Classification of Web Elements Using Machine Learning

Erkka E. Virtanen

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 8.4.2019

Thesis supervisor:

Prof. Antti Oulasvirta

Thesis advisor:

D.Sc. (Tech.) Markku Laine

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Erkka E. Virtanen

Title: Classification of Web Elements Using Machine Learning

Date: 8.4.2019 Language: English Number of pages: 7+38

Department of Communications and Networking

Professorship: Human Computer Interaction

Supervisor: Prof. Antti Oulasvirta

Advisor: D.Sc. (Tech.) Markku Laine

Basic image segmentation is a fairly simple task for human beings and even young children can accomplish it naturally, but for machines, this can be a burdensome and difficult task. Segmenting large amounts of documents manually can be rather labour-intensive exercise and many gains in productivity could be had if machines could be automated to do the routine segmentation and classification tasks.

The website hosting company Suomen Hostingpalvelu Oy is transitioning from their old web site builder software to a new in-house developed site builder and they were faced with the problem of how to effortlessly allow users to move their old websites from the old site builder to the new one. This thesis explores the solution for this problem based on the fact, that the new site builder software uses semantic building blocks to build a website. By identifying the given semantic parts present on a given website through machine learning, we can provide the corresponding building blocks for site transitioning in the new site builder.

In this thesis, a novel way of segmenting web pages to their semantic parts is presented. This is accomplished by building a prototype which parses a given web site, gathers all the relevant features of the site's web elements and captures images of each web element. The gathered data is employed to create a training and testing data set which is used to train a machine learning model to classify web site segments. Three different machine learning algorithms, random forests, gradient boosting machines and a neural networks are examined and tested. After cross-validation, the highest achieved classification accuracy score of the trained machine learning model was a competent 81% allowing the prototype to be used in production at Hostingpalvelu. Finally, we will explore ideas for future research and for the improvement of the prototype.

Keywords: machine learning, classification, supervised learning, web element

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Erkka E. Virtanen

Työn nimi: Verkkosivuelementtien luokittelu koneoppimisen avulla

Päivämäärä: 8.4.2019 Kieli: Englanti Sivumäärä: 7+38

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Human Computer Interaction

Työn valvoja: Prof. Antti Oulasvirta

Työn ohjaaja: TkT Markku Laine

Kuvien jakaminen sen merkitseviin osiin on verrattain helppo tehtävä ihmisille ja jopa pienet lapset osaavat sen luonnostaan, mutta koneille tämä voi olla hyvinkin haastava tehtävä suoritettavaksi. Suurten tiedostomäärien käsin luokitteleminen ja osiin jakaminen on aikaa vievää ja työteliästä ja jos tietokoneet voitaisiin automatisoida tekemään nämä rutiinityöt, ihmiset voisivat ohjata työpanoksensa merkitsevämpiin asioihin.
IT-alan yritys Suomen Hostingpalvelu Oy on siirtymässä pois vanhasta kotisivukoneestaan uuteen talon sisällä kehitettyyn kotisivukoneeseen ja heillä oli ongelmana vanhojen kotisivujen siirto vanhasta kotisivukoneesta uuteen. Tämä diplomityö käsittelee tämän ongelman ratkaisemista perustuen siihen, että uusi kotisivukone käyttää semanttisia lohkoja sivujen rakentamiseen. Tunnistamalla vanhalla kotisivukoneella tehdyistä sivuista niiden semanttiset osat, voidaan sivuston siirto uuteen kotisivukoneeseen automatisoida.
Tässä diplomityössä esitellään uudenlainen lähestymistapa verkkosivun semanttiseen jakamiseen osiksi. Tämä tehdään rakentamalla prototyyppiohjelma, joka ensin jäsentää sille annetun verkkosivun, keräää jokaisen sivulla esiintyvän elementin ominaispiirteet ja ottaa niistä kuvat. Tästä datasta muodostetaan opetus- ja testidata, jolla opetetaan koneoppimismallia luokittelemaan verkkosivun semanttiset osat. Työssä esitellään kolme koneoppimisalgoritmia, random forests, gradient boosting machine ja neuroverkot, joita testataan prototyypissä. Ristiinvalidoinnin jälkeen korkein saatu luokittelutarkkuus oli 81%, joka on tarpeeksi korkea mahdollistaakseen prototyypin ottamisen tuotantokäyttöön Hostingpalvelulla. Lopuksi tutkimme vielä ideoita tulevaisuuden tutkimukseen ja mahdollisia tapoja, jolla prototyyppiä voitaisiin parantaa.

Avainsanat: koneoppiminen, luokittelu, ohjattu oppiminen, verkkosivuelementti

Preface

I want to sincerely thank Professor Antti Oulasvirta and Dr. Markku Laine for all the insight, advice and guidance that allowed me to finish this thesis and broadened my academic vision. I want to thank Hostingpalvelu's Jussi and Manu for allowing me to work on this thesis one day a week and for believing in me in the first place and giving me the chance to work in a fun company. I also want to thank my brother Tuukka for his ceaseless support at home. Also, thanks to my loving girlfriend Petra, without your endless love lifting me up this topic would have crushed me. I also want to thank my parents Jaana and Kari, for like bedrock they stand by me, and my good study friends Aku, Teemu, Jesse, Juha, Leo, Jaakko, Henri and Olli for showing me the example on how to be a great engineer and a life-loving teekkari.

I would like to end this preface with a quote from a terrific author, who was a hundred years ahead of his time, masterful with words and had a enough curiosity to feed a lifetime of boredom. To me, his quote crystallizes my journey of writing this thesis and the journey of life in general.

"But the man who comes back through the Door in the Wall will never be quite the same as the man who went out. He will be wiser but less cocksure, happier but less self-satisfied, humbler in acknowledging his ignorance yet better equipped to understand the relationship of words to things, of systematic reasoning to the unfathomable Mystery which it tries, forever vainly, to comprehend."

-Aldous Huxley,
The Doors of Perception & Heaven and Hell

Otaniemi, April 8, 2019

Erkka E. Virtanen

Contents

Abbreviations

CMS	Content Management System
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation

1 Introduction

Basic image segmentation is a fairly simple task for human beings and even young children can accomplish it naturally, but for machines, this can be a burdensome and difficult task depending on the situation and type of segmentation. But segmenting large amounts of documents by hand can be rather labour-intensive and boring exercise and many gains in productivity could be had if machines could be automated to do the routine segmentation and classification tasks. Also, by combining automatic segmentation with data mining of massive data sets, we could export more higher-level information from the old data and we could create new innovations and find new use-cases for them.

This thesis explores and tries to understand and develop an answer to a clear use case for classification of web elements using machine learning methods. The goal of this thesis is to segment web pages to semantic sections which can be understood by humans. To achieve this goal, we will create a prototype and a data set, which will be used to train a supervised learning algorithm to correctly categorize web page segments. The approaches and algorithms available to segmentation and classification are diverse, ranging from text classification[20] to neural networks[33] and graph theory[13].

1.1 Motivation

The main motivation for this research came from the web hosting industry. For the last year, I've been working as full-stack web developer for the Finnish hosting company Suomen Hostingpalvelu Oy and they had been offering a website builder for their customers for many years. This sitebuilder, called Web Presence Builder, allowed customers to create their own websites from different 'blocks' without the need for programming knowledge. The sites produced by the sitebuilder were sufficient at the time but lack certain aspects of the modern web like dynamic content, responsive designs and overall a more stylish look. For this reason, Hostingpalvelu is developing a brand new sitebuilder application in-house which I've had the opportunity to take part in developing in a great team. We have created the new sitebuilder with the same kind of 'block' segment logic to allow creation of websites without the need for programming similar to a Content Management System(CMS).

Hostingpalvelu has over two thousand sites hosted which are created with the old sitebuilder. These sites look a bit outdated and would benefit from updating their presentation by translating them to the new sitebuilder. Unfortunately, the old and the new sitebuilder are built with different technologies and going through all the old sites and translating them by hand is not a tempting proposition. From this dilemma, the thesis research problem is formulated. This thesis tries to find an answer to this problem by making the translation of these old sites to the new sitebuilder automatic which then allows the users to update their website's look and feel effortlessly.

1.2 Research question

The main research question of this thesis is "How to classify HTML documents to separate parts by their semantic function with machine learning". That is, given a website as input, how can the site be automatically classified to semantic sections like ”gallery”, ”blog post”, ”contact information” etc. utilizing machine learning techniques. To answer this question, a prototype application was developed applying the popular machine learning library h2o and a dataset of parsed web elements created. This thesis focuses only on static websites which omit the JavaScript functionality.

2 Background

In this chapter, we briefly define machine learning, review the relevant machine learning algorithms, define the related web technologies and page segmentation and end with related research by others.

2.1 Machine learning

One way of defining machine learning is to say, that it's the study of how a program or an algorithm progressively improves it's ability on a specific task based on previous data without human intervention [7]. For example, when viewing and searching for cat videos on YouTube, the service is collecting data on your behaviour and learning what kind of videos you like to watch and it will recommend more similar videos based on the viewing habits of other similar users and your past behaviour. Similar machine learning methods are now used widely on a wide-range of services ranging from Google to Facebook to enhance the user experience and to more accurately target and display adds.

Today, almost all of world's most valuable companies including Alphabet(Google's parent company), Amazon, Apple, Facebook and Microsoft are technology companies [21]. All of these enterprises are currently developing, investing and advancing more diverse machine learning applications and trying to find still uncovered niches to utilize machine learning in. This is partly because of the easy availability of data, which is now possible because of the mature state of the Internet infrastructure in the developed nations and the current social culture, in which people are willing and even want to share as much of their data as possible. These accruing large data masses couldn't be stored efficiently before the rise of affordable data-warehousing and fast and available bandwidth and connectivity. These data masses, dubbed ”Big Data”, are an essential part of large enterprises business plans in this new era in the market [14], in which data is called the new oil. But this data is not valuable without the insights from data mining and machine learning, as these new tools and techniques help to uncover the value which inherently lies to be discovered in the data centers' servers hard disks.

