Novay / University of Twente

Master's Thesis

Classification of Semantically Coherent Segments in Web Pages

Author:
Zwier Kanis

Graduation Committee:
dr. E.M.A.G. van Dijk
dr. C. Wartena
dr.ir. H.J.A. op den Akker

Human Media Interaction Group
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede
The Netherlands

UNIVERSITY OF TWENTE.

August 16, 2011

Contents

1 Introduction

The Internet is the cornerstone of the digital information age we currently find ourselves in. It made the world a smaller place by allowing people to exchange information at dazzling speeds. The wealth of information and different ways of communicating nowadays requires most people to be connected to the Internet. But the use of the Internet is not restricted to humans only. In a broad sense, the users of the Internet can be subdivided into humans and software agents, each consuming the information that is specifically designed for them.

More and more, the Internet brings forth the need to automatize the repetitive and tedious tasks normally performed by people. This means that software agents need to consume information designed specifically for humans. To efficiently present information from the Internet, it is interwoven with markup, layout and other types of information, before it is visually presented as a webpage to the user. Since this form of information is primarily destined for human readers, a problem is often introduced when it needs to be processed by software agents. Information in this case is defined as the data relevant to a user, or consumer of a webpage.

Three specific problems can be identified that prevent software systems from interpreting webpages on the Internet:

- Most of the information is unstructured, or at best semi-structured, and therefore often not directly interpretable by software. This prevents software from 'understanding' the information.
- Relevant information in webpages is interwoven with information about structure and design. In the case of webpages, this results in a document often written in (X)HTML, or (Extensible) Hypertext Markup Language. The practically infinite diversity of structure for webpages poses a big problem since the software interpreting it must anticipate every kind of structure possible.
- Webpages offer a lot of additional information alongside the relevant information the user is actually interested in (e.g., a navigation menu or widgets containing information like the weather). This makes it harder to retrieve only the information that is relevant to the user.

Identifying the different types of information in webpages can aid many different types of applications. Most prominent among these is the extraction of the main, central information on a webpage, for obvious reasons. Another application is that of restructuring webpages to a format suitable to be shown on different screen dimensions. This need is caused by the proliferation of devices with Internet browsing capabilities. Since developers of websites cannot anticipate the different devices the website will be shown on, the device itself needs to deal with its deviating resolution. Additionally it would take too much time for developers to take different devices into account by creating different layouts for a website.
Besides improving accessibility by adapting the layout, people coping with a visual disability can also benefit. To assist people without the ability to read information from webpages themselves, applications with structural knowledge about websites can assist. Visually impaired people use screen readers to acquire information from websites. For

them to efficiently browse the Internet, these screen readers must have a very precise idea about the structure and contents of the webpage.

Different proposals have been made to aid software in determining the meaning of information, among which the semantic web[3] is well known. The semantic web extends HTML with techniques that describe the information shown on a webpage, enabling software systems to also grasp the meaning of the information. But as long as the webpages on the Internet do not adhere to such an ideology, software stands on its own in figuring out the meaning of information contained in webpages.

The general approach to automatically extract information from webpages is currently to use wrappers; software tailored for specific webpages. Since the structure and design of a webpage is defined in the webpage source code, this information can be exploited. Of course, this way of automation inevitably leads to problems when the layout or structure of the webpage changes. Additionally it requires a great deal of manual labor, since the system needs to be specifically configured for each website.

The main aim of this project is to dehumanize the process of extracting and identifying the segments of information contained in webpages. Instead of tailoring software to specific webpages, we propose a general purpose method that will restrict the need for tailoring and might even be capable of extracting information from sources other than webpages, such as PDF files or digital news papers. To keep the method generally applicable and runnable on different devices, it is required that it performs its operations very fast, enabling real-time application. A general purpose algorithm can be applied in a wide range of situations, making it a valuable tool. Information can be extracted (e.g., for a specific information need or serving screen readers), modified (e.g., changes in layout or eliminating unwanted information), augmented (e.g., showing additional information or adding related advertisements), etc. The single constant factor of the information on webpages (and other sources that offer information to humans) is that it is structured in a way to allow human readers to efficiently discern and identify the different segments of information offered. Because most information will be made suitable for the human eye, we approach the problem of finding information segments from a visual perspective, using only visual aspects that are available to the human reader. Throughout this document, we will use the terms 'segment' and 'cluster' to denote a piece of information on a webpage that serves some particular function, which we define in chapter 5. The cluster term is specifically used to emphasize the fact that it is made up of smaller elements, a result from our automated approach we will elaborate on later in this document.

