns of matrix U contain the eigenvectors of XX^T. The tutorial [35] also demonstrates how PCA and SVD are intimately related, and how to interpret the result of the SVD decomposition with PCA. The Feature Selection using PCA through SVD is represented in algorithm 2.

Algorithm 2 Feature Selection using PCA through SVD

Input: Data Matrix (M×N) ▷ M represents the number of samples, and N the number of features
Output: Top M features

1: Perform mean normalization in the Data Matrix

2: Calculate the SVD decomposition of the Data Matrix
Resulting in USV^T.

3: Select the eigenvectors that correspond to the first n largest singular values
And denote these vectors as $K_1, ..., K_n$, respectively.

4: Calculate the contribution, of each feature component as follows $c_j = \sum_{p-1}^{m} |K_{pj}|$
where K_{pj} denotes the j entry of K_p, $j = 1, 2, ..., N$, $p = 1, 2, ..., m$. $|K_{pj}|$ stands for the absolute value of K_{pj}

5: Sort c_j in the descending order.
And select the M features corresponding to the M largest order in c_j.

2.3 Machine Learning

Machine Learning is a huge field in computer science. Most people uses daily some machine learning algorithm, and often without knowing it. For example, the websites like Amazon or Netflix that recommend books to the client to buy or movies to rent, these are examples of learning algorithms that have learnt what sort of products would the client like to buy or rent, and give these customized recommendations to them.
In 1959, Arthur Samuel [32] defined Machine Learning like:

> *"Field of study that gives computers the ability to learn without being explicitly programmed"*

During that time he wrote a checkers program, which would play games of checkers against itself. After a few games, Arthur Samuel noticed that he had a checkers program that actually learn to play checkers by learning what are the boards positions that tend to be associated with wins and the ones associates with losses.

A more recent and formal definition of ML was give by Tom Mitchell [28], who says that a well-posed learning problem is defined as follows:

> *"A computer program is set to learn from an experience E with respect to some task T and some performance measure P if its performance on T as measured by P improves with experience E"*

In the case of the program of checkers, the experience E is the experience of playing lots of games of checkers against itself. The task T is the task of playing checkers, and the performance measure P

will be the fraction of games it wins against a human opponent. With this, we can conclude that Arthur Samuels checkers program has learned to play checkers.

There are many ML problems, but this work is focused on classification problems, which is the goal to categorize objects into a fixed set of categories (e.g., optical character recognition, face detection, spam filtering, topic spotting, medical diagnosis, weather prediction, webpage classification and much more).

The next sections approach the algorithm type of ML used in this work and some of the algorithms built to do classification.

2.3.1 Supervised Learning

Since our goal is to do classification, ML algorithms analyze our collected features and adjust weights, thresholds, and other parameters to maximize the performance according to those goals. The term learning comes from this process of parameter adjustment to meet our goal.

To achieve our goal, i.e. classification, we have a training data set, and a testing data set to pass to the classifier (Figure 2.9). First we use our training set to train our classifier, to then be able to give a prediction over the samples in the testing set. Basically supervised ML, takes a set of training examples with known responses (labels) to the examples, and seeks to build a predictor model that will generates reasonable predictions for the responses (labels) to the Testing examples.

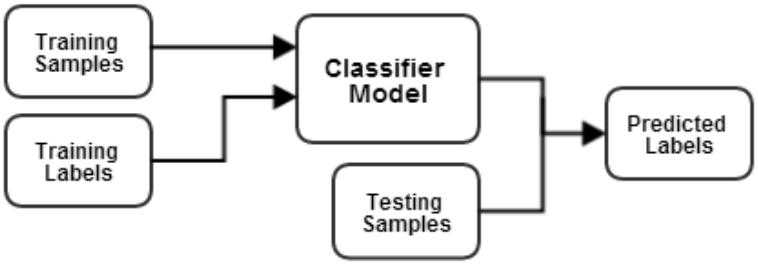


Figure 2.9: Steps in a Supervised Learning Machine. Training and Prediction.

2.3.1.1 Training and Testing Dataset

When the most relevant features are chosen, they are saved in a csv (comma-separated values) file. In the C/C++ Machine Learning API of OpenCV, the training and testing data is given as a cv::Mat matrix. The constructor of cv::Mat is defined as:

```
Mat::Mat(int rows, int cols, int type)
```

Where

- *rows - number of samples of all classes.*
- *cols - number of features for each sample.*
- *type - array of features type.*

So the ordering of the training and testing matrix is row sampling. The class labels corresponding to the training and testing samples is also given in a cv::Mat matrix with equal length, but only one column. So the training of the classifiers is done by passing the matrix of training data and the vector with the corresponding labels.

2.3.1.2 Classifier Evaluation

The confusion matrix (table 2.2) allows the visualization of the performance of our supervised learning algorithm. The columns of the matrix represent the instances in a predicted class, while each row represents the instances in an actual class.

Table 2.2: Confusion Matrix for two classes only

		Prediction outcome		
		p	**n**	**total**
Actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

From the confusion matrix many performance measures can be calculated (all vary between 0 and 1). The True positive rate (TPR) is also know as sensitivity or recall, it is given by equation 2.24. The True Negative Rate (TNR) is calculated as equation 2.25, where is also know as specificity.

$$Sensitivity = \frac{\textit{Number of True Positives}}{\textit{Number of Positives}} \tag{2.24}$$

$$Specificity = \frac{\textit{Number of True Negatives}}{\textit{Number of Negatives}} \tag{2.25}$$

2.3.2 Machine Learning Algorithms

OpenCV supports some of the most useful currently available statistical approaches to machine learning. OpenCV tends to support discriminative and generative algorithms, respectively, one give the probability of the label given the data, the other give the distribution of the data given the label. The data sets stored in .csv file are define to data storage matrices, to pass to the classifiers. The following algorithms are the ones implemented to perform classification of web pages.

2.3.2.1 Naïve Bayes

A supervised classifier, that is called "naïve", because assumes that the features are Gaussian distributed and statistically independent from each other, which sometimes is not true. However, it is an effective classifier that can handle multiple classes. This classifier uses the Bayes Theorem, that says:

$$p(c_j|d) = \frac{p(d|c_j)p(c_j)}{p(d)}$$

- $p(c_j|d)$ = *probability of instance* d *being in class* c_j.
- $p(d|c_j)$ = *probability of generating instance* d *given class* c_j.
- $p(c_j)$ = *probability of occurrence of class* c_j.
- $p(d)$ = *probability of instance* d *occurring.*

Since we have many features, to simplify the task, Naïve Bayesian classifiers as explain assume that attributes have independent distributions and thereby estimate

$$p(d|c_j) = p(d_1|c_j) \times p(d_2|c_j) \times \cdots \times p(d_n|c_j)$$

Naïve Bayes classifiers combine this model with a decision rule. One rule is to pick the hypothesis that is most probable. This classifier predicts a new sample $D = \{d_1, d_2, \ldots, d_n\}$ *that belongs to class* C_j *as:*

$$classify(D) = \arg\max_j p(C = j) \prod_{i=1}^{n} p(D = d_i|C = c_j)$$

This classifier is implemented using the CvNormalBayesClassifier$\{\}$ *class in OpenCV ML library.*

2.3.2.2 Support Vector Machine

This N-class classifier algorithm works by projecting the data into a higher-dimensional space and finding the optimal linear separator between the classes. So given a labeled training data the SVM will output an optimal hyperplane (Figure 2.10) that gives the largest minimum distance (margin) to the training samples. Hence it is possible to use linear classification techniques that in some sense optimally separate classes in the data to categorize new data.

A hyperplane is formally defined by:

$$f(x) = \beta_0 + \beta^T x$$

where β *is known as the weight vector and* β_0 *as the bias. The optimal hyperplane can be represented in an infinite number of different ways by scaling of* β *and* β_0*. Where the one chosen is:*

$$|\beta_0 + \beta^T x| = 1$$

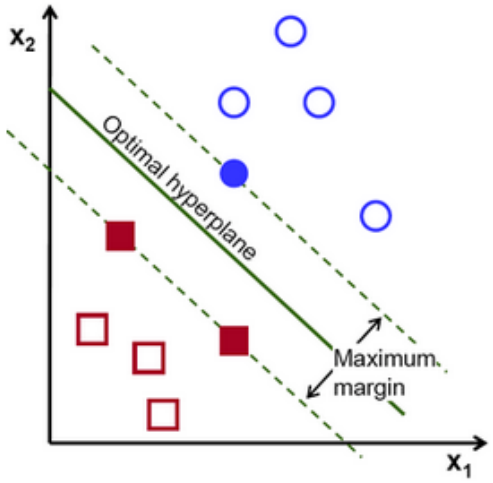


Figure 2.10: Example of a optimal hyperplane with the largest minimum distance between two classes. Illustration withdrawal from the OpenCV documentation.

where x symbolizes the training examples closest to the hyperplane. The training examples closest to the hyperplane are called suppot vectors (also known as canonical hyperplane). The distance between a point x and a hyperplane (β, β_0) is given by:

$$distance = \frac{|\beta_0 + \beta^T x|}{|\beta|}$$

For the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is:

$$distance_{supportvectors} = \frac{1}{|\beta|}$$

the margin M, is twice the distance to the closest examples $M = \frac{2}{|\beta|}$.