The first mathematical insights for machine learning were developed in the 1960's but the knowledge was kept in the academic world until the 2000's when newer applications of the machine learning started to emerge and the raw computing power and data requirements for modern algorithms were easier to meet. In the present day, almost all of the largest software projects including self-driving cars [8], artificial intelligence [2], social media feeds and speech-assistants [18] all utilize machine learning to some degree. Machine learning in itself is not some kind of ”magic bullet” which automatically improves an existing service or an application, as proper utilization of machine learning requires large amounts of good quality data and expertise to be of value. These requirements can also speed up the accumulation of machine learning expertise to the largest companies like Google and Apple which have large enough user bases to be able to amass enough high quality data to utilize machine learning techniques to their fullest potential.

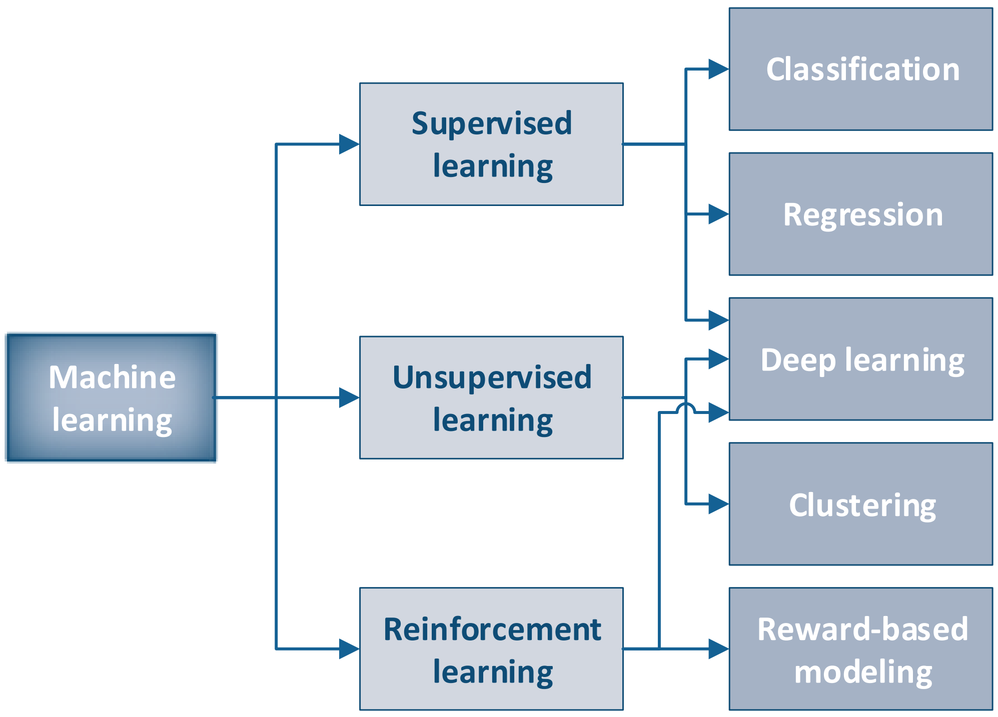


Figure 1: Machine learning concepts

2.1.1 Supervised learning

Supervised learning is a type of machine learning which utilizes large amount of data to ”teach” the machine learning application how to operate on another set of similar data [39]. These two data sets are called the training set and the testing set and they should contain comparable data. Each training data sample is composed of the vectorized sample data and a output goal value, which is also called the supervisory signal. The machine learning algorithm will try to infer the classification function by comparing the input output pairs of both data sets.

The raw training data is usually converted to a vector by a process called feature extraction. In feature extraction, the raw sample data, for example a positive comment on a website, is transformed into a vector by some chosen feature extraction algorithm, for example the different word frequencies in a sentence(called Bag-of-Words model). There are multiple choices to choose the actual learning algorithm depending on the amount of training data, number of dimensions of the feature space and computational speed among others. Learning algorithms used in supervised learning include support vector machines, decision trees, random forests, gradient boosting, deep learning, naïve bayes and linear regression algorithms.

2.1.2 Unsupervised learning

Unsupervised learning is a form of machine learning which doesn't require large training data sets with prelabeled classes as the problem of defining and finding the classes from the data is left for the algorithm. For example, a unsupervised learning algorithm can find and define specific clusters found in the data and label these as a distinct class from other clusters. Algorithms used for unsupervised learning include neural networks and k-means clustering.

2.1.3 Reinforcement learning

Reinforcement learning is the third paradigm of machine learning, which characterizes how a computer program with agency should select it's actions to maximize a numerical reward signal [26]. In other words, how should a computer program behave and map it's actions to its surroundings to optimize cumulative reward? The two important concepts of trial-and-error search and delayed reward are the characterizing features of reinforcement learning.

For a computer program to survive it's objective, it has to be able to do the three following things. It has to be able to take in signals from it's environment and to evaluate them and take actions accordingly to it's programming goals. The program must be able to possibly adapt its sub goals to serve its master goal of maximizing the cumulative reward over time. To model these three variables, the sensation, the action and the goal, we can use incompletely-known Markov decision processes. [26]

Examples of reinforcement learning include video games, which are an uncomplicated platform for reinforcement learning applications, as internal goals of the game can be used for the metrics of the application's reward system. Then, increasing the computer player's score or skill level in the game, displays a direct improvement on the performance of the model. Reinforcement learning has been used for example to build chess playing artificial intelligence [43].

2.2 Algorithms

In this section we will detail the three relevant machine learning algorithms that will be used by the prototype.

2.2.1 Random forests

Random forests algorithm functions by creating a forest of weak learners(decision trees) which vote on the final classification class by majority vote. Thus, random forests is an ensemble machine learning method. A formal definition of random forests algorithm according to Breiman [11] is as follows:

A random forest is a classifier consisting of a collection of tree-structured classifiers h(x, k), k = 1,... where the k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x.

A single decision tree can suffer from high variance and can be susceptible to noise, but as each tree is generated independently from a random subset of the data, the majority voting on the multiple decision trees reduces variance and cancels out potential errors in the model. In other words, random forests models are not prone to overfitting [25] and are robust against missing data points, but at times it can be difficult to interpret what features or choices the model made and why.

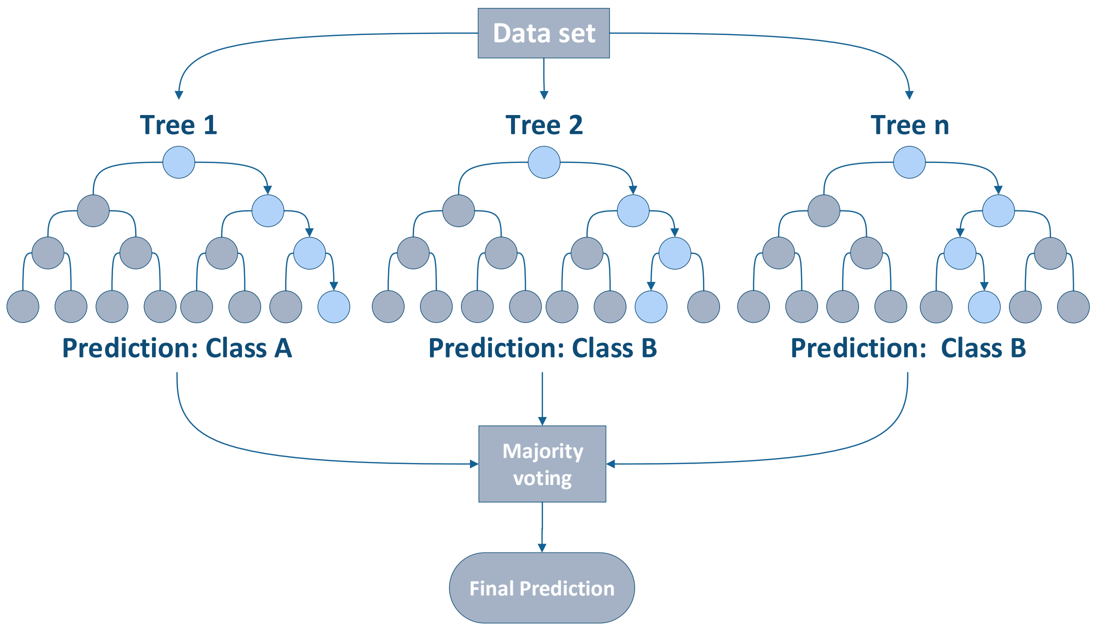


Figure 2: Random forests are comprised of multiple decision trees which vote on the final class prediction

2.2.2 Gradient boosting machine

The gradient boosting machine model builds decision trees sequentially. It starts by creating a simple model(one decision tree) and calculating its accuracy and error. It then creates a model of the difference between the target function and the prediction and combines this residual model with the current model. This process is then repeated and with each repetition a new tree is added and the model tries to minimize the residual by adjusting the weights for all previous decision tree predictions. [37]

Gradient boosting is similar to the random forests algorithm in that they both start with many weak learners(decision trees) that are combined to create one strong learner [22]. Also, with random forest algorithm, each decision tree is independent from one another, but in gradient boosting they are dependant as each consecutive tree solves for the net error of prior trees. They are also a more prone to over fitting and to noise than random forests.

Gradient boosting is used widely in many different domains with different implementations and variations. Most notable ones are XGBoost [15] and Adaboost [40]. For example, gradient boosting is used by the Yahoo! search engine [16].

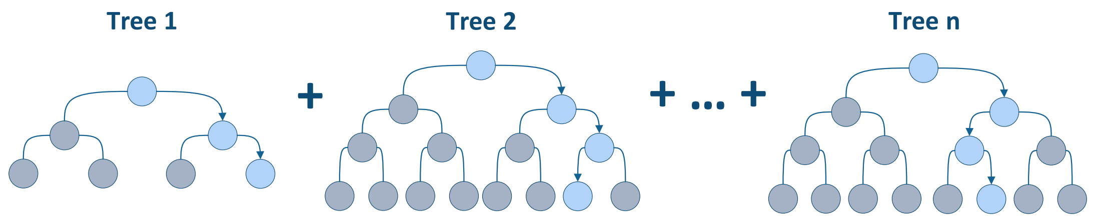


Figure 3: The gradient boosting machine combines many weak learners and calculates the difference between the target function and prediction after adding each new tree.