Given the magnitude of the project, it is subdivided into three, more or less separate parts that correspond to the main problems we want to tackle. The first task of the project is a user experiment. In this experiment we will examine the consistency of participants in discerning different segments of information on various webpages. Additionally the output of this experiment will be a collection of information segments for each webpage, which serves as a training and evaluation set for our automated methods of segmentation and identification. The second task of the project is to create a method that automatically discerns the different sections of information using only visual properties of a webpage. The webpages that were prepared during the user experiment phase of this project will serve as a collection of examples that will allow us to optimize our segmentation method. After splitting up the webpage in different segments, each of them needs to be identified. The third task is therefore to identify the different segments of information that were extracted in the second phase. To accomplish this, a classification model will be build

using the collection of examples from first phase.

To clearly outline the approach we will take, an overview is given in the following methodology chapter. Chapter three will give an overview of the state of the art work already done concerning the extraction of information from websites. In chapter four we will generalize these approaches and give an overview of the different sources of information available to our approach in dealing with webpages. Additionally this chapter will form a starting point for our approach. Chapter five provides a detailed description of the user experiment we conducted, while chapters six and seven cover the clustering and classification phases of our approach in more detail. We conclude this document with an overview of the results, discussion and conclusion in chapter 8,9 and 10, respectively.

2 Methodology

To make sure the approach we take in this project is clear, we will start by explaining the three subproblems, as mentioned in the introduction, in more detail. Starting with the user experiment.

2.1 User Experiment

The user experiment essentially is a data composing activity where participants segment various webpages and label the individual segments with categories we defined. We will further elaborate on the user experiment in chapter 5. The results from this process serve three different functions, which are listed below. Instead of segments, we will often use the term 'clusters', since the data is composed of multiple elements, which will become clear in the following chapters. The term 'reference clusters' is used to indicate the clusters that were composed by the human participants in this user experiment. These reference clusters will later serve as a basis for our evaluations and training sets.

Consistency of human classification It is very likely that there is a discrepancy between reference clusters when they are created by different participants. Either because the classification instructions are not correctly specified, or because some webpages contain clusters that are too ambiguous to be classified with our categories. To measure the agreement between different participants, we have the different human classifiers work on the same subset of webpages and measure the consistency between their resulting reference clusters and classifications. We reckon the results to depend heavily on the clarity of the classification instructions and the willingness of the participants. Both factors are to a large extent in our own hands.

Clustering performance measurement To be able to improve the performance of our clustering approach, we need to compare different clustering methods and their different configurations with each other. The reference clusters give us an indication of the performance that could be attained, which we must work towards. Additionally we will use these reference clusters to optimize the parameters used by our clustering method.

Classification training Next to clustering, we also need to classify the clusters based on their contents or properties. Either to support the clustering process or to indicate to the end user what content it most likely resembles. The reference clusters will serve as a set of training samples which will be used to create a model that embodies the relation between cluster properties and categories. This model can then be used as a classifier for the clusters generated by our clustering method.

2.2 Segmentation

After collecting a set of clusters with the user experiment, we can use these reference clusters to evaluate our method of extracting segments from webpages. The aim of segmentation is to extract various segments from the webpage in which the content of each

segment has some particular function to which all elements in the segment contribute, i.e., there must be a certain degree of coherence between the elements in a segment. Segmentation in this case roughly implies that we divide the webpage into a number of pieces, but we can also approach the problem from the other end with clustering, which is the process of finding elements that relate to each other according to some function they share. Segmentation and clustering in this case are two sides of the same coin and result in the same type of information. If we take the complete webpage as one single piece of information, segmentation can be viewed as the top down approach, used to divide it into different segments. On the other hand, if we represent a webpage by its elementary parts, clustering is the bottom up approach, used to merge the elements into segments, or clusters.

Earlier research with goals common to ours mostly undertook a top down approach, often using the internal webpage structure, which we will later explain in more detail, to segment a webpage and subsequently use a series of heuristics to look for particular segments. Consequently, using this structural information creates a webpage specific dependency and requires updates when the structure of a webpage changes.

In our approach we aim to depend only on visual information i.e., the information people can directly perceive in a structure that is specifically meant for them. This means we cannot use any language semantics or source dependent information for operations other than extracting the visual information we require. The main reason for this is that the method will be more robust and more likely to be portable to other types of input. We therefore prefer visual information over structural or semantic information. Visual information is made available to us through the DOM (Document Object Model, more about this in chapter 4) of the webpage in the form of text elements with properties describing their visual characteristics. Simply put, this DOM is a tree structure, where the elementary texts are located in the leaves. Note that this hierarchical structure of the DOM tree does not have to correspond to the visual appearance of the texts in the rendered webpage. This motivates us to start from the bottom up and merge these elements into groups with the help of the different visual properties of those elements.