Finally, the problem of maximizing M is equivalent to the problem of minimizing a function (equation 2.26), subject to some constraints. These constraints model the requirement for the hyperplane to classify correctly all the training examples x_i.

$$min_{\beta,\beta_0} L(\beta) = \frac{1}{2}|\beta|^2 \quad \textit{subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \ \forall i \tag{2.26}$$

where y_i represents each of the labels of the training examples. The weight vector β and the bias β_0 of the optimal hyperplane are solved using the Lagrange multipliers[2] (Lagrangian optimization).

This technique is implemented in the CvSVM{} class in OpenCV's ML library, and the parameters for the SVM have to be defined in the structure CvSVMParams. Where the parameters chosen are:

- *svm_type= CvSVM::C_SVM - n-class classification ($n \geq 2$), allows imperfect separation of classes with penalty multiplier C for outliers.*
- *kernel_type = CvSVM::LINEAR - no mapping is done, linear discrimination is done in the original feature space. Fastest option.*

[2]Is a strategy for finding the local maxima and minima of a function subject to equality constraints.

2.3.2.3 Decision Tree

A decision tree is a binary tree (tree where each non-leaf node has two child nodes). In classification, each tree leaf is marked with a class label. The tree finds one data feature and a threshold at the current node that best divides the data into separate classes. The data is splited and the procedure is recursively repeated down the left and right branches of the tree. The tree is built recursively, starting from the root node. All training data (feature vectors and responses) is used to split the root node. From each node the optimum decision rule is found based on some criteria. The essence of this algorithm is to define an impurity metric relative to the data in every node of the tree. We want to minimize the sum of differences ("impurity") in each node of the tree. For categorical labels, it is defined a measure that is minimal when most values in a node are of the same class. In ML gini impurity *criteria is used for classification. These method is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. It is computed by summing the probability of each item being chosen multiplied by the probability of a mistake in categorizing that item. If all cases in the node fall into a single target category reaches its minimum (zero). The impurity of each node is given by the following equation:*

$$i(N) = \sum_{j \neq i} P(\omega_i) P(\omega_j)$$

where $P(\Omega_j)$ denote the fraction of patterns at node N that are in class Ω_j.

Once this metric is finished, a binary decision tree searches through the feature vector to find which feature combined with which threshold most purifies the data. By convention, features above the threshold are "true" and that data thus classified will branch to the left, the other data points branch right.

Decision trees are widely used in classification, due to their simplicity of implementation, ease of interpretation of results, flexibility with different data types (categorical, numerical, and more), the ability to handle missing data through surrogate splits, and way of assigning importance to the data features by order of splitting. Also form the basis for the boosting algorithm used in this work, explained in the next subsection.

In OpenCv the class CvDTree$\{\}$ implement the classifier, and for training the tree it is necessary to fill out the tree parameter structure CvDTreeParams.

2.3.2.4 AdaBoost

Boosting algorithms are used to train weak classifiers h_t, with $t \in \{1, \ldots, T\}$. In most cases these classifiers are decision trees with one split (called decision stumps) or at most a few levels of splits. For each call of a weak classifier, a distribution of weights D_t is updated that indicates the importance of samples in the data set for the classification.

The key feature of boosting is that, as the algorithm progresses the weak classifiers trained later will focus on the data points that the earlier trained weak classifiers tended to do poorly on. Briefly it is a process of combining the performance of multiple classifiers strategically to produce a powerful "committee". A score is assigned to each classifier, and the final classifier is defined as the linear

combination of the classifiers from each stage. The algorithm is as follows:

1. $D_1(i) = \frac{1}{m}$, $i = 1, \ldots, m$.
2. *For* $t = 1, \ldots, T$*:*
   (a) *Find the classifier* h_t *that minimizes the* $D_t(i)$ *weighted error:*
   (b) $h_t = \arg\min_{h_j \in H} \epsilon_j$, *where* $\epsilon_j = \sum_{i=1}^{m} D_t(i)$ *(for* $y_i \neq h_j(x_i)$*) as long as* $\epsilon_j < 0.5$*;*
   (c) *Set the* h_t *voting weight* $\alpha_t = \frac{1}{2} log[(1 - \epsilon_t)/\epsilon_t]$, *where* ϵ_t *is the arg min error from step 2b.*
   (d) *Update the data point weights:* $D_{t+1}(i) = [D_t(i) e^{(-\alpha_t y_i h_t(x_i))}]/Z_t$, *where* Z_t *normalizes the equation over all data points* i.

In step 2b) if a classifier with less than a 50% error rate is not found then quit. Meaning that probably better features are required. When this training algorithm is finished, the final strong classifier takes a new input vector x *and classifies it using a weighted sum over the learned weak classifiers* h_t*:*

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

The boost tree classifier in OpenCV can only be trained for 2-class problems, for multi-class it is necessary to transform the training database by "unrolling" *each training sample as many times as the number of classes. In* "unrolling" *we add an additional attribute to each classification, e.g., if the database has N classes, by doing unrolling, N new samples are added for every original sample, one for each possible class, but only one with the correct class as an additional attribute value has a new binary class of 1, all the rest of the new samples have a new binary class of 0. This adapt any binary classifier for N-class classification problems, but makes both training and precision significantly more expensive.*

This classifier is implemented in the CvBoost{} class in OpenCV's ML library, and the parameters are defined in the structure CvBoostParams. Where one of the parameters is the boost type, and the type selected was CvBoost::REAL, which correspond to the real AdaBoost that is a technique that utilizes confidence-rated predictions and works well with categorical data.

Chapter 3

Web Pages Database

In this work, different web page classification experiments are evaluated. There are two binary classifications and one multi-category classification. The two binary classifications are: the aesthetic value of a web page, i.e., if a web page is beautiful or ugly (a measure that depends on the notion of aesthetic of each person), and the recency of a web page, i.e., trying to distinguish between old fashioned and new fashioned web pages. The multi category classification involves classification on the web page topic.

Using the Fireshot plugin[1] *for the Firefox web browser, allows to retrieve a screen shot of a web page and save it as a .PNG file. Different Training sets of 30, 60 and 90 pages are buit for each class of the classification experiment.*

3.1 Aesthetic Value

The notion of aesthetic differs from person to person, because what can be beautiful for someone, can be ugly for another. That's why this classification depends of each classifier and it's an objective classification.

In this classification experiment two classes are defined: Ugly and beautiful web pages. In Aesthetic, the important aspect is the visual design ("Look and Feel") of a web page, and not the quality of information or popularity of the page.

- *The Ugly pages were downloaded from:*

    1. *The article "The World's Ugliest Websites" a design weblog [2], and corresponding comment section.*

[1] https://addons.mozilla.org/pt-pt/firefox/addon/fireshot/

2. *The article 10 worst websites for 2013 [36] and corresponding comment section.*
3. *And the website Worst Websites of the Year 2012 - 2005 [14]*

- *The beautiful pages were retrieved, consulting a design web log, listing the author's selection of the most beautiful web pages of 2008, 2009, 2010, 2011 and 2012 [10].*

An ugly web page, don't transmit a clear message, uses too much powerful colors, lacks clarity and a consistent navigation. While a Beautiful web page has an engaging picture, an easy navigation, the colors compliment each other and it's easy to find what the information needed.

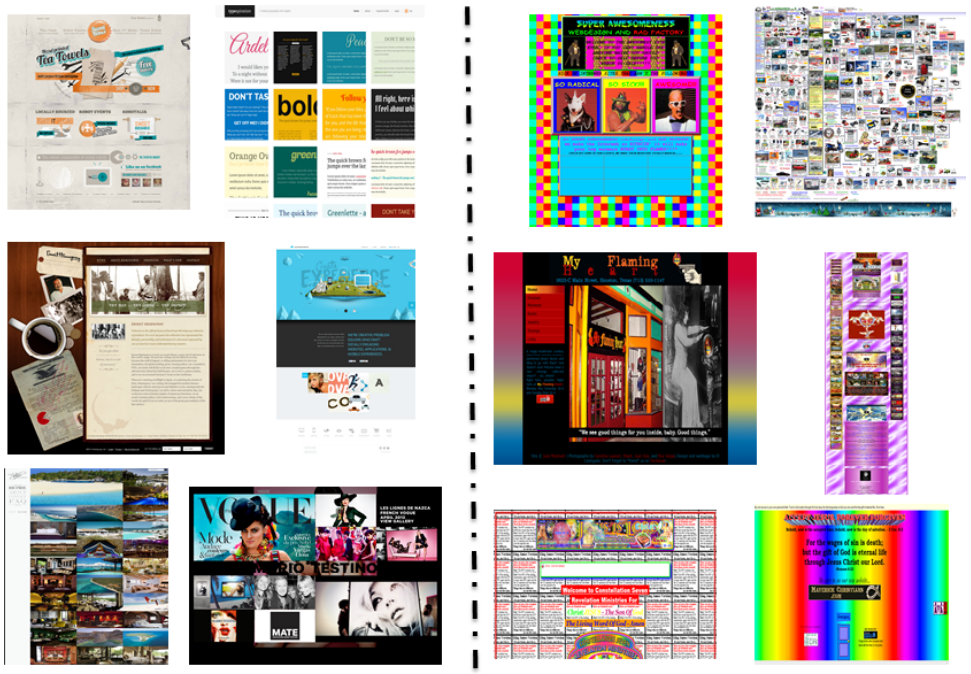


Figure 3.1: An example of the web pages retrieved for the Aesthetic classification. In the left, there are 6 beautiful web pages, and in the right, 6 ugly web pages.