2.2.3 Multilayer feedforward neural network

Many different algorithms and implementations fall under the neural network term, such as Convolutional Neural Networks and Recurrent Neural Networks, but here we present the multilayer perceptron (multilayer feedforward neural network) as implemented in h2o machine learning library.

The multilayer perceptron [38] always has at least three layers, the input layer, the hidden layer and the output layer. First on the input layer we have the input data and on the output layer we have the wanted output classes. Each layer has a set amount of nodes which use tanh as the linear activation function.

The h2o implementation uses stochastic gradient descent as the back-propagation method. This means looking at the desired output and what our current model predicts, and if the prediction is wrong, we propagate the error backwards the hidden layers calculating the gradients and adjusting the weights of all the nodes in the hidden layers.

Today, the multilayer perceptron is one of the most common and straight forward types of neural networks. The most common applications include speech recognition, image recognition and machine translation [45]. The current state-of-the-art neural networks are more intricate and complicated than a simple multilayer perceptron and can produce results which rival even the human experts such as Google Deepmind's AlphaGo [42].

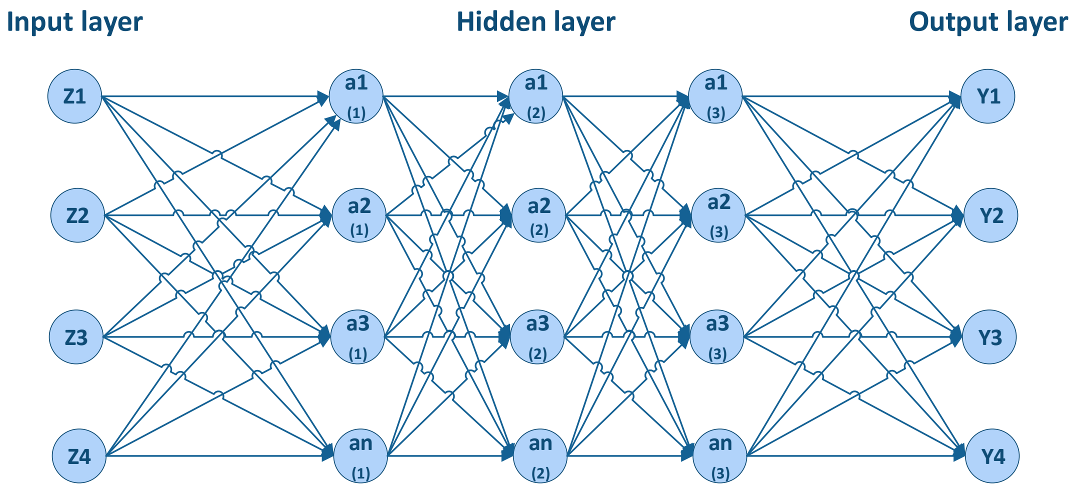


Figure 4: A neural network consisting of the input layer, three hidden layers and the output layer.

2.3 Web technologies

A modern functional website, rendered in a browser with animations and dynamic content, is comprised of several interlocked parts. As the basis we have the HTML document and it's Document Object Model(DOM), the appearance or styling of the document with Cascading Style Sheets(CSS) and lastly ECMAScript(implemented as JavaScript), the official scripting language of the HTML5 specification. A modern dynamic website is usually making many calls to other servers for downloading dynamic content like images and comments or just for applying specific tracker scripts and utilities(for example Google Analytics tracker script). With JavaScript the whole DOM and styling of the website can be changed dynamically according to events and functions executed by the user. These modern dynamic websites are in contrast to more traditional static websites which don't make requests to other servers. For the purposes of this thesis, we will focus only on static websites which omit the JavaScript functionality.

2.3.1 HTML

Hyper Text Markup Language or HTML, is the standard language of internet websites. It was first proposed by Time Berners-Lee in 1990 and today there exists over 1.5 billion websites [44]. HTML document describes the page semantically with different semantic portions separated with angle brackets. These HTML elements can contain for example images, text and interactive inputs.

When a user surfs the web and visits a domain to view a web page, a HTML document is requested from a server. The document is then interpreted and rendered by the web browser with the aid of the accompanying style definitions and executable code. When the browser interprets the received HTML document, it reconstructs the document in its internal presentation to a Document Object Model, which the user then interacts with through the browser.

2.3.2 Document Object Model

The Document Object Model is a language agnostic specification for HTML documents, which underlies the structure of a website [46]. The DOM is comprised of different elements, such as <body>, <h1>, <a> as specified in the HTML5 documentation. For example, the <p> tag, marks a paragraph on a website. The DOM elements can be placed inside other elements producing a hierarchical structure with parent and child nodes. The full representation of this hierarchical arrangement is called the DOM tree. This logical and hierarchical structure of the DOM allows for consistent parsing and rendering of the document. [19]

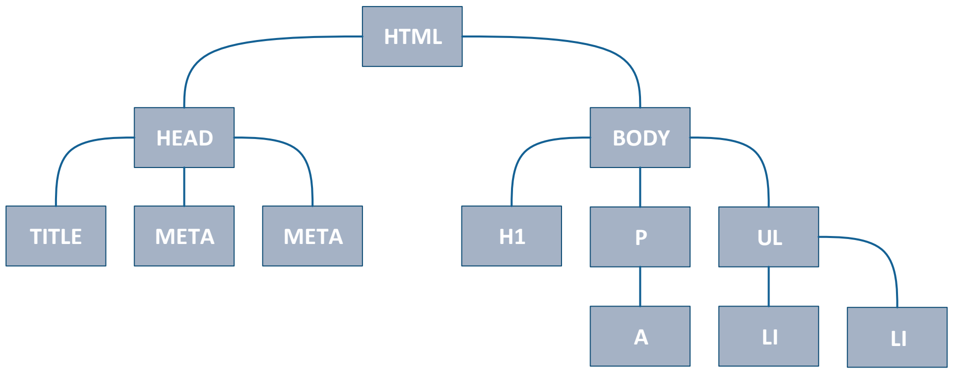


Figure 5: Visualization of the hierarchical structure of the Document-Object-Model(DOM) of a simple website.

2.3.3 Cascading Style Sheets

Cascading style sheets(CSS) is a style sheet language used to style and add animations to web pages [31]. CSS defines the presentation of all aspects of a page's web elements, from their size to color and from their font families to animations.

CSS language is defined by rules made of selectors and declarations. The selector part defines the scope of the rule and the declaration determines the wanted effect or function. Selectors can be tags, pseudo-classes or classes, which are arbitrarily defined strings appointed by the developer. The selectors can include logic rules to scope the rules effectiveness. The declaration part of the rule specifies the desired functionality, like the color or size of a text block.

The utilization of a separate styling language from the document's structure(HTML markup) allows for the separation of content and presentation. This enables the effortless adjustment of a web site's aesthetic without altering its functionality and allows users to modify their web clients styling, improving the user experience. For example, the user can change the browsers default styling to enhance the contrast or to increase text size for readability and accessability. [32]

2.4 Page segments

Hostingpalvelu's new sitebuilder website building logic is based on building blocks(called ”sections”) which comprise the whole website. At the moment of writing, the currently implemented list of categories in the new sitebuilder is comprised of Hero, Gallery, Story, Video, Map, Contact, Features, Products, Menu, Team, and Quote sections. Next we will present and define the selected six segments for the prototype.

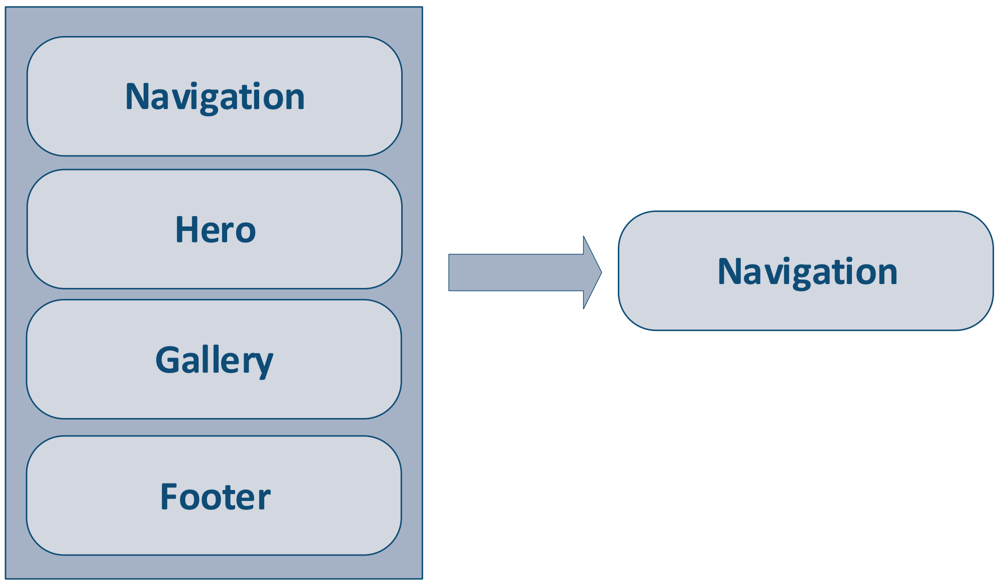


Figure 6: Example of semantic segments of a page

2.4.1 Navigation

The navigation category comprises of different implementations of the classical horizontal navigation bar, which holds the links to other parts of the website such as contact or gallery pages. Vertical navigation bars are also included in this category.

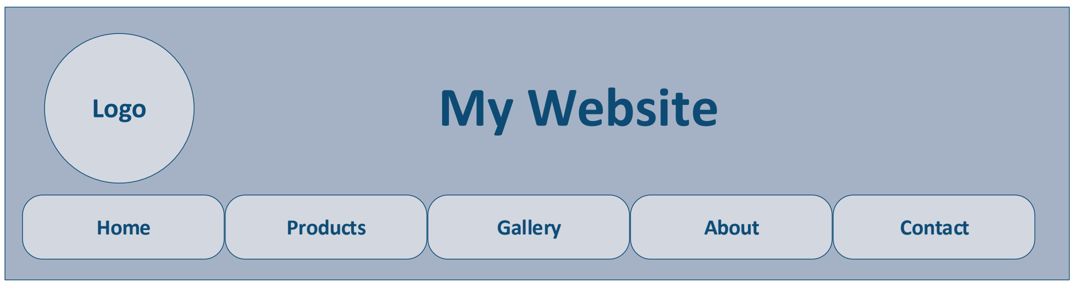


Figure 7: Example of the navigation category

2.4.2 Hero

Hero is a common term used in web development to describe the first eye-catching element of a website which usually has the most important visual information that showcases the main idea or products of the website. In this data set it usually comprised of a large image with interlaced text and a call-to-action button.