In psychology, it is thought that principles of perception from the Gestalt theory[19] account for the ability to visually 'understand' wholes from groups of individual elements. This gives us a firm basis for our clustering methods and thus seems to be a suitable start for our approach. Although we as humans ultimately construct the wholes from the visual data we perceive, it should be mentioned that it is not necessarily the case that the visual properties solely account for our understanding of wholes. Semantic or meta knowledge about the elements we perceive also contribute to our ability of clustering visual information. An important question here is to what extend it is possible to determine the coherent sections on a webpage using only the individual elements with their visual properties, which will partly follow from the results we will attain using the clustering methods.

Based on these principles of perception, we will create a clustering method and attempt to have it generate clusters that resemble the reference clusters created during the user experiment. The reference clusters themselves and their webpages will serve as an evaluation set that also allows us to optimize the parameters used by our clustering method.

2.3 Classification

Clusters that were created by our cluster method still lack any indication of what type of content they contain. To get from a cluster to a category, a classification model needs to be build that will map a cluster, with the help of its properties, to one of the categories we defined. Identifying clusters only has a descriptive function, where we classify the cluster after the clustering process is finished. While it can also be used supportively, in assisting the clustering methods, we do not use it in this way.

Again we use the reference clusters created during the user experiment to serve as a set of examples. Using different features from the reference clusters, we will experiment with different classification methods to build a model that will best fit our reference cluster set. The model performing best will then be used to categorize the generated clusters.

3 Related Work

Most of our work involves the segmentation and identification of webpages. Research that includes these operations has been conducted in various fields, where the majority of this research operates directly on the DOM (Document Object Model). We will further elaborate on the DOM in the next chapter, but for now it will suffice to know that the DOM is a standardized representation of a webpage that offers access to all its properties. For each field we will now look at most relevant research.

3.1 Information Retrieval

A field that involves a lot of interaction with webpages on the Internet is Information Retrieval. Segmentation and identification can, for instance, be used to assist search engines with query expansion. For this purpose Yu et al.[24] proposed the Vision based Page Segmentation (VIPS) algorithm. This algorithm uses visual cues combined with the webpage DOM to create a hierarchical structure that reflects the visual representation of the webpage. Elements in this tree structure are visually separated from their siblings. Although this gives a good representation of the visual layout of the webpage, it is still necessary to find the coherent sections located somewhere in the tree. A threshold can be given that specifies a permitted degree of coherence, stopping the segmentation process at that level. The VIPS algorithm was initially developed to assist selection of terms for query expansion, but is now used in various other projects.

A system that specifically targets product information was built by Wu et al.[21]. Their system builds a DOM from a webpage and subsequently extracts chunks from this DOM. They then filter these chunks on the basis of several characteristics like spatial cues and features that are expected to be found in product information chunks. They also use a DoC (Degree of Coherence) value to determine when to stop filtering. The authors experimented with shopping websites and concluded that their product based algorithm outperformed VIPS, which most likely results from the fact that their algorithm is specially tailored to match the product blocks.

Mehta et al.[13] built a segmentation system based on VIPS, combined with text analysis to determine the (semantic) coherence threshold. A pre-trained naive bayes classifier is used to determine the number of topics in each segment. When no segments contain more than one topic, the segmentation process stops. The algorithm delivers a semantic structure of the webpage, indicating segments and topics.

To increase precision for document retrieval, Chibane and Doan[6] based their approach on topic analysis. Their segmentation algorithm uses visual properties (lines and colors) and structural tags (paragraphs and subtitles) to maximize a solution where the content within segments is coherent (measured by the relation between terms and a topic) and distances between segments are large.

3.2 Webpage Transformation

Techniques that aim to transform the layout of a webpage are becoming more common. The main cause for this is that various Internet browsing enabled devices are being created

with different screen resolutions (e.g., cellphones and PDAs (personal digital assistants)). Since it is awkward to browse the Internet on a small screen, researchers look for solutions by transforming webpages to fit the screen size. Another cause of research in this field is that content delivery is often expensive. By fragmenting webpages, data transmission can be kept to a minimum. Baluja[2] developed a system that tries to partition webpages into nine pieces, each containing one coherent piece of content, which can be used on cellular phones with WAP (Wireless Application Protocol) browsers. To come to nine correct pieces, a decision tree is created with the help of an entropy measure. This measure is biased by the size of the elements, since there should be nine, and the depth of the DOM node, since lower level DOM nodes usually divide a semantically uniform section.