3.2 Recency

The objective of this classification is to be able to distinguish from old fashioned and new fashioned pages. The principal differences between these pages is that nowadays the web design of a page has firmly established itself as an irreplaceable component of every good marketing strategy. The pages have larger background image, blended typography, colorful an flat graphics, that is, every design element brings relevant content to the user. In the past the use of GIFs, very large comprised text and blinding background were common in most sites.

The old web pages were retrieved consulting the article [41], that shows the most popular pages in 1999, and using the Internet Archive web site[2] *allowed to retrieve the versions of those websites in that*

[2]http://archive.org/web/web.php

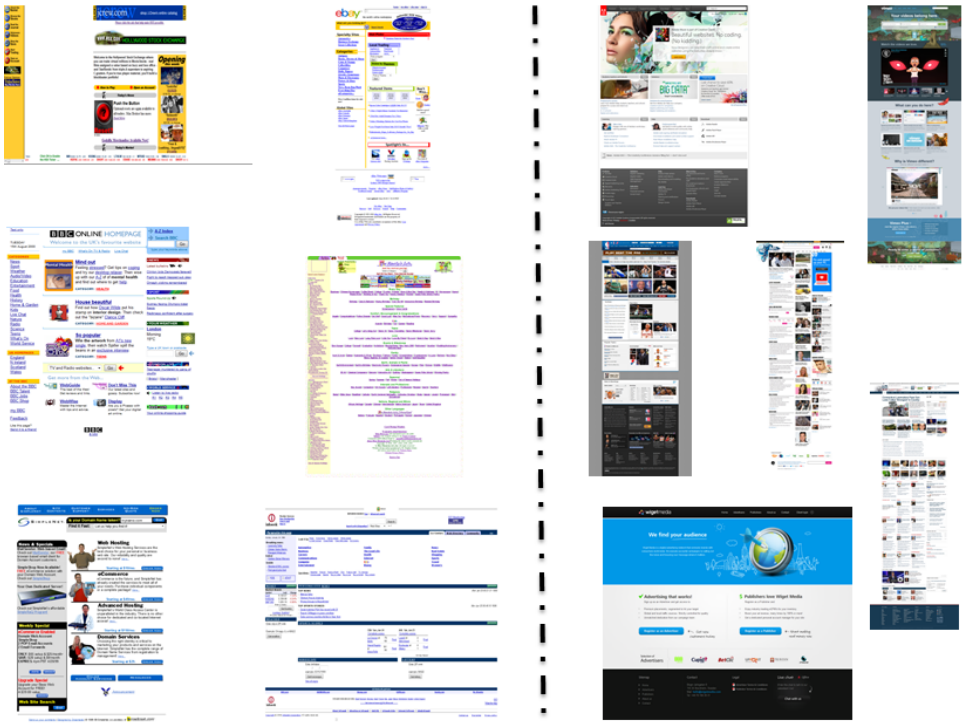


Figure 3.2: An example of the web pages retrieved for the Recency classification. In the left, there are 6 old fashioned web pages from 1999, and in the right 6 new fashioned web pages from 2012.

year. To retrieve the new pages, the Alexa [1] web page popularity rankings was used, selecting then the 2012 most popular pages.

3.3 Web page Topic

In this classification eight classes are defined. These classes are newspaper sites, hotel sites, celebrity sites, conference sites, classified advertisements sites, social networks sites, gaming sites and video-sharing sites.

For the newspaper and celebrity classes, the Alexa [1] popularity rankings was consulted, retrieving the most well-known and popular newspapers and celebrity sites. The celebrity sites also include popular fan sites. The conferences class consist in the homepages of the highest ranked Computer Science Conferences. And for the hotel class, different sites from bed-and-breakfast businesses [4] are retrieved. The classes include different pages from different countries.

The classified advertisements sites were extracted using also the Alexa popularity ranking, retrieving the most visited sites of classifieds of all world (sections devoted to jobs, housing, personals, for sale, items wanted, services, community, gigs and discussion forums). The video-sharing class and the gaming class (company gaming websites and popular gaming online websites), were extracted consulting the google search engine for the most popular sites in this type of websites. Social networks class consist in the major social networking websites homepages (e.g., websites that allow people share interest,

activities, backgrounds or real-life connections).

A topic of a web site is a relevant area in the classification of web pages. Each topic has a relevant visual characteristic that distinguishes them, being possible to classify the web pages despite of their language or country. Looking at the pages retrieved (Figure 3.3 and Figure 3.4), it is possible to see a distinct visual characteristic in each class. The newspapers sites have a lot of text followed with images, while celebrity sites have more distinct colors and embedded videos. The conferences sites usually consist in a banner in the top of the page, and text information about the conference. Hotel sites have a more distinct background, with more photographs. Classifieds sites consist almost in blue hyperlinks with images or text, wth a soft color background and banner. The body content of a video-sharing site consist in video thumbnails. The gaming sites have a distinct banner (an image or huge letters), with a color background and embedded videos. The social networks homepages, have a color pattern that is persistent.

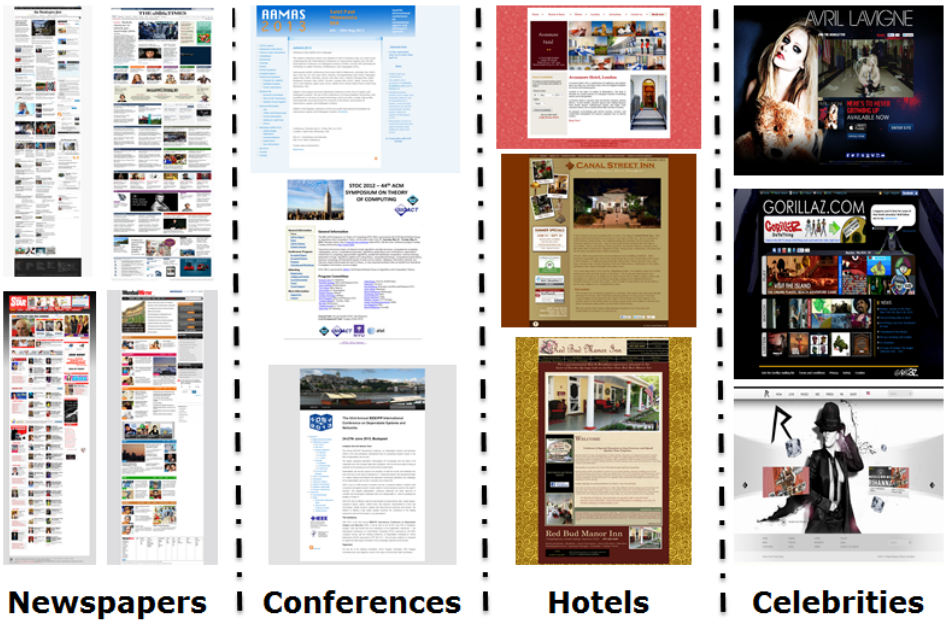


Figure 3.3: Examples of web pages extracted for four web site topic classes.

Figure 3.4: Examples of web pages extracted for the other four web site topic classes.

Chapter 4

Results

By training our classifiers with different training data sets, different comparisons can be made. Different evaluations were made to analyze what features and which classifiers are better for each classification task. Each classifier was evaluated with the low feature descriptor (containing 166 features), just the Color Histogram, Edge Histogram, Tamura Features, Gabor Features, and the descriptor containing the most relevant features selected by the methods of feature selection. Additionally the same datasets were used to train the classifiers with the SIFT descriptor using the bag of words model. The results for each classification task are shown in the next sections, as well as a comparison with the results of [40].

4.1 Aesthetic Value Results

Boer et al. [40] in this experiment with the 166 features had an accuracy using the Naive Bayes and a J48 Decision Tree of 68% and 80% respectively. Using just the Simple Color Histogram and Edge Histogram they correctly classified 68% and 70% respectively for the Naive Bayes, and 66% and 53% for the J48 Decision Tree classifier.

For this experiment, Figure 4.1 show the best rate prediction for our classifiers. When trained the model using just the HSV Histogram attributes, the results show an accuracy of 65% for Naive Bayes, 85% in SVM, 70% for the Decision Tree and 85% using the AdaBoost when trained with 90 images for each class. We also obtained a accuracy rate (70%) for the Naive Bayes and SVM by just using the Gabor descriptor.

When selected the top attributes to train the classifiers, the best results using the Chi-Squared method was when the classifiers was trained with the top 50% attributes. The Naive Bayes and SVM had an accuracy of 65%, the Decision Tree 80% and the AdaBoost an accuracy of 75%. The classifiers when trained with the top 20% attributes selected using the PCA method, the Naive Bayes had an accuracy of 75%, SVM predict correct 65% of the pages, the Decision Tree and the AdaBoost both had an accuracy of 80%.