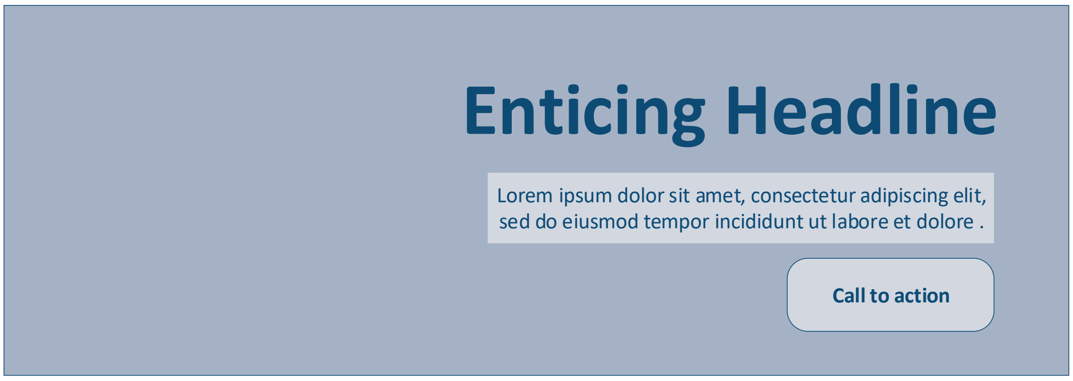

Enticing Headline
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore .
Call to action

Figure 8: Example of the hero category

2.4.3 Gallery

The gallery category encompasses different kinds of image galleries and carousel elements commonly found on webpages. They are usually laid out in a grid formation or on a carousel layout, which scrolls through images.

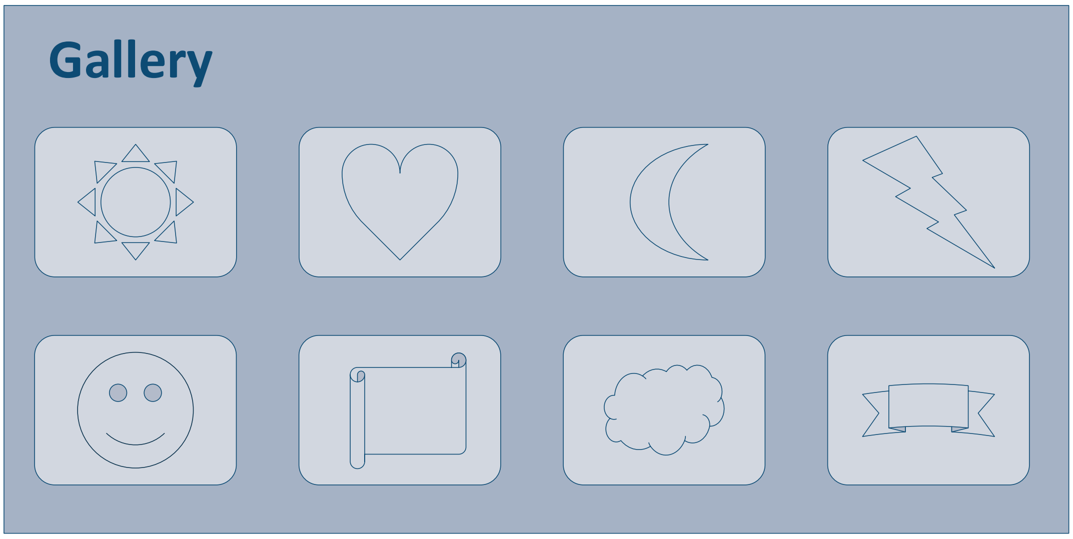


Figure 9: Example of the gallery category

2.4.4 Story

The story category can be varied, but it usually contains a text block, explaining the purpose of site, display latest developments on the site or showcase the products of a given website. This can be text only block or contain an image in conjunction with the text.

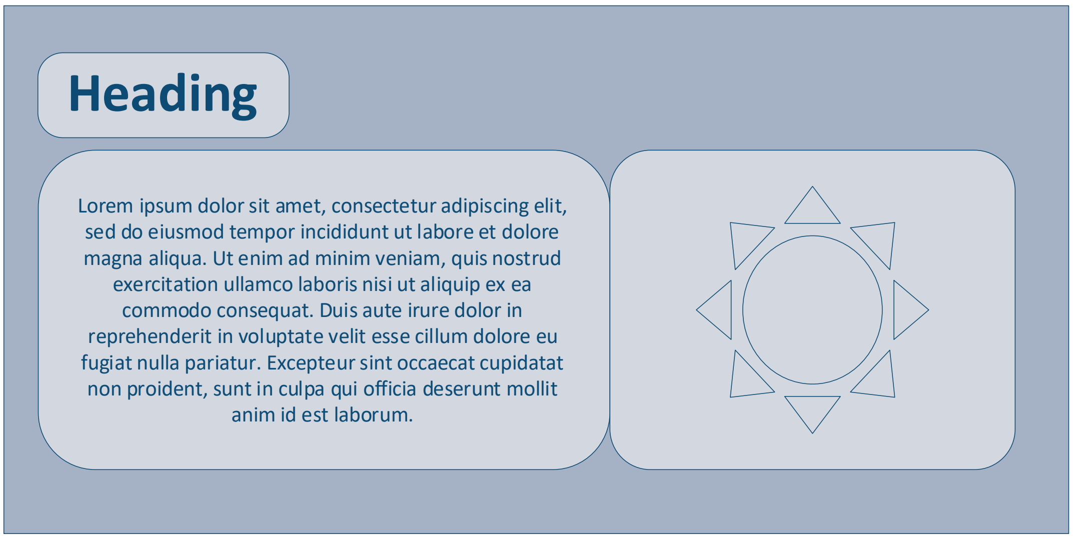

Heading
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Figure 10: Example of the story category

2.4.5 Contact

The contact section displays the contact details, usually describing a phone number or a street address or directions. Sometimes the contact section is paired with a embedded Google Maps widget.

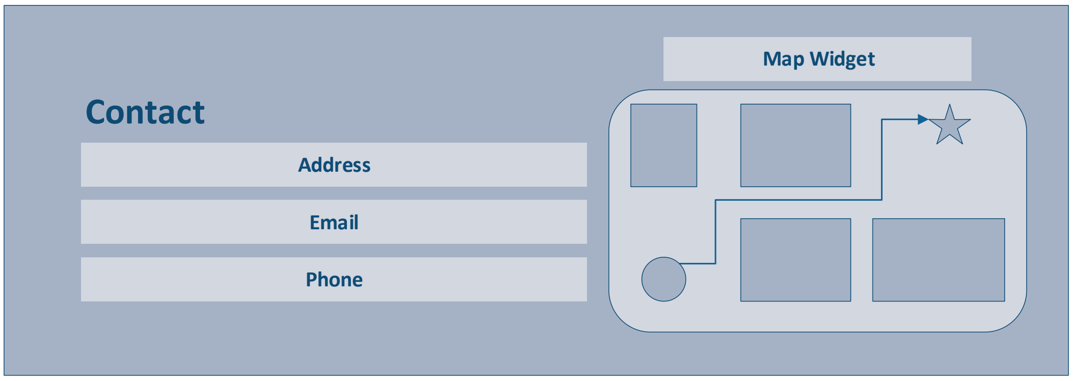


Figure 11: Example of the contact category

2.4.6 Footer

The footer section is at the bottom of the page and it usually displays the copyright notice and navigation or links or an image.

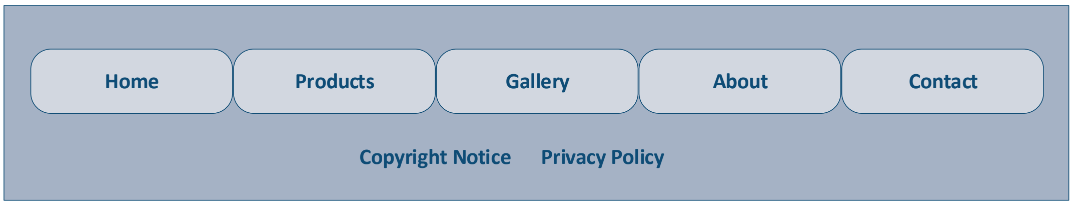


Figure 12: Example of the footer category

2.5 Related work

When considering the previous related work on classification of web documents, multiple different approaches have been studied. Some approaches have been based on image recognition [30] [1], Natural Language Processing(NLP), DOM-parsing and ensemble methods combining multiple techniques. The current state-of-the-art implementations combine visual and DOM-parsing approaches to achieve the best results.

A recent paper published by Bakaev et al. [4] implemented a vision based tool, build on top of the computer vision library OpenCV [9], which takes a screenshot of a webpage and produces a JSON file [10] as the output with all the human readable interface elements. The upside of this completely visual-based implementation is that what the machine learning algorithm sees is exactly the same as what the human user sees. In contrast, a wholly DOM based approach in which the CSS and JavaScript can arbitrarily change the visual appearance and location of elements of the original DOM tree, can look quite different when the final page is rendered in the browser. This makes the mapping of an element's visual look and it's data structure more difficult for machine learning algorithms.

Other notable element classification research based on visual features include VIPS, an algorithm which detects the websites layout from its visual representation [12] and Bento algorithm [29], which is an ensemble approach used to transfer the content from another website to another layout.

3 Research material and methods

To answer the research question of this thesis with a concrete solution, a working prototype is required. This chapter will go through the different parts, theoretical underpinnings and engineering choices that were chosen for the prototype.