For the purpose of browsing the Internet on small screens, Xiang et al.[22] developed a segmentation algorithm based on the webpage DOM structure and visual cues. They start by building a tag tree from the DOM. By looking for certain tags that cause a line break, they recursively merge all 'continuous' elements. The tag tree is then analyzed for patterns in tag sequences. After that, groups are formed by looking for patterns in the tag sequences. A weakness here is that the method is very dependent on the tags that need to be defined as line breaking and non line breaking.

Yang and Zhang[23] developed a method used for adaptive content delivery. They create a structured document, a hierarchical structure containing container objects. All elements from a webpage are clustered into container objects based on visual similarity and other custom defined rules. Clustering of elements here is based on the DBSCAN[8] clustering algorithm. Suffix trees are then built from the series of clusters and analyzed for patterns with the help of a list of heuristics.

Romero and Berger[16] worked on a segmentation algorithm to partition webpages into segments visible on small devices. Their method starts by building a DOM from a webpage. In a bottom up fashion, adjacent leaves are iteratively clustered together according to a cost function. In their work, this cost function is a combination of a few DOM distance measures that determine how far apart elements in the DOM tree are, which can be sufficient for certain webpages. It is however clear that this method is very dependent on DOM structure, and that complex webpages require different cost functions.

3.3 Screen Readers

People with a visual impairment that partake in webpage interaction are in need of tools that can present a webpage in a way they can perceive. A well known application in this area is the screen reader, which often is a type of text to speech application. To present a webpage in an effective way to the user, screen readers need to correctly interpret it by determining the meaning and relation of the various parts of the webpage. To simply process a webpage in a sequential order, would be very inefficient. For example in the case where a navigation menu is located at the bottom of a webpage and all other information is read, or processed, first when the user only wants to navigate. It is therefore of substantial importance that screen readers can identify the different parts of a webpage to be able to present them in an effective way. A developer of a webpage can support visually impaired people in different ways, for example by adding "talklets"[1] to the webpage to improve accessibility. A disadvantage of this approach is that visually

[1] http://www.textic.com/

impaired users are dependent on the website creator. The Firefox extension Fire Vox[2] is a popular screenreader for websites that is able to identify certain tags like images and links, which can already help users navigate more efficiently. A better approach however would be to create a screen reader that can interpret a webpage and identify the different segments that are presented. A user can then order the screenreader to immediately read the main text, or the navigation panel, saving the user a lot of time.

3.4 General

Gupta et al. [10] created an application that essentially tries to remove all the clutter from a webpage, leaving the actual contents to be processed. First a DOM structure is created from the webpage, which is then modified to extract the core contents of the webpage. Different filtering techniques are used that remove certain tags, attributes, advertisements, or other things from the DOM. The filtering algorithms used rely extensively on the contents of the DOM elements.

Song et al.[18] tried to derive a relation between properties of content blocks on webpages and the importance of those blocks. They first had five people classify over 4500 blocks of many pages on about four hundred different websites. Every block was classified with an importance level ranging one to four, respectively ranging from noisy information like advertisements to the main content or headlines. It seemed that the distinction between level two and three was not very clear, so for the experiments these were combined into a single level, leaving three levels. The experiment lead to the observation that people have consistent opinions about the importance of content blocks on webpages. For each block 42 different features were extracted, being spatial, absolute and relative to the webpage. In this system, the VIPS algorithm was used to extract the blocks from the webpage, how they dealt with the segmentation threshold was not mentioned, however. Learning algorithms like support vector machines and neural networks were used to create a model of the relation between the features and the importance level. The experiment showed that the performance of the classification algorithm came very close to that of the human classifiers. The websites chosen in this case all came from three sub websites from yahoo (news, science and shopping), which are likely to have a similar layout, making the results dependent on these particular websites.

Chen et al.[5] try to uncover the intention an author had towards certain parts of the webpage, by first transforming a webpage into an intermediate structure, the FOM (Function Object Model). A basic FOM is created by first retrieving basic objects from a webpage with their properties like decoration, navigation and interaction. A DBSCAN[8] like algorithm is used to cluster the basic objects on the webpage into composite objects. Following is the generation of a specific FOM. Here the objects are classified, determined by the properties of an object. For every category, a specific detection algorithm is needed. An example here is an algorithm that detects a navigation bar by using a list of rules, including the in- and out degree of hyperlinks in an object. *We are trying to do something similar, except we use other properties and employ machine learning techniques to categorize parts or the webpage.*

Work that comes very close to the idea brought forward in this document was undertaken by Snasel[17]. The idea behind their algorithm is about equivalent to ours (even Gestalt

[2]http://firevox.clcworld.net

principles are briefly mentioned) except that they take a very different approach with their application. Their work is based on pattern dictionaries, where each pattern describes proximity, similarity, continuity and closure of its elements. The algorithm then tries to extract segments that are similar to the defined patterns.