The standard features selected when selecting the top 1%, 5%, 10%, 20% and 50% attributes of the low-level descriptor using the Chi Square method and the PCA method are:

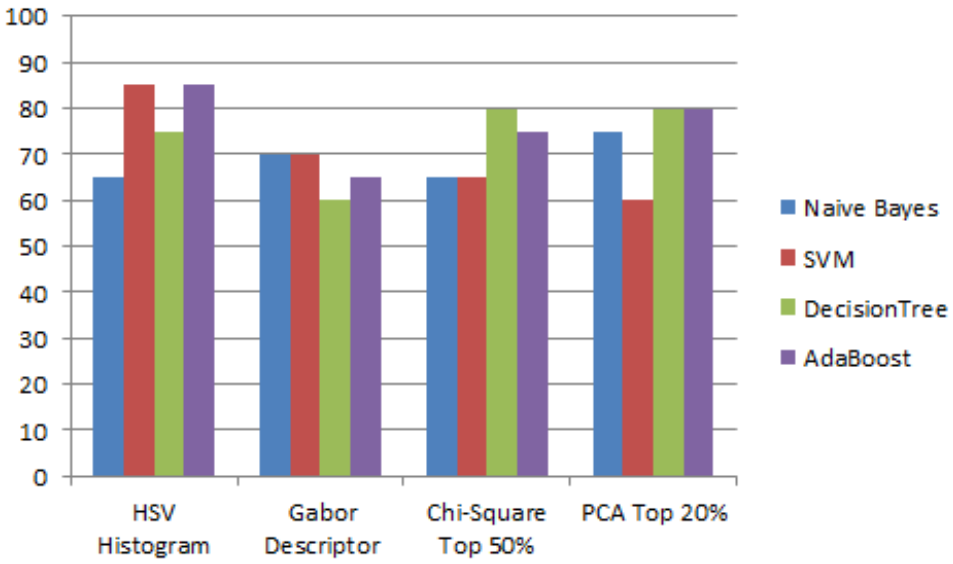


Figure 4.1: Best prediction results for the Aesthetic Value for four different classifiers, using the low-level descriptor. All these predictions values, were obtained by training the classifiers using 90 images for each class.

- *The HSV Histogram bins between 19 and 32 were selected by both methods, but the Chi-Squared also selected the lower bins between 2 and 9. High values in the higher bins, indicate that a web page has bigger probability of being 'beautiful', because correspond to images with lighter colors.*
- *Bins between the 37 and 65 in the Edge Histogram were selected by both methods - indicates difference in the edges in the header of the web pages.*

In table 4.1 using the best predictions accuracy tests, it is possible to verify the average proportions of web pages of each class (beautiful and ugly) that are best predicted in each classifier. These tests were performed using different data size for the training of the classifiers and can be found in appendix A.1.

The Naïve Bayes and SVM classifiers identify better the ugly web pages. The Decision Tree classifier obtain a better rate in identifying the beautiful webpages, while the AdaBoost have a similar rate for each class, but is slightly better in predicting beautiful webpages.

Table 4.1: Proportions of positive (beautiful webpages) and negative (ugly webpages) that were correctly classified, for the four classifiers. The average is calculated using the best three prediction results, using different features for each classifier.

	Naïve Bayes		**SVM**		**Decision Tree**		**AdaBoost**	
	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
HSV	72,7	77,8	81,2	88,9	85,7	69,2	88,9	81,2
Chi2	70	70	70	70	87,5	75	72,72	77,78
PCA	66,7	100	63,6	63,7	87,5	75	80	80
Average:	69,8	82,6	71,6	74,2	86,9	73	80,5	79,6

The results shown in Figure 4.2, are relative to the SIFT descriptor. Using different sizes for the

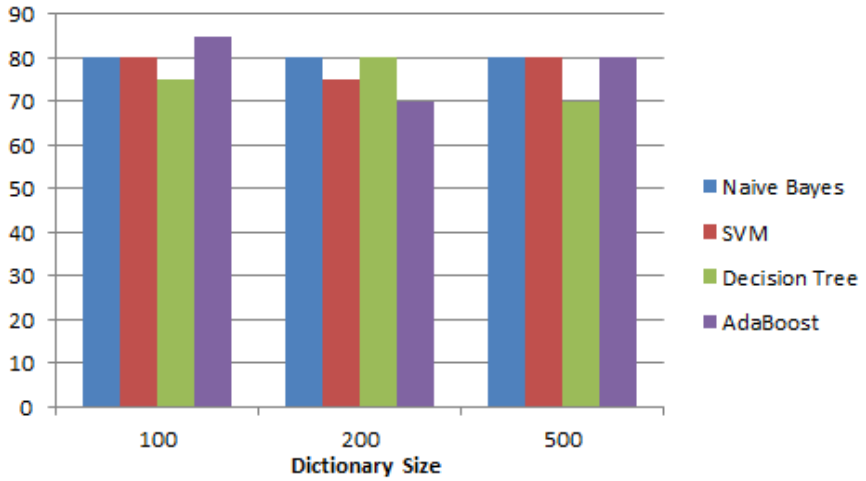


Figure 4.2: SIFT Descriptor using BoW Model prediction results with different dictionary sizes (100, 200 and 500) for the Aesthetic Value.

dictionary, we obtained good result for each classifier. The best results for the Naïve Bayes, SVM and the Decision Tree was of 80%, and for the AdaBoost a predict accuracy of 85%.

All the classifiers show a high prediction accuracy, with different features. Since most of the features choose by the feature selection method are from the Color Histogram, it is possible to achieve a good prediction rate just by passing this simple descriptor. The SIFT descriptor results also show that the images from this two classes have distinctive keypoints, since it was possible to obtain good predictions accuracy for each classifier.

4.1.1 Recency Results

In this experiment, Boer et al. [40] using the complete feature vector had an accuracy using the Naïve Bayes and a J48 Decision Tree of 82% and 85% respectively. Using just the Simple Color Histogram the Naïve Bayes performed slightly less than the baseline and the J48 Decision Tree classifier sightly better. Using only the edge information, the both models correctly classified 72% and 78% respectively for the Naïve Bayes and J48 Decision Tree classifier.

This experiment best results using the low-level descriptor are shown in Figure 4.3. The Naïve Bayes, SVM and Adaboost had an accuracy of 100%, when the top 5% attributes were selected using the chi-square method for the first one and the Gabor descriptor for the other two. The Decision Tree best accuracy (95%), was when the PCA method select the top 5% attributes. These tests were performed using different data size for the training of the classifiers and can be found in appendix A.2.

The standard features selected when selecting the top 1%, 5%, 10%, 20% and 50% attributes of the low-level descriptor using the Chi Square method and the PCA method are:

- *Most of the bins from the Tamura Features are selected for both methods . Higher values in the directionality features is common when a page is new fashioned.*
- *Both methods also selected features from the HSV bins between 15 - 29. A higher values in these bins, indicate a higher probability of the page being new fashioned.*

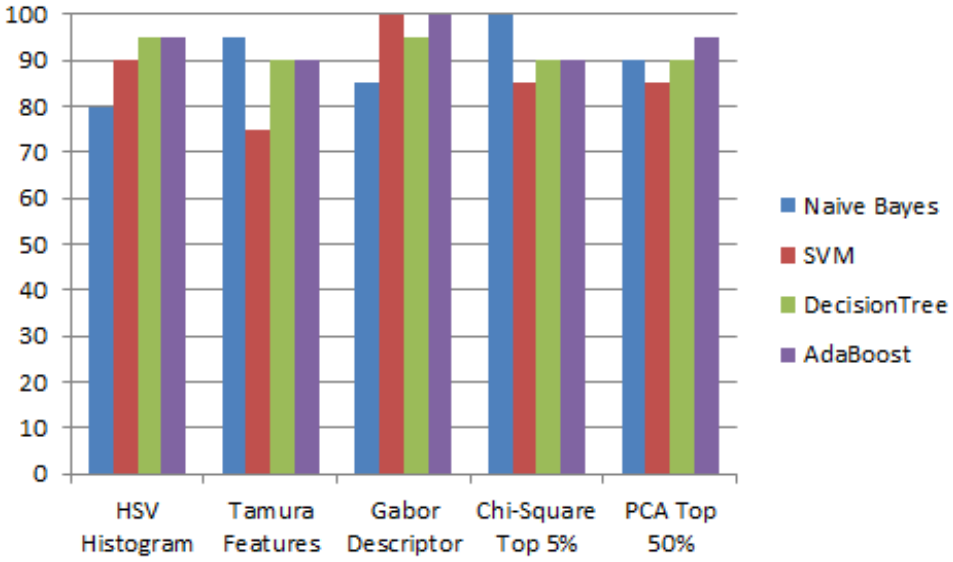


Figure 4.3: Best prediction results for the Recency value for four different classifiers, using the low-level descriptor. All these predictions values, were obtained by training the classifiers using 90 images for each class.

In table 4.2 using the best predictions accuracy tests, it is possible to verify the average proportions of web pages of each class (new and old fashioned web pages) that are best predicted in each classifier. The Naïve Bayes, SVM and Decision Tree classifiers identify better the old fashioned web pages. The AdaBoost have a similar rate for each class, but is slightly better in predicting new fashioned web pages.

Table 4.2: Proportions of positive (new fashioned webpages) and negative (old fashioned webpages) that were correctly classified, for the four classifiers. The average is calculated using the best three prediction results, using different features for each classifier.