3.1 Goal of the prototype

As outlined in the first chapter, the goal of the prototype is, given a HTML-document as input, to be able to classify different parts of the website by their semantic function using machine learning algorithms. This requires multiple steps. First, the prototype program must be able to parse the relevant data from the HTML-document and extract the relevant feature vectors from the data to create a training and testing data sets. Then a supervised learning algorithm is taught with the training data set. After adjusting and fine tuning the algorithm, the newly-trained classifier is tested on the training data set and it's accuracy is cross-validated. Once validated, the program should be able to recognize and classify new HTML-documents as intended.

To achieve this goal, Python 3 [36] with Selenium [41] and h2o [23] were chosen as the development tools for the prototype. Python is a widely used programming language, Selenium is a widely used web automation tool and h2o is a multi-platform machine learning library. The prototype is built from three parts, the parser, the classification tool and the data set and the machine learning model implementation. Next, we will go through each part detailing the engineering and design choices.

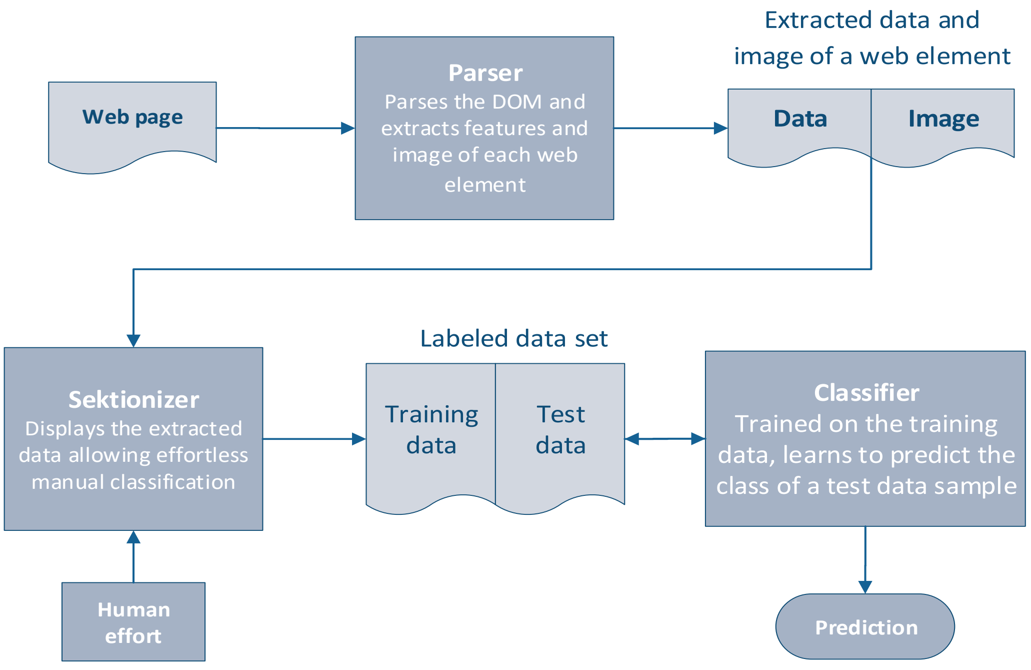

Web page
Parser
Parses the DOM and extracts features and image of each web element
Extracted data and image of a web element
Data
Image
Sektionizer
Displays the extracted data allowing effortless manual classification
Labeled data set
Training data
Test data
Classifier
Trained on the training data, learns to predict the class of a test data sample
Human effort
Prediction

Figure 13: Overview of the functions of the prototype

3.1.1 Parser

The first part of the prototype is the HTML document parser. This is achieved using the Selenium, a widely used web automation tool. The prototype fetches the desired website using Selenium and Google Chrome webdriver and creates a Selenium Web Element abstraction of the underlying DOM tree. These elements can then be accessed by their methods and properties to extract relevant feature data, such as the size, location and styling of a web element. For each parsed web element, a JSON file is generated which holds all the parsed feature data and which is later used in the data set creation process.

The built parser prototype can take five to ten minutes to parse a given a website with long nested DOM nodes. This can be made faster by restricting the parsing depth. For example, the prototype can be adjusted to not parse a DOM node if it has more than three levels of children for example. This is a justified engineering choice, as our objective is to classify quite high-level and abstract categories like ”navigation”, which itself can contain many levels of nested elements. The lowest three level of the elements children don’t provide more information than the highest level, which already saves the number of different child elements.

3.1.2 Features

Choosing the correct features to extract from the data is a critical task for successful classification, as web elements are presented in multitude of different colors, sizes, contrasts and amounts of text and all of these variables contribute to the upper level characteristic of a certain class. These different features are then distilled and grouped to a feature vector, which represents all of the requested features of a single data point [7].

Large number of feature vectors can be both favourable and unfavourable depending on the context. More feature vectors can increase the classification accuracy but too many feature vectors will introduce the ”curse of dimensionality” [5] [27]. This means, that as the number of feature vectors increase, the volume of the solution space will increase in size exponentially, requiring very large data sets for the results to be statistically significant.

Kumar et al. [29] identified many of these prominent web element feature vectors as did Minukovich et al [34]. Based on their work and by selecting the most practicable and computable features, the following features were chosen to be parsed by the prototype:

Feature	Explanation
Horizontal sidedness	How far is the element from the horizontal centerline of the webpage
Vertical sidedness	How far is the element from the vertical centerline of the webpage
Left sidedness	How far is the element from the utmost left of the webpage
Top sidedness	How far is the element from the top height of the webpage
Location	X and Y position of the web element
Size	Width and height of the web element
Aspect ratio	Width of the web element divided by height of the element the web element
Tag	Type of the tag of the web element
Class	CSS class or classes of the web element
Parent	Parent element of the web element and its type of tag
Style	Computed style of the web element, the relevant style properties of the web element
Text content	The text content of the web element
Word count	Number of words in the web element
Image count	Number of images in the web element
Link count	Number of links in the web element
Child count	Number of div, p, ul, li, span, h1, h2 and h3 children of the web element

Figure 14: Selected features for the prototype

3.1.3 Data set

One of the most important parts of any machine learning endeavour is the data set, as a properly curated and procured data set will in part define the success of the whole prototype. Because acquiring a decent data set is usually a laboursome and demanding task, in general it's good advice to use an external data set if possible. Scouring different sources, the following possible data set was identified.

Rico [17], the largest mobile app data set to date. It contains data from over 9.7 thousand Android Apps and spans 27 different categories. Rico would have been perfect to use in this project, have it not been a mobile only data set.

None of the other data sets were deemed not suitable for this research task, so a new web element data set built from the ground up was needed. This process started by researching the literature of web element classification and deciding whether to use a only DOM based or visual based or a ensemble method. A hybrid approach was decided, which parses the DOM and saves each element with its CSS styling and also captures a picture of the element. This is accomplished by the the parser part of the prototype. When parsing the document, each element's data is saved as a JSON-file containing all the feature vector data and a picture file taken from a particular element.

```json
{
    "depth": 2,
    "tag": "li",
    "id": 67,
    "class": "normal navigation-item-expand",
    "location": {
        "x": 44,
        "y": 340
    },
    "size": {
        "height": 32,
        "width": 273
    },
    "textContent": "\n\t\t\t\n\t\t\t\n\t\t\t",
    "aspectRatio": 8.53125,
    "numImages": "0",
    "numLinks": "8",
    "horizontalSidedness": -0.9140625,
    "verticalSidedness": -0.11458333333333333,
    "leftSidedness": 0.04296875,
    "topSidedness": 1.5572916666666665,
    "wordCount": 3,
    "style": {
        "bottom": "auto",
        "box-shadow": "none",
        "cursor": "auto",
        "float": "none",
        "font-family": "Tahoma, sans-serif",
        "font-size": "13px",
        "font-weight": "400",
        "height": "32px",
        "justify-items": "normal",
        "left": "auto",
        "letter-spacing": "normal",
        "line-height": "18.46px",
        "margin-bottom": "4px",
        "margin-left": "0px",
        "margin-right": "0px",
        "margin-top": "0px",
        "max-height": "none",
        "max-width": "none",
        "min-height": "0px",
        "min-width": "0px",
        "overflow-x": "visible",
        "overflow-y": "visible",
        "padding-bottom": "0px",
        "padding-left": "0px",
        "padding-right": "0px",
        "padding-top": "0px",
        "text-align": "left",
        "text-shadow": "none",
        "top": "auto",
        "width": "273px",
        "word-break": "normal",
        "z-index": "auto",
        "align-items": "normal",
        "flex-basis": "auto",
        "flex-grow": "0",
        "flex-shrink": "1",
        "flex-direction": "row",
        "flex-wrap": "nowrap",
        "justify-content": "normal",
        "order": "0",
        "transform": "none",
        "fill": "rgb(0, 0, 0)",
        "opacity": "1"
    },
    "count": {
        "div": "0",
        "p": "0",
        "ul": "2",
        "li": "7",
        "span": "16",
        "h1": "0",
        "h2": "0",
        "h3": "0"
    },
    "parent": {
        "tag": "ul",
        "class": "navigation"
    },
    "label": "navigation"
}
```

Listing 1: Example of a single parsed DOM element

After the parsing of a website is complete, the human part of the data set creation begins with the manual classification of each web data element to the proper category. For this purpose a data set building tool called Sektionizer was built with Python 3. Sektionizer allows a user to input a web address and to start the parsing process, which saves all the parsed web element JSON files and images into a single folder. After parsing, the user can open the folder in Sektionizer and cycle through the images while classifying the element by clicking the labeled class buttons. This tool speeds up the manual process of parsing and labeling the data set and its graphical user interface allows also non-technical users to grow the size of data set after teaching the user the specifications of each class.

Also, as the parsing of a website can take five to ten minutes, a parse batching utility was created. This tool inputs a list of sites to be parsed and starts parsing through it. This saves time, as the batching utility can run its parsing during the night and leave only the classification process left for the user.

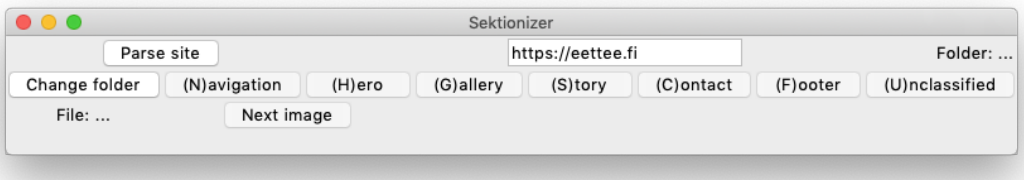


Figure 15: Graphical user interface of the built Sektionizer prototype

3.1.4 Classifier

The final part of the prototype is the classifier. Its role is to create a machine learning model based on the training data and a chosen algorithm and then it's accuracy is evaluated by cross-validating the model with the testing data set.

First version of the classifier was built with the scikit-learn machine learning library [35], but ultimately, the java virtual machine based h2o [23] was chosen as the implementation library, since it provides greater assortment of machine learning algorithms and allows the random forest algorithm to work better with categorical data.

Choosing the correct machine learning algorithm is of great importance, as it in great part defines the accuracy of our prototype. Three different machine learning algorithms were utilized with the prototype, random forests, multilayer feedforward neural networks and the gradient boosting machine algorithm. The results of the models built with these three algorithms are showcased in the next chapter.

4 Results

This chapter will first go through the results of the prototype utilizing the three different machine learning algorithms described in the previous chapter and evaluate them.

Variable	Mean	Standard Deviation	Valid #1	Valid #2	Valid #3	Valid #4	Valid #5
Accuracy (%)	0.811	0.046	0.792	0.742	0.909	0.860	0.750
Error (%)	0.189	0.046	0.208	0.258	0.091	0.140	0.250
Log loss	0.966	0.288	0.828	0.794	0.450	1.087	1.672
Max per class error	0.720	0.243	1.000	0.313	0.286	1.000	1.000
Mean per class accuracy	0.830	0.054	0.794	0.848	0.964	0.807	0.734
Mean per class error	0.170	0.054	0.206	0.152	0.036	0.193	0.266
Mean squared error	0.200	0.036	0.232	0.250	0.131	0.144	0.242
R2	0.934	0.013	0.923	0.905	0.955	0.953	0.932
Root mean square error	0.443	0.042	0.482	0.500	0.362	0.380	0.492

Table 1: Cross-validation results for the random forest model

4.1 Random forest

4.1.1 Cross-validation scores

The random forest model was trained on a data set comprised of 309 web element feature vectors which was split to 75% training and 20% testing sets. The model was validated with k-cross-validation with k value of five.

The cross validation scores for the random forest algorithm are shown in Figure 16. The accuracy score mean is almost 81% which is an adequate result.

4.1.2 Most and least important features

The model identified the CSS style font-weight as the most important feature of the model with a percentage weight of 13.4%. The five least important features were all styling features with all of them having marginal or no importance.

4.1.3 Confusion matrix

From the confusion matrix we can see what classes are identified erroneously by the model. There are some error with classifying the footer as the story section and also with story and hero, as they can also have similar characteristics such as position and image in conjunction with text.

Variable	Relative importance	Scaled importance	Percentage
styleFontWeight	3483.298	1.000	0.134
styleTop	2014.869	0.578	0.078
class	1691.500	0.486	0.065
verticalSidedness	1287.124	0.370	0.050
textContent	1265.136	0.363	0.049
—	—	—	—
styleFontSize	17.124	0.005	0.001
styleMaxHeight	13.552	0.004	0.001
styleOrder	12.239	0.004	0.000
styleTextAlign	10.362	0.003	0.000
styleFill	9.602	0.003	0.000

Table 2: The five most and least important features identified by the random forest model

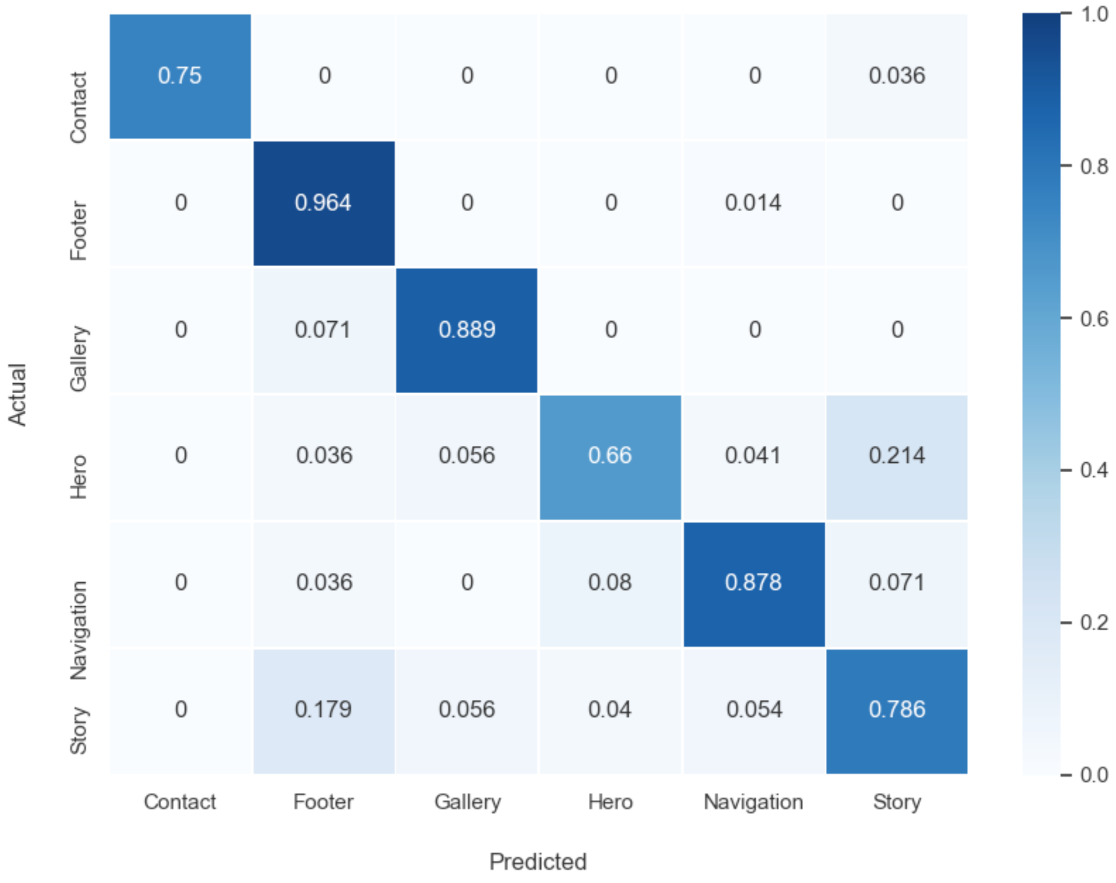


Figure 16: Confusion matrix for the random forests model

4.1.4 Duration

On a Macbook Pro of 2014 with a 2.6 GHz Intel Core i5 and 8 GB 1600 MHz DDR3 the model took 44.2 seconds to train.

	Mean	Standard deviation	Valid #1	Valid #2	Valid #3	Valid #4	Valid #5
Accuracy (%)	0.738	0.037	0.729	0.677	0.795	0.800	0.688
Error (%)	0.262	0.037	0.271	0.323	0.205	0.200	0.313
Log loss	2.075	0.681	2.424	3.324	0.960	0.939	2.726
Max per class error	0.780	0.192	1.000	0.400	0.500	1.000	1.000
Mean per class accuracy	0.763	0.070	0.752	0.810	0.888	0.776	0.589
Mean per class error	0.237	0.070	0.248	0.190	0.113	0.224	0.411
Mean squared error	0.228	0.043	0.238	0.300	0.162	0.155	0.284
R2	0.924	0.016	0.921	0.885	0.945	0.949	0.921
Root mean square error	0.473	0.046	0.488	0.548	0.402	0.393	0.533

Table 3: Cross-validation results for the gradient boosting model

4.2 Gradient boosting machine

The results for the gradient boosting machine model.

4.2.1 Cross-validation scores

The gradient boosting machine model was trained on a data set comprised of 309 web element feature vectors which was split to 75% training and 20% testing sets. The model was validated with k-cross-validation with a k value of five.

The cross validation scores for the gradient boosting machine model are shown in Figure 17. The mean accuracy score is 73.8% which is the lower compared to the random forest model but on the same level as the neural network model.

4.2.2 Most and least important features

The model chose the styleFontWeight as the most important feature with importance weight of 49.5%, so it is a very significant feature for this model's classification process. Then is the vertical location, web element's parent class, the height of the web element and the textual content of the element. These feature vectors seem very intuitive to a web developer as these features are widely set in the web elements and a they have lot of communicative power to affect the website's look.

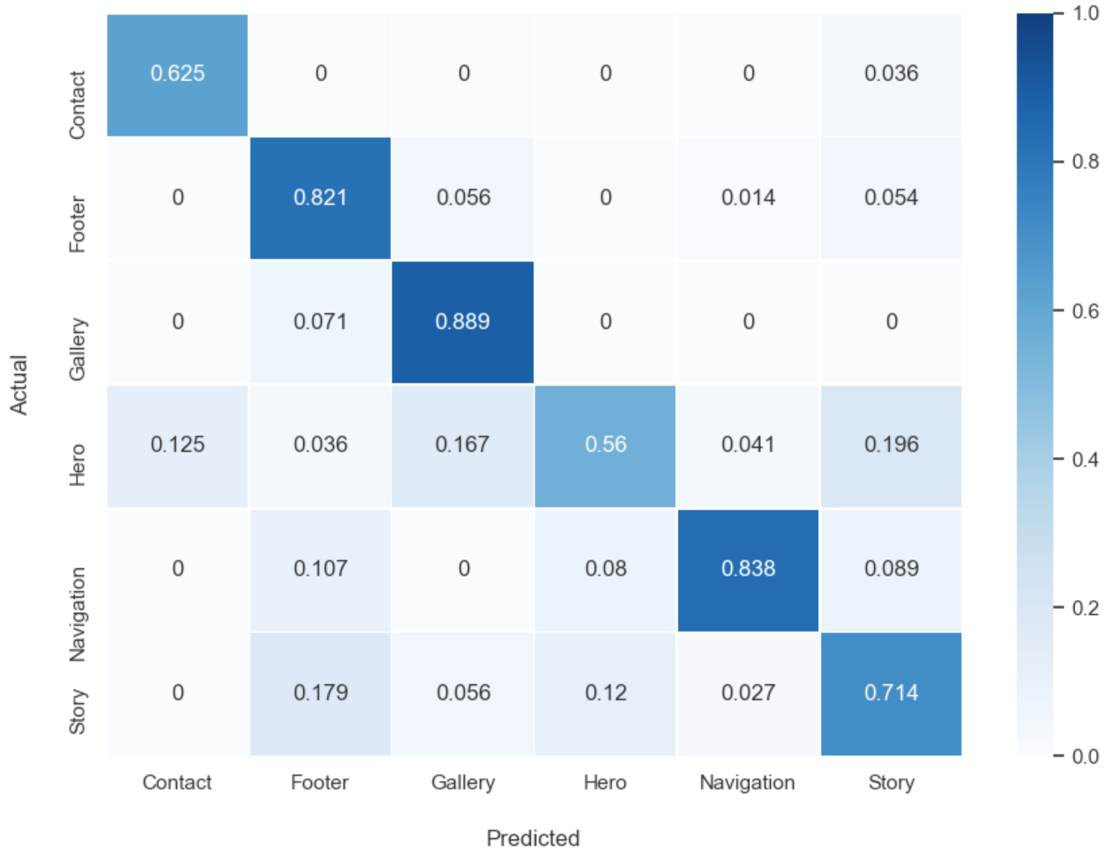
The five least important values are the stylePaddingLeft, styleTextAlign, style-WordBreak, styleOrder and lastly styleFill. All of them shared the value of zero. One explanation for this could be, that these style attributes are used so widely that their visual value might not convey enough information in this classification, or that the training set didn't have enough examples of these values being set.