	Naïve Bayes		**SVM**		**Decision Tree**		**AdaBoost**	
	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
HSV	80	80	90	90	90,9	100	100	90,9
Chi2	100	100	88,9	81,8	90	90	83,3	100
PCA	83,3	100	76,9	100	90	90	100	90,9
Average:	87,8	93,3	85,3	90,6	90,3	93,3	94,4	93,9

The results shown in Figure 4.4, are relative to the SIFT descriptor. It is possible to verify that all the classifiers obtain a good accuracy. Noteworthy that all the classifiers obtain an accuracy of 90% when they used a dictionary size of 500. The best accuracy result achieved was for the Naïve Bayes with a 95% rate of success, with a dictionary size of 200 words.

This classification indicates that can be learnt just by using simple visual features. All the classifiers obtained good accuracy around 85%, using just the top 1% attributes selected by both methods (appendix A.2). Instead of using a more complex method like BoW, the use of simple visual features allows to decrease the computational cost for larger databases.

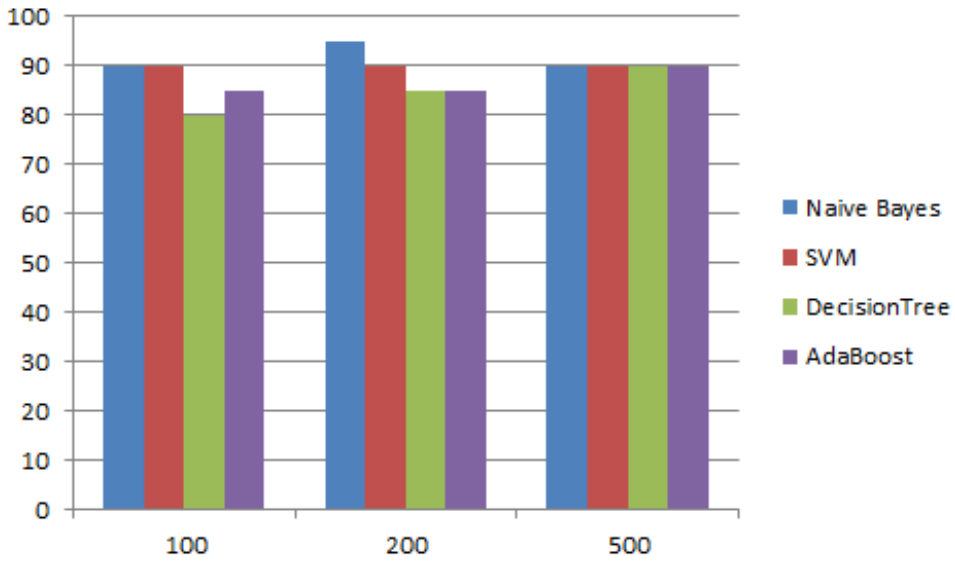


Figure 4.4: SIFT Descriptor using BoW Model prediction results with different dictionary sizes (100, 200 and 500) for the Recency Value.

4.1.2 Web Page Topic Results

4.1.2.1 Experiment 1 - Four classes

In [40], Boer et al. define four classes, that correspond to web site topics (newspapers, hotel, celebrities and conference sites). The classification results obtained were the following: when all features are used, an accuracy of 54% and 56% for the Naïve Bayes and the J48 respectively. Using the Color Histogram subset result in much worse performance of correct classifications. Using only the Edge Histogram attributes, the Naïve Bayes predict with an a accuracy of 58%, whereas the J48 predicts with 43% of accuracy. When they performed feature selection they show that the best predicting attributes are all from the Tamura and Gabor feature vectors. Using the top 10 attributes a prediction accuracy of 43% for both classifiers was obtained.

The standard features selected when selecting the top 1%, 5%, 10%, 20% and 50% attributes of the low-level descriptor using the Chi Square method and the PCA method are mostly from Tamura and Gabor Features.

Examining the results of the confusion matrix (tables 4.3, 4.4, 4.5, 4.6) that correspond to the best predictions of each classifier in Figure 4.5, it is easily verified that the classifiers struggle more with the celebrities web pages. The use of distinct colors and embedded videos can lead to errors the classifiers with the newspapers (lot of text followed with images) or hotel web pages (colorful background with photographies). The tests performed using different data size for the training of the classifiers can be found in appendix A.3.

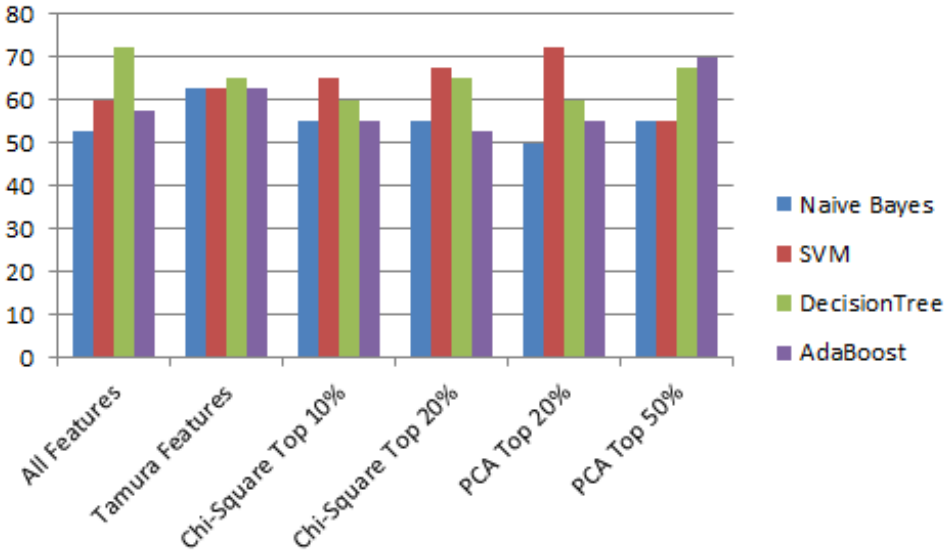


Figure 4.5: Best prediction results for the Topic web page for four different classifiers, using the low-level descriptor. All these predictions values, were obtain by training the classifiers using 90 images for each class.

Table 4.3: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **Naïve Bayes** classifier, using the low-level descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	7	0	1	0
	Conference	0	8	2	2
	Celebrity	2	1	6	3
	Hotel	1	1	1	5

Table 4.4: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **SVM** classifier, using the low-level descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	8	1	4	0
	Conference	2	9	1	0
	Celebrity	0	0	2	1
	Hotel	0	0	3	9

Table 4.5: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **Decision Tree** classifier, using the low-level descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	7	0	3	0
	Conference	0	9	0	0
	Celebrity	3	0	4	1
	Hotel	0	1	3	9

Table 4.6: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **AdaBoost** classifier, using the low-level descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	8	1	3	0
	Conference	0	6	0	0
	Celebrity	2	1	7	3
	Hotel	0	2	0	7

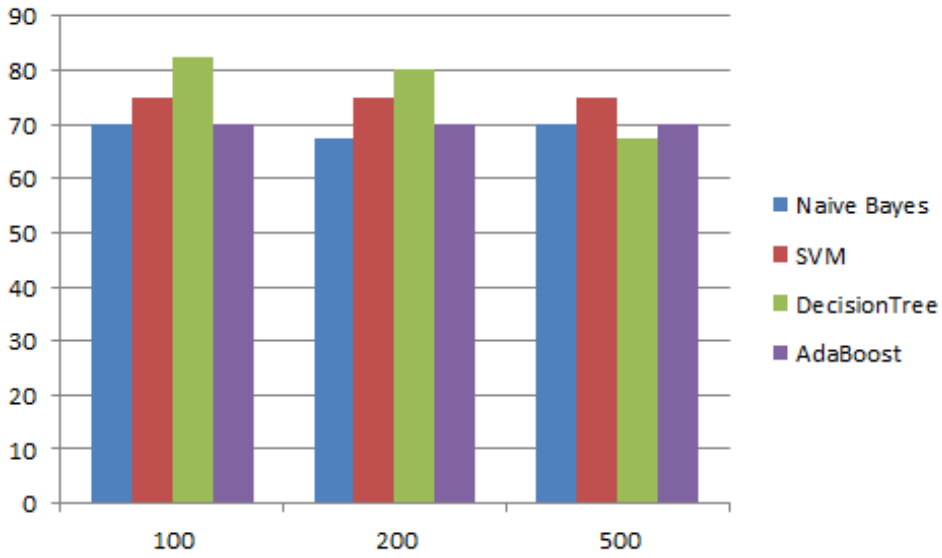


Figure 4.6: SIFT Descriptor using BoW Model best prediction results with different dictionary sizes (100, 200 and 500).

The results of the confusion matrix (tables 4.7, 4.8, 4.9, 4.10) correspond to the best predictions of each classifier using the SIFT with BoW model (Figure 4.6). It can be verified when analyzing the accuracy by class that the Naïve Bayes, Decision Tree and AdaBoost perform much worse for the Hotel class. The Naïve Bayes and AdaBoost classifiers reports false positives for the Hotel class as Conference or Celebrity pages. While the Decision Tree returns false positives for Celebrities web pages as Hotel web pages, and vice versa. The SVM classifiers performs much worse for the Celebrity web

pages, were most of the instances are wrong classified as Hotel web pages. Since the Newspapers and Conference class have simpler designs, when compared with the other classes, thus making it harder to distinguish between these two classes (Hotel and Celebrity).

Table 4.7: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **Naïve Bayes** classifier, using the SIFT descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	7	0	0	0
	Conference	2	7	2	2
	Celebrity	0	0	8	2
	Hotel	1	3	0	6

Table 4.8: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **SVM** classifier, using the SIFT descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	10	1	1	0
	Conference	0	8	1	0
	Celebrity	0	0	4	2
	Hotel	0	1	4	8

Table 4.9: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **Decision Tree** classifier, using the SIFT descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	10	0	1	0
	Conference	0	9	0	1
	Celebrity	0	1	7	2
	Hotel	0	0	2	7

Table 4.10: Confusion Matrix for 4 classes each with 10 web pages, for the best prediction result of the **AdaBoost** classifier, using the SIFT descriptor.

		Actual			
		Newspapers	Conference	Celebrity	Hotel
Predicted	Newspapers	10	0	1	0
	Conference	0	6	1	3
	Celebrity	0	1	8	3
	Hotel	0	2	0	4

In comparison with the results obtained by V. de Boer, M.W. van Someren and T. Lupascu, the results show in Figure 4.6 are an improvement in the accuracy of approximately 22% using the BoW model. Every classifier have an acceptable accuracy, where the best accuracy result is 82,5% for the Decision Tree using just 100 words to construct the dictionary. In fact all the classifiers have an accuracy grater than or equal to 70% when used just 100 words in the dictionary.

Using the same low-level descriptor they used all the classifiers obtain better results (the Naïve bayes had an accuracy of 62,5%, the SVM and Decision Tree had an accuracy rate of 72,5% while the AdaBoost classifier achieved an accuracy of 70%).

4.1.2.2 Experiment 2 - Eight classes

Along with the four classes defined in the experiment 1, four additional classes were added to this classification (classified advertisements sites, gaming sites, social networks sites and video-sharing sites).

Figure 4.7, shows the best accuracy for each classifier using the low-level descriptor. The Naïve Bayes had the best accuracy with 47,5% when trained with 60 images. The SVM achieved an accuracy of 41,25% also when trained with 60 images. The Decision Tree and AdaBoost classifiers had a poor performance, where the best accuracy was 37,5% and 33,75%, respectively. When used the Chi-Squared and PCA method to select the top attributes the classifiers performance didn't improve. We can conclude that for this type of classification is necessary more complex features.

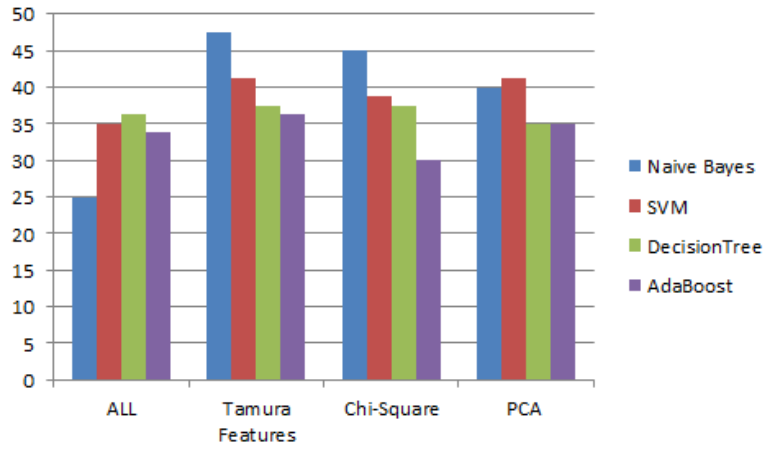


Figure 4.7: Best prediction results for the Topic web page, using the low-level descriptor. All these predictions values, were obtain by training the classifiers using 30 and 60 images for each class.

When used the SIFT descriptor (Figure 4.8) all the classifiers had a better accuracy related to the low-level descriptor. The SVM achieved an accuracy of 58,75%, and the Naïve Bayes 53,75%. The Decision Tree best accuracy was 48,75% , while the Adaboost only predict the correct class in 38,75%. All these tests were performed using different data size for the training of the classifiers can be found in appendix A.4.

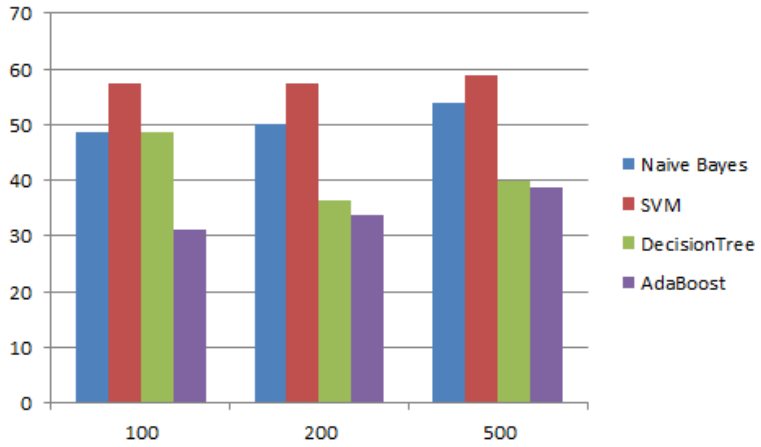


Figure 4.8: SIFT Descriptor using BoW Model best prediction results with different dictionary sizes (100, 200 and 500). All these predictions values, were obtained by training the classifiers using 30 and 60 images for each class.

In the confusions matrix (table 4.11 and 4.12), it is possible to verify that both classifiers have problems distinguishing celebrities web pages. The Naïve Bayes also struggles in identify Video-Sharing pages (only 3 correct predictions), the SVM have troubles in identify Social Networks web pages (only 2 correct predictions). The body of video-sharing web pages that consist mostly in video thumbnails, are easily mistaken as a newspapers web page (mostly images followed by text). In both classifiers some classifieds advertisements web pages are also predicted as newspapers (most classifieds advertisement websites use a simple color background with a lot of images).

Table 4.11: Confusion Matrix for 8 classes each with 10 web pages, for the best prediction result of the **Naïve Bayes** classifier, using the SIFT descriptor.

		Actual							
		Newsp.	Conf.	Celeb.	Hotel	Classif.	Gaming	Social N.	Video
Predicted	Newsp.	9	0	1	1	3	1	0	4
	Conf.	1	5	0	0	1	0	0	0
	Celeb.	0	0	3	2	0	2	2	1
	Hotel	0	1	0	5	0	1	1	0
	Classif.	0	1	1	1	6	0	0	1
	Gaming	0	0	5	0	0	6	0	0
	Social N.	0	1	0	1	0	0	6	1
	Video	0	0	0	0	0	0	1	3

Table 4.12: Confusion Matrix for 8 classes each with 10 web pages, for the best prediction result of the **SVM** classifier, using the SIFT descriptor.

		Actual							
		Newsp.	Conf.	Celeb.	Hotel	Classif.	Gaming	Social N.	Video
Predicted	Newsp.	9	1	1	1	4	0	1	2
	Conf.	1	8	0	0	0	0	0	0
	Celeb.	0	0	4	2	0	3	2	1
	Hotel	0	0	0	7	0	1	1	1
	Classif.	0	0	1	0	6	1	0	0
	Gaming	0	0	4	0	0	5	2	0
	Social N.	0	1	0	0	0	0	2	0
	Video	0	0	0	0	0	0	2	6

Chapter 5

Conclusions

In this work we described an approach for the automatic web page classification of web pages that uses the page rendered by a web browser visual content. The results obtained are quite encouraging, proving that the visual content of a page should not be ignored, when performing classification of web pages. This approach is therefore independent of the environment and format in which the site was constructed.

This implementation uses a method for categorization based on low-level features extracted from the page. Where different selection methods are applied to the feature vector. The results show that based on aesthetic value and recency, simple features such as color histogram and edge provide quite good results. For the classification of web pages by their topic, the use of a Bag of Words provide better results.

As expected when more websites topics are added to topic classification, the classification gets harder and the classifiers accuracy decreases. Which indicates that even if the pages have visual characteristic that distinguishes them, also have some attribute or characteristic in common. But the objective was to demonstrate that it is possible to classify web pages in different topics with some accuracy, and that the visual content should not be ignored.

Classification using the visual features has its advantages: if the page has poor quality, the accuracy in the classification will drastically be reduced. Other advantage is that many topic web page, have very common patterns in their design, making very hard to the classifier to distinguishe between them.

This work in addition to the scientific interest, may have practical uses: Assist in the combat of cyberterrorism (e.g., if a webcam is pointing to a computer), without knowing the URL of a website and only having the visual content of the same, having a database of dangerous websites, can be possible to detect a possible threat. This is only possible if all the terms of privacy are fulfilled.

A construction of an effective filter for improper visual content in web sites, based on the visual content of nude and non-nude images.

An advice system that assist the design and rating of web sites. It also enables a wide range of possibilities for research on the relation between visual features of websites and the effect that has on the users.

5.1 Future Work

Following the work developed in this thesis, another methods can be used to improve the classification of web pages using the visual content.