Variable	Relative importance	Scaled importance	Percentage
styleFontWeight	333.498	1.000	0.495
locationY	59.549	0.179	0.088
parentClass	41.565	0.125	0.062
height	38.692	0.116	0.057
textContent	34.202	0.103	0.051
—	—	—	—
stylePaddingLeft	0.000	0.000	0.000
styleTextAlign	0.000	0.000	0.000
styleWordBreak	0.000	0.000	0.000
styleOrder	0.000	0.000	0.000
styleFill	0.000	0.000	0.000

Table 4: The five most and least important features identified by the gradient boosting model

4.2.3 Confusion matrix

The confusion matrix for the gradient boosting model is quite similar to the deep learning model. They both have quite fairly results with identifying the same type of classes, but with this model the miss labeled predictions are spread out more evenly over multiple classes. Still, the largest error is with classifying the footer class as the story class and the hero class with the story classes as was with the random forest model.



Actual \ Predicted	Contact	Footer	Gallery	Hero	Navigation	Story
Contact	0.625	0	0	0	0	0.036
Footer	0	0.821	0.056	0	0.014	0.054
Gallery	0	0.071	0.889	0	0	0
Hero	0.125	0.036	0.167	0.56	0.041	0.196
Navigation	0	0.107	0	0.08	0.838	0.089
Story	0	0.179	0.056	0.12	0.027	0.714

Figure 17: Confusion matrix for the gradient boosting machine model

4.2.4 Duration

On a Macbook Pro of 2014 with a 2.6 GHz Intel Core i5 and 8 GB 1600 MHz DDR3 the model took 26.9 seconds to train, which is significantly faster than the random forest model and within similar level of the neural network model.

	Mean	Standard Deviation	Valid #1	Valid #2	Valid #3	Valid #4	Valid #5
Accuracy (%)	0.738	0.046	0.646	0.677	0.795	0.760	0.813
Error (%)	0.262	0.046	0.354	0.323	0.205	0.240	0.188
Log loss	1.332	0.333	2.240	1.013	0.928	1.218	1.263
Max per class error	0.802	0.174	1.000	0.438	0.571	1.000	1.000
Mean per class accuracy	0.778	0.055	0.661	0.797	0.904	0.764	0.766
Mean per class error	0.222	0.055	0.339	0.203	0.096	0.236	0.234
Mean squared error	0.228	0.049	0.348	0.268	0.169	0.180	0.177
R2	0.923	0.019	0.884	0.898	0.942	0.941	0.951
Root mean square error	0.473	0.050	0.590	0.518	0.411	0.424	0.420

Table 5: Cross-validation results for the neural network model

4.3 Multilayer feedforward neural network

4.3.1 Cross-validation scores

The multilayer feedforward neural network model was trained on a data set comprised of 309 web element feature vectors which was split to 75% training and 20% testing sets. The model was validated with k-cross-validation with k value of five.

The cross validation scores for the neural network model are shown in Figure 18. The accuracy score mean is almost 73.8% which is adequate and matched by the gradient boosting model's performance but less successful than the random forest model.

4.3.2 Most and least important features

The model identified the web element wordCount as the most important feature of the model with a percentage weight of 0.002%. Similar to other models, the bottom most important features are styling or missing tag features, which affect the models decisions marginally not at all. The deep learning model's reasoning behind these feature choices are difficult to analyze, but wordCount, width and height seem like a intuitively reasonable features for classifying web elements.

4.3.3 Confusion matrix

The model had the largest challenge at identifying the story class by confusing it with contact, footer and hero classes. This can be explained by the comparable features of these classes, as they all have text and can contain images and can be placed on similar locations on the web page.

Variable	Relative importance	Scaled importance	Percentage
wordCount	1.000	1.000	0.002
countP	0.913	0.913	0.002
countSpan	0.874	0.874	0.002
width	0.869	0.869	0.002
height	0.830	0.830	0.002
—	—	—	—
styleMarginBottom.5px	0.597	0.597	0.001
styleTop.958.547px	0.597	0.597	0.001
stylePaddingLeft.50px	0.586	0.586	0.001
tag.missing(NA)	0.000	0.000	0.000
parentTag.missing(NA)	0.000	0.000	0.000

Table 6: The five most and least important features identified by the neural network model

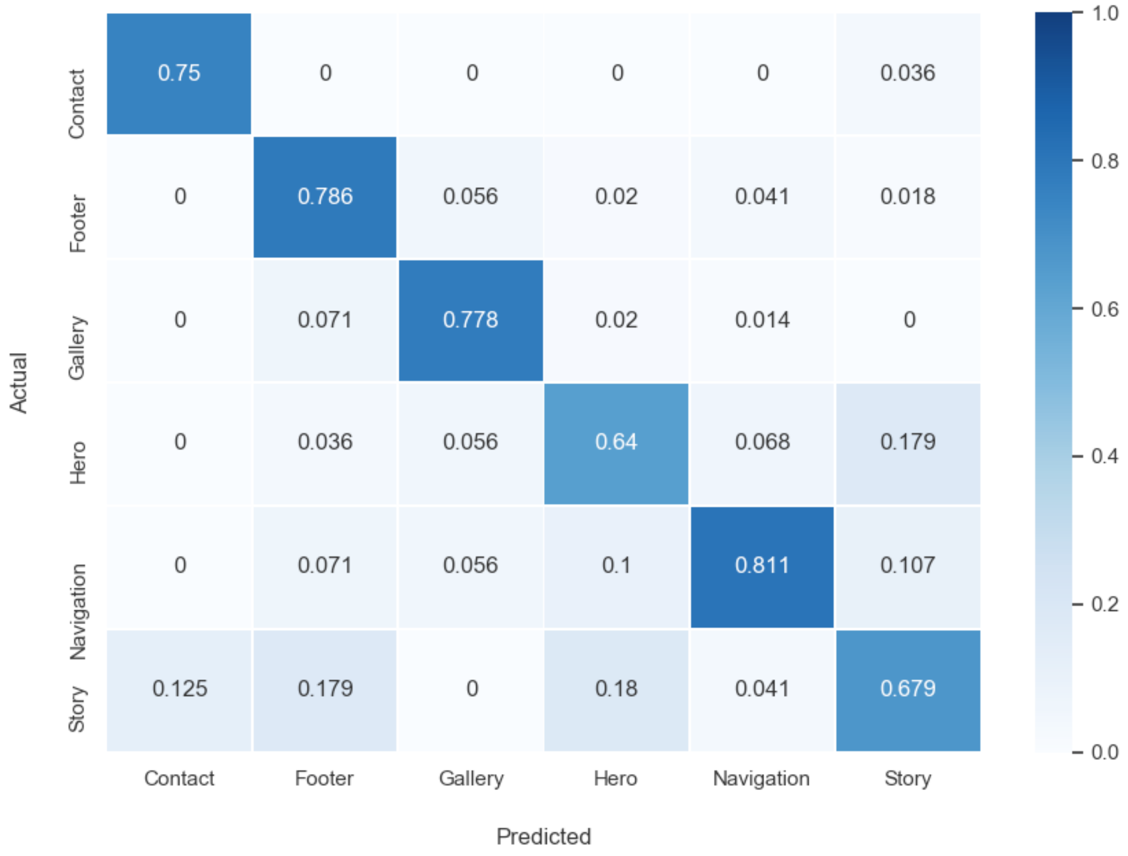


Figure 18: Confusion matrix for the neural network model

4.3.4 Duration

On a Macbook Pro of 2014 with a 2.6 GHz Intel Core i5 and 8 GB 1600 MHz DDR3 the model took 23.5 seconds to train, which is the smallest duration of the three models and almost two times faster than the random forest model.

4.4 Evaluation of the results

Comparing the results of all the three shown models, the random forest with 81%, the gradient boosting model with 74% and the multilayer feedforward neural network with 74%, the highest accuracy was scored by the random forest model. The random forest model accuracy is the highest with a seven percent margin and the gradient boosting and the neural network scored a matching accuracy score. The result is not such a surprise, as the random forest models have been shown to be particularly effective in sorting mostly categorical data.

The neural network model's choices for the most important features were not easily interpreted intuitively, so it's hard to say what kind of rules the neural network model "learned". Both the random forests and gradient boosting models chose the font weight and the vertical location on their top five most important features. These seem more intuitive and logical, as they both contribute greatly to a web element's look and semantics.

From these results the best model to use in our prototype is the random forest model. The random forest model also had the largest duration for the model training, but as the durations still are quite small(44.2s vs 23.5s), the computational load isn't that too large of a significance.