- *Since a good accuracy was achieved using the BoW model, the following methods may be used to improve the accuracy of the classifiers:*

  1. *An extension to the SIFT descriptor can be used, like the Hue-SIFT [24], that aimed at adding color information to the original SIFT.*
  2. *Another technique is the use of part dictionaries or "visual codebooks". These codebooks contain a variety of possible image structures. The basic idea is that websites with the same semantic meaning occur at the same location(s) on a web page. For example, in the case of newspaper websites have always lot of text followed with images in their body structure.*
  3. *The extraction of the BoW have a high computational cost, making those not scalable to large databases. Instead of using the classical approach based on k-means clustering algorithm, use a simple random selection, without statistically significant loss of information, but with a drastic decrease in the computational cost.*

- *Another approach is the extraction of more elaborate features. More local features can also have a positive effect (identifying the different visual elements on a web page), more abstract features can be used to better classify web pages.*

- *The integration of these visual features with other features of web pages. The analysis of the visual appearance of a web page can be combined with analysis based on textual content, URL, the underlying HTML, or other. In this case associate this visual features with the textual content, may give rise a powerful classification system.*

References

[1] *Alexa. Alexa the web information company, 2012.*

[2] *L. Andrade. The worlds ugliest websites!!! retrieved october 2009., 2009.*

[3] *Arul Prakash Asirvatham, Kranthi Kumar Ravi, Arul Prakash, Asirvatham Kranthi, and Kumar Ravi. Web page classification based on document structure, 2001.*

[4] *Bed and Breakfast. Bed and breakfast diamond collection, 2012.*

[5] *G. Bradski. The OpenCV Library.* Dr. Dobb's Journal of Software Tools*, 2000.*

[6] *Gary Bradski and Adrian Kaehler.* Learning OpenCV*. O'Reilly Media Inc., 2008.*

[7] *Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: web spam detection using the web topology. In* Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 423–430, New York, NY, USA, 2007. ACM.*

[8] *Dong Kwon Park Chee Sun Won and Soo-Jun Park. Efficient use of mpeg-7 edge histogram descriptor.* ETRI Journal*, 24:23–30, 2002.*

[9] *Rung Ching Chen and Chung Hsun Hsieh. Web page classification based on a support vector machine using a weighted vote schema.* Expert Syst. Appl.*, 31(2):427–435, 2006.*

[10] *Crazyleafdesign.com. Most beautiful and inspirational website designs, 2013.*

[11] *Thomas Deselaers. Features for image retrieval. Master's thesis, Human Language Technology and Pattern Recognition Group, RWTH Aachen University, Aachen, Germany, 2003.*

[12] *Sven Meyer Eissen and Benno Stein. Genre classification of web pages: User study and feasibility analysis. In* IN: BIUNDO S., FRUHWIRTH T., PALM G. (EDS.): ADVANCES IN ARTIFICIAL INTELLIGENCE*, pages 256–269. Springer, 2004.*

[13] *Gabriel Fiol-Roig, Margaret Miró-Julià, and Eduardo Herraiz. Data mining techniques for web page classification. In* PAAMS (Special Sessions)*, pages 61–68, 2011.*

[14] *Vicent Flanders. Worst websites of the year 2012 - 2005, 2012.*

[15] *Koraljka Golub and Anders Ardö. Importance of html structural elements and metadata in automated subject classification. In* Proceedings of the 9th European conference on Research and Advanced Technology for Digital Libraries*, pages 368–378, Berlin, Heidelberg, 2005. Springer-Verlag.*

[16] *Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update.* SIGKDD Explor. Newsl.*, pages 10–18, 2009.*

[17] *Nicholas Holden and Alex A. Freitas. Web page classification with an ant colony algorithm. In* IN PARALLEL PROBLEM SOLVING FROM NATURE - PPSN VIII, LNCS 3242*, pages 1092–1102. Springer-Verlag, 2004.*

[18] *Min-Yen Kan. Web page classification without the web page. In* Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 262–263, 2004.*

[19] *Min-Yen Kan and Hoang Oanh Nguyen Thi. Fast webpage classification using url features. In* Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 325–326, 2005.*

[20] *Milos Kovacevic1, Michelangelo Diligenti, Marco Gori, and Veljko Milutinovic1. Visual adjacency multigraphs, a novel approach for a web page classification.* Proceedings of the Workshop on Statistical Approaches to Web Mining (SAWM)*, pages 38–49, 2004.*

[21] *Oh-Woog Kwon and Jong-Hyeok Lee. Web page classification based on k-nearest neighbor approach. In* Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, pages 9–15, 2000.*

[22] *Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In* Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, TAI '95, Washington, DC, USA, 1995. IEEE Computer Society.*

[23] *Jialu Liu. Image retrieval based on bag-of-words model.* CoRR*, abs/1304.5168, 2013.*

[24] *Ana P. B. Lopes, Ra E. F. De Avila, Anderson N. A. Peixoto, Rodrigo S. Oliveira, and Arnaldo De A. Araujo. A bag-of-features approach based on hue-sift descriptor for nude detection, 2010.*

[25] *David G. Lowe. Distinctive image features from scale-invariant keypoints, 2003.*

[26] *Mathias Lux and Savvas A. Chatzichristofis. Lire: lucene image retrieval: an extensible java cbir library. In* Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 1085–1088, New York, NY, USA, 2008. ACM.*

[27] *The Internet Archive Wayback Machine. The internet archive wayback machine, 2009.*

[28] *Thomas M. Mitchell.* Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.*

[29] *Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms.* ACM Comput. Surv.*, 41(2):1–31, 2009.*

[30] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. ACM Comput. Surv.*, 41(2):12:1–12:31, February 2009.*

[31] Vladimir Risojević, Snježana Momić, and Zdenka Babić. Gabor descriptors for aerial image classification. In Proceedings of the 10th international conference on Adaptive and natural computing algorithms - Volume Part II*, ICANNGA'11, pages 51–60. Springer-Verlag, 2011.*

[32] Arthur L. Samuel. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development*, 44(1):206–227, 2000.*

[33] Ali Selamat and Sigeru Omatu. Web page feature selection and classification using neural networks. Inf. Sci. Inf. Comput. Sci.*, pages 69–88, January 2004.*

[34] Dou Shen, Zheng Chen, Qiang Yang, Hua-Jun Zeng, Benyu Zhang, Yuchang Lu, and Wei-Ying Ma. Web-page classification through summarization, 2004.

[35] Jonathon Shlens. A tutorial on principal component analysis. In Systems Neurobiology Laboratory, Salk Institute for Biological Studies*, 2005.*

[36] Matthew Shuey. 10-worst-websites-for-2013, 2013.

[37] Fengxi Song, Zhongwei Guo, and Dayong Mei. Feature selection using principal component analysis. In System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2010 International Conference on*, volume 1, pages 27–30, 2010.*

[38] Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Textural features corresponding to visual perception. IEEE Transaction on Systems, Man, and Cybernetics*, 8:460–472, 1978.*

[39] Jan Ulrich. Dividing the haystack: Web page classification, 2007.

[40] M.W. van Someren V. de Boer and T. Lupascu. Classifying web pages with visual features. WEBIST*, 2010.*

[41] waxy.org. Den.net and the top 100 websites of 1999, 2010.

[42] Wahyu Wibowo and Hugh E. Williams. Simple and accurate feature selection for hierarchical categorisation. In IN DOCENG 02: PROCEEDINGS OF THE 2002 ACM SYMPOSIUM ON DOCUMENT ENGINEERING*, pages 111–118. ACM Press, 2002.*

[43] Dengsheng Zhang, Aylwin Wong, Maria Indrawan, and Guojun Lu. Content-based image retrieval using gabor texture features. In IEEE Transactions PAMI*, pages 13–15, 2000.*

Appendix A

Classifiers Predictions

A.1 Aesthetic Value

A.1.1 Using the Low- Level Descriptor

Table A.1: Accuracy of each classifier when trained using **30 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	65	50	60	25	60
SVM	60	60	75	65	40
Decision Tree	65	40	50	55	75
AdaBoost	75	50	40	50	70

Table A.2: Accuracy of each classifier when trained using **60 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	75	70	60	50	60
SVM	55	55	75	50	40
Decision Tree	75	55	50	50	75
AdaBoost	65	60	40	55	70

Table A.3: Accuracy of each classifier when trained using **90 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	65	65	65	70	55
SVM	85	35	75	70	60
Decision Tree	70	45	55	60	75
AdaBoost	85	40	50	65	65

A.1.1.1 Using Chi Square Feature Selection

Table A.4: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	65	65	65	65	50
SVM	70	65	60	60	55
Decision Tree	55	50	50	50	75
AdaBoost	60	55	65	55	75

Table A.5: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	60	70	65	35	35
SVM	65	65	60	70	60
Decision Tree	55	55	55	55	80
AdaBoost	50	65	70	65	70

Table A.6: Accuracy of each classifier using the selected top attributes, when trained with **90 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	60	65	70	70	65
SVM	60	60	55	60	65
Decision Tree	55	55	55	55	80
AdaBoost	45	70	60	65	75

A.1.1.2 Using PCA Feature Selection

Table A.7: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	50	60	35	55	40
SVM	60	65	65	60	65
Decision Tree	45	65	60	60	80
AdaBoost	40	65	45	70	75

Table A.8: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	**5%**	**10%**	**20%**	**50%**
Naive Bayes	50	55	60	65	60
SVM	55	40	50	45	60
Decision Tree	70	65	50	50	50
AdaBoost	50	60	50	70	70

Table A.9: Accuracy of each classifier using the selected top attributes, when trained with **90 images** per class.

	1%	**5%**	**10%**	**20%**	**50%**
Naive Bayes	50	60	70	75	70
SVM	55	60	60	50	55
Decision Tree	70	75	75	80	75
AdaBoost	80	65	85	80	75

A.1.2 SIFT using BoW model for Aesthetic value

Table A.10: Accuracy of each classifier with a dictionary size of **100 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	75	80	70
SVM	65	80	80
Decision Tree	65	75	75
AdaBoost	50	85	55

Table A.11: Accuracy of each classifier with a dictionary size of **200 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	80	75	75
SVM	65	70	75
Decision Tree	60	80	75
AdaBoost	55	70	70

Table A.12: Accuracy of each classifier with a dictionary size of **500 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	70	70	80
SVM	65	80	80
Decision Tree	65	70	65
AdaBoost	60	70	80

A.2 Recency Value

A.2.1 Using the Low- Level Descriptor

Table A.13: Accuracy of each classifier when trained using **30 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	50	50	50	50	50
SVM	50	50	50	50	50
Decision Tree	45	45	50	50	45
AdaBoost	45	45	50	50	45

Table A.14: Accuracy of each classifier when trained using **60 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	55	50	60	50	50
SVM	50	75	55	65	75
Decision Tree	40	45	80	55	55
AdaBoost	35	35	85	60	50

Table A.15: Accuracy of each classifier when trained using **90 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	80	50	95	85	70
SVM	90	35	75	100	80
Decision Tree	95	55	90	95	90
AdaBoost	95	60	90	100	95

A.2.1.1 Using Chi Square Feature Selection

Table A.16: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	85	80	80	75	70
SVM	65	60	70	70	75
Decision Tree	90	90	90	90	85
AdaBoost	65	75	80	85	70

Table A.17: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	90	85	85	70	60
SVM	85	85	85	50	85
Decision Tree	90	90	90	90	90
AdaBoost	65	85	80	90	95

Table A.18: Accuracy of each classifier using the selected top attributes, when trained with **90 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	85	100	85	80	80
SVM	85	85	90	60	60
Decision Tree	90	90	90	90	90
AdaBoost	75	90	80	90	90

A.2.1.2 Using PCA Feature Selection

Table A.19: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	80	95	80	50	55
SVM	65	65	70	65	65
Decision Tree	95	95	95	95	90
AdaBoost	85	60	90	75	90

Table A.20: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	75	90	80	85	40
SVM	70	80	75	80	60
Decision Tree	95	95	95	90	95
AdaBoost	85	65	90	75	90

Table A.21: Accuracy of each classifier using the selected top attributes, when trained with **90 images** per class.

	1%	**5%**	**10%**	**20%**	**50%**
Naive Bayes	55	65	85	95	90
SVM	55	50	75	50	85
Decision Tree	55	90	95	70	90
AdaBoost	50	95	85	85	95

A.2.2 SIFT using BoW model for Recency value

Table A.22: Accuracy of each classifier with a dictionary size of **100 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	75	85	90
SVM	85	90	85
Decision Tree	70	80	75
AdaBoost	70	85	75

Table A.23: Accuracy of each classifier with a dictionary size of **200 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	75	85	95
SVM	85	90	85
Decision Tree	80	80	85
AdaBoost	75	80	85

Table A.24: Accuracy of each classifier with a dictionary size of **500 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	75	85	90
SVM	85	90	85
Decision Tree	80	90	90
AdaBoost	70	90	85

A.3 WebPage Topic for Four classes

A.3.1 Using the Low- Level Descriptor

Table A.25: Accuracy of each classifier when trained using **30 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	37,5	32,5	52,5	35	40
SVM	50	17,5	52,5	57,5	57,5
Decision Tree	67,5	30	65	62,5	70
AdaBoost	40	30	45	32,5	57,5

Table A.26: Accuracy of each classifier when trained using **60 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	55	27,5	60	32,5	45
SVM	60	22,5	60	50	62,5
Decision Tree	60	27,5	47,5	37,5	65
AdaBoost	47,5	25	60	50	67,5

Table A.27: Accuracy of each classifier when trained using **90 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	60	45	62,5	65	52,5
SVM	60	25	62,5	62,5	60
Decision Tree	60	40	65	45	72,5
AdaBoost	40	32,5	62,5	47,5	57,5

A.3.1.1 Using Chi Square Feature Selection

Table A.28: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	35	55	42,5	42,5	40
SVM	32,5	45	50	55	60
Decision Tree	47,5	65	65	65	67,5
AdaBoost	22,5	67,5	62,5	60	67,5

Table A.29: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	40	55	57,5	57,5	50
SVM	32,5	60	60	50	50
Decision Tree	45	47,5	47,5	57,5	62,5
AdaBoost	42,5	52,5	55	65	57,5

Table A.30: Accuracy of each classifier using the selected top attributes, when trained with **90 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	40	45	55	55	42,5
SVM	37,5	45	65	67,5	52,5
Decision Tree	45	42,5	60	65	65
AdaBoost	35	57,5	55	52,5	57,5

A.3.1.2 Using PCA Feature Selection

Table A.31: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	45	57,5	27,5	32,5	35
SVM	42,5	47,5	50	52,5	57,5
Decision Tree	30	57,5	45	45	47,5
AdaBoost	45	60	65	47,5	47,5

Table A.32: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	42,5	45	40	50	47,5
SVM	42,5	52,5	47,5	72,5	57,5
Decision Tree	30	47,5	47,5	60	67,5
AdaBoost	27,5	47,5	52,5	55	62,5

Table A.33: Accuracy of each classifier using the selected top attributes, when trained with **90 images** per class.

	1%	**5%**	**10%**	**20%**	**50%**
Naive Bayes	32,5	50	47,5	57,5	55
SVM	30	47,5	57,5	52,5	55
Decision Tree	45	47,5	55	55	67,5
AdaBoost	32,5	50	65	70	70

A.3.2 SIFT using BoW model for four classes

Table A.34: Accuracy of each classifier with a dictionary size of **100 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	72,5	62,5	70
SVM	70	70	75
Decision Tree	57,5	60	82,5
AdaBoost	62,5	60	70

Table A.35: Accuracy of each classifier with a dictionary size of **200 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	72,5	70	67,5
SVM	62,5	70	75
Decision Tree	60	67,5	80
AdaBoost	65	57,5	70

Table A.36: Accuracy of each classifier with a dictionary size of **500 words**, when trained with 30, 60 and 90 images per class.

Images per class	**30**	**60**	**90**
Naive Bayes	75	70	70
SVM	62,5	75	75
Decision Tree	50	65	67,5
AdaBoost	65	67,5	70

A.4 WebPage Topic for Eight classes

A.4.1 Using the Low- Level Descriptor

Table A.37: Accuracy of each classifier when trained using **30 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	35	15	40	22,5	22,5
SVM	38,75	13,75	36,25	28,75	31,25
Decision Tree	27,5	16,75	30	17,5	33,75
AdaBoost	22,5	15	26,25	17,5	22,5

Table A.38: Accuracy of each classifier when trained using **60 images** per class.

	HSV	Edge	Tamura	Gabor	ALL
Naive Bayes	32,5	15	47,5	22,5	25
SVM	41,25	13,75	41,25	33,75	35
Decision Tree	20	21,25	37,5	23,75	36,25
AdaBoost	30	13,75	36,25	18,75	33,75

A.4.1.1 Using Chi Square Feature Selection

Table A.39: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	26,25	31,25	36,25	28,75	31,25
SVM	30	26,25	38,75	33,75	38,75
Decision Tree	27,5	37,5	30	33,75	33,75
AdaBoost	22,5	23,75	30	26,25	21,25

Table A.40: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	5%	10%	20%	50%
Naive Bayes	21,25	41,25	45	41,25	26,25
SVM	16,75	45	43,75	51,25	30
Decision Tree	20	33,75	31,25	33,75	33,75
AdaBoost	20	35	27,5	36,25	30

A.4.1.2 Using PCA Feature Selection

Table A.41: Accuracy of each classifier using the selected top attributes, when trained with **30 images** per class.

	1%	**5%**	**10%**	**20%**	**50%**
Naive Bayes	21,25	22,5	40	22,5	20
SVM	21,25	23,75	32,5	41,25	26,25
Decision Tree	18,75	23,75	22,5	35	33,75
AdaBoost	18,75	24,75	18,75	30	23,75

Table A.42: Accuracy of each classifier using the selected top attributes, when trained with **60 images** per class.

	1%	**5%**	**10%**	**20%**	**50%**
Naive Bayes	20	23,75	33,75	12,5	25
SVM	12,5	32,5	35	36,25	40
Decision Tree	21,25	23,5	22,5	23,75	33,75
AdaBoost	18,75	25	35	26,25	28,75

A.4.2 SIFT using BoW model for 8 classes

Table A.43: Accuracy of each classifier with a dictionary size of **100 words**, when trained with 30 and 60 images per class.

Images per class	**30**	**60**
Naive Bayes	48,75	36,25
SVM	47,5	57,5
Decision Tree	33,75	48,75
AdaBoost	27,5	31,25

Table A.44: Accuracy of each classifier with a dictionary size of **200 words**, when trained with 30 and 60 per class.

Images per class	**30**	**60**
Naive Bayes	50	46,25
SVM	45	57,5
Decision Tree	36,25	36,25
AdaBoost	33,75	23,75

Table A.45: Accuracy of each classifier with a dictionary size of **500 words**, when trained with 30 and 60 per class.

Images per class	**30**	**60**
Naive Bayes	45	53,75
SVM	48,75	58,75
Decision Tree	40	33,75
AdaBoost	38,75	38,75