We can now investigate the accuracy score results other known approaches in the web element classification research. For comparison, there exists VIPS [12] by Microsoft Research, which is a vision based page segmentation algorithm. By their performance testing, it classified the semantic content with a 93% accuracy. The accuracy of the developed prototype and the accuracy of VIPS give some general insight to their predictive power, but for a direct comparison to be made, access to the same training data VIPS utilized would be required.

	Accuracy	Error	LogLoss	Mean per class accuracy	Mean per class error	Training duration
Random forest	0.811	0.189	0.966	0.830	0.170	44.2
Gradient boosting machine	0.738	0.262	2.075	0.763	0.237	26.9
Neural network	0.738	0.262	1.332	0.778	0.222	23.5

Table 7: Comparison of the results of the three models

5 Summary

This chapter will overview the thesis research question, discussion and the results, the feasibility of the build prototype in production use and ideas for future research.

5.1 Overview

This thesis started with the research question of ”How to classify HTML-documents to separate parts by their semantic function with machine learning” and we have looked at the background of machine learning in general, the machine learning algorithms and their functionality, the document object model, the development of the prototype, and showcased the results of three different machine learning models.

Chapter three detailed the building of the prototype, which is separated into three parts. First is the parser, which parses the given website into discrete web elements and creates a data file containing the relevant features and an image file of the element. Secondly is the data set, which was built with the Sektionizer tool by manually classifying the parsed data by their semantic function. And lastly, the machine learning models, which classify the given web elements according to the fed training and testing data sets.

The prototype achieved a five-fold cross validated accuracy rating of 81%, which is an adequate result for production usage at Hostingpalvelu, though the final production tool will possibly require some added heuristics and adjustments.

5.2 Discussion

How to improve the accuracy of the prototype? One answer is to improve the data sets. By gathering a larger data set we can cover more edge cases which help the machine learning model separate between harder to distinguish sections such as ”Hero” and ”Story” classes. Other possibility could be adding more features to the parsed data, but the currently incorporated features already implement the ones found in literature.

One possible improvement to the prototype could be the addition of a natural-language-processing(NLP) machine learning model. As the encompassing text of each web element is already collected, the textual data could be further fed to a specifically taught machine learning model, possibly utilizing the bag of words-model with a support vector machine or a naïve bayes algorithm to gain additional information from the text to help the classification of the segment.

Trying different machine learning algorithms could also improve the accuracy. The current state-of-the-art deep learning neural network models use massive amounts of data coupled with large server clusters utilizing ensemble techniques incorporating many different models sequentially. These require a high-level of expertise to be tuned properly to avoid over fitting and bias. But even with the created prototype, fine-tuning of the parameters and improving the size and quality of the training data we could see an improvement in the accuracy score.

To enable faster creation of new data sets, the parser's speed of parsing a website could be improved from the current five to ten minutes. This could be done by identifying the most time-consuming parts of the parser's functions and seeking known faster alternative programming maneuvers.

5.3 Case fit for Hostingpalvelu

The research question for this thesis came from the web development industry, so it is important to asses the built prototype's feasibility in production use.

At Hostingpalvelu, the prototype will be used to translate customer's websites built with the older site builder tool Web Presence Builder to Hostingpalvelu's new in-house developed site builder Oidom. An automation tool will be built on top of the research presented in this thesis, which will automatically offer users a new Oidom website based on data from their old web site. The new tool will run the prototype's machine learning model on the old web site and output the semantic sections which are present on the site. It will then offer these identified sections as already filled in building blocks on the new site builder.

The prototypes accuracy result of 81% is precise enough to be competent, but will probably require some added heuristics and possibly building a larger data set to cover more edge cases and to improve the classifier accuracy when used in production.

Also, in the future, the prototype could be adapted to work with websites created with other site builders. This would allow the customer to transition from a different hosting provider's sitebuilder tool Oidom. This would require teaching the machine learning algorithm with a different training data, data which is parsed from sites generated with a particular tool. This would massively boost the usefulness of the prototype.

5.4 Future research

To improve the prototype further some questions remain to be answered by future research. Firstly, how would adding a image recognition classifier in conjuction with the current DOM data based approach improve the results of the classification? As we already have captured the images of each web element by the current parser, the image data for this concept is already collected. This could be done with a convolutional neural network model, which are widely used in image recognition and can achieve high accuracy [28].

To gain a better understanding of the underlying classification process, a better explanation of the choices a machine learning model made is required. What reasons made the machine learning model classify this as a ”navigation” rather than a ”hero” class? For example, the random forest model's most and least important features seem logical to a web developer, but the corresponding features for the deep learning neural network are non-intuitive for humans. This acknowledged machine learning as a black box problem [6] is a massive dilemma in different domains utilizing machine learning from medical [24] to financial industries [3] and can cause wrong decision to be made, endangering lives and creating unjust situations where decisions can't be backed by sound and logical step-by-step reasoning.

One arduous part of this project was the data set generation. Could this process somehow be made more effortless or the prototype be converted to work on mobile data set like Rico? Supervised learning models do always require the learning signal or label to be taught, but there are ways to bypass the manual process of labeling each data point by hand. For example, one could create a tool which generates real looking web layouts at random with the given labeling inherently built-in into the layout creation process and then teach the machine learning model with it. This approach could bypass the manual labour in classifying each element, but the results and accuracy are not know. It is an idea worth exploring, as the data set building process can be very laborious and time-consuming.

References

[1] Anton Akusok, Alexander Grigorievskiy, Amaury Lendasse, Yoan Miche, T Villmann, and FM Schleif. Image-based classification of websites. *Machine Learning Reports*, 2:25–34, 2013.

[2] Itamar Arel, Derek C Rose, Thomas P Karnowski, et al. Deep machine learning-a new frontier in artificial intelligence research. *IEEE computational intelligence magazine*, 5(4):13–18, 2010.

[3] Bart Baesens, Rudy Setiono, Christophe Mues, and Jan Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management science*, 49(3):312–329, 2003.

[4] Maxim Bakaev, Sebastian Heil, Vladimir Khvorostov, and Martin Gaedke. Hci vision for automated analysis and mining of web user interfaces. In *International Conference on Web Engineering*, pages 136–144. Springer, 2018.

[5] R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. *Dynamic Programming.* Rand Corporation research study. Princeton University Press, 1957.

[6] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, 8(5):1156–1164, 1997.

[7] Christopher M Bishop. Pattern recognition and machine learning (information science and statistics) springer-verlag new york. *Inc. Secaucus, NJ, USA*, 2006.

[8] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

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

[10] Tim Bray. The javascript object notation (json) data interchange format. Technical report, 2017.

[11] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[12] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Vips: a vision-based page segmentation algorithm. 2003.

[13] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. A graph-theoretic approach to webpage segmentation. In *Proceedings of the 17th international conference on World Wide Web*, pages 377–386. ACM, 2008.

[14] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile networks and applications*, 19(2):171–209, 2014.

[15] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[16] David Cossock and Tong Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

[17] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 845–854. ACM, 2017.

[18] Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, 2013.

[19] Terence Eden, Shwetank Dixit, Scott O'Hara, Bruce Lawson, Xiaoqian Wu, Sangwhan Moon, and Patricia Aas. HTML 5.3. W3C working draft, W3C, October 2018. https://www.w3.org/TR/2018/WD-html53-20181018/.

[20] Sébastien Eskenazi, Petra Gomez-Krämer, and Jean-Marc Ogier. A comprehensive survey of mostly textual document segmentation algorithms since 2008. *Pattern Recognition*, 64:1–14, April 2017.

[21] Forbes. The world's most valuable brands. `https://www.forbes.com/powerful-brands/list/`, 2018. [Online; accessed 13-March-2019].

[22] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[23] H2o. An open source artificial intelligence platform. `https://www.h2o.ai/`, 2019. [Online; accessed 27-February-2019].

[24] Anna Hart and Jeremy Wyatt. Evaluating black-boxes as medical decision aids: issues arising from a study of neural networks. *Medical informatics*, 15(3):229–236, 1990.

[25] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[26] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[27] Eamonn Keogh and Abdullah Mueen. *Curse of Dimensionality*, pages 314–315. Springer US, Boston, MA, 2017.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[29] Ranjitha Kumar, Jerry O Talton, Salman Ahmad, and Scott R Klemmer. Bricolage: example-based retargeting for web design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2197–2206. ACM, 2011.

[30] Leonardo Espinosa Leal, Kaj-Mikael Björk, Amaury Lendasse, and Anton Akusok. A web page classifier library based on random image content analysis using deep learning. In *Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference*, PETRA '18, pages 13–16, New York, NY, USA, 2018. ACM.

[31] Håkon Wium Lie and Bert Bos. Cascading style sheets, level 1. W3C recommendation, W3C, September 2018. https://www.w3.org/TR/2018/SPSD-CSS1-20180913/.

[32] Hakon Wium Lie, Bert Bos, C Lilley, and I Jacobs. *Cascading style sheets.* Pearson India, 2005.

[33] Pikakshi Manchanda, Sonali Gupta, and Komal Kumar Bhatia. On the automated classification of web pages using artificial neural network. *IOSRJCE, ISSN*, pages 2278–066, 2012.

[34] Aliaksei Miniukovich and Antonella De Angeli. Computation of interface aesthetics. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1163–1172, New York, NY, USA, 2015. ACM.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[36] Python. A general purpose programming language. https://www.python.org/, 2019. [Online; accessed 27-February-2019].

[37] Greg Ridgeway. The state of boosting. *Computing Science and Statistics*, pages 172–181, 1999.

[38] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961.

[39] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[40] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[41] Selenium. A web browser automation tool. https://www.seleniumhq.org/, 2019. [Online; accessed 27-February-2019].

[42] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[43] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[44] Internet Live Stats. Total number of websites. http://www.internetlivestats.com/total-number-of-websites/, 2019. [Online; accessed 29-March-2019].

[45] P. D. Wasserman and T. Schwartz. Neural networks. ii. what are they and why is everybody so interested in them now? *IEEE Expert*, 3(1):10–15, Spring 1988.

[46] Lauren Wood, Arnaud Le Hors, Vidur Apparao, Steve Byrne, Mike Champion, Scott Isaacs, Ian Jacobs, Gavin Nicol, Jonathan Robie, Robert Sutor, et al. Document object model (dom) level 1 specification. *W3C recommendation*, 1, 1998.