3.5 Conclusions

The work reviewed here contains a lot of elements that closely resemble parts of the project we are undertaking. Depending on the application, most systems concentrate on one type of information, most prominently being the main information section on a webpage. Some of the methods directly interpret the webpage, others transform a webpage into an intermediate structure enabling applications to find useful information by analyzing this structure. To achieve the various goals, the methods draw different sources of information from the webpages, such as structural and visual information.

The focus of our project will be to develop a general purpose method that is not restricted to the recognition of a single type of information. Additionally we will try to keep dependencies to a minimum by ignoring any information inherent to a specific source, such as structural information in webpages. This leaves only visual information for us to use, i.e., the information that is also available to the human perceiver. How we extract this information is explained in the next chapter.

4 Data Collection

Although we indicated to build a general purpose algorithm, initially extracting the required data is highly dependent on the information source. Given our aim to only use (rendered) information that is available for human sight, it would be ideal if a tool exists that will directly give us this data from analyzing any visual presentation of information. To our knowledge such a tool does not yet exist, so for now we will specifically focus on webpages and how we can extract the required data from them. In this chapter we will first look at the different types of data contained in webpages, followed by the extraction process and a description of the data that is collected.

4.1 Resources

The core information on a webpage, i.e., the information people are actually interested in, is mixed with presentational, descriptive, and sometimes, procedural markup to indicate to a browser how the information should be interpreted and presented. This markup is what provides structure to a HTML document. The core information itself is however often not well described or annotated, preventing computer systems from identifying and recovering the actual meaning of the data. Additionally, webpages are often augmented with extra pieces of information, often not very interesting for the perceiver.

The (presentational) markup reveals relations between the various pieces of information included in a webpage, which enables humans to quickly differentiate between them. Given the advantage we gain from the addition of the markup, we cannot simply remove it from a webpage to obtain the original information. We would end up with a heap of text and lose a lot of useful information. We will use the data that comes with the core information to our advantage to best be able to extract and identify the original information included in a webpage.

Globally, we can discern three different kinds of information in webpages, being: visual, structural and semantic. Visual information includes the elements that directly determine the visual appearance of the webpage. This includes color, spatial and font information. Structural information includes the markup and logical structure of the document. This is not directly visible in the rendered webpage, but does add functional meaning to the information in the webpage. An example of this is a paragraph tag that is used to group elements with some related function. Finally we have semantic information of the textual information on the webpage, where the actual meaning of the contents comes into play.

In spite of all the information readily available to us, there are some drawbacks when using these different types of information. We therefore prioritize the use of them according to these drawbacks, and aim to utilize only the preferred types. Following now is a detailed description of each of the three types of information.

Structural Information We already briefly mentioned the different types of markup that is added to webpages. Markup is included in the HTML document in the form of a nested tag structure which contains all the information. This structure represents the structural information and is reflected by the DOM (more about the DOM in section 4.2) that is generated by browsers, or other webpage interpreters. Using this structural information of a webpage comes down to analyzing the tag structure, or the DOM. Important

to note here is that the visual representation of the website does not have to correspond with the logical structure of the document, i.e., the location of an information segment in the markup structure, does not necessarily determine its location in the visual presentation of the webpage.

A problem with structural information is that the method relies directly on the code or tag structure of the webpage. Since the underlying structure of a webpage can be subject to change, it would require maintenance for a system using these tags. Additionally, analyzing the tag structure makes a method of information extraction useless in combination with other sources, such as PDF, since they most likely use another kind of internal representation. Another significant problem is that various structural tags can be interpreted in different ways, making it hard to find out their actual function in a webpage. An example is the table tag, which is often abused by developers to structure page layout instead of using it for the intended function of structuring information.

Visual Information Visual information is data that has a direct influence on our perception of the webpage and includes color, spatial and font information. While structural information may be directly available from the source code, acquiring visual information about the elements of a website requires an additional step in the process. This is because visual properties are often defined in scripts other than the immediate source code. An example is the use of CSS (Cascading Style Sheets) files. After a webpage is rendered by the browser, the visual data is made available through the DOM.

The relation between the visual representation of content and the function of that content is that viewers must be able to visually discern the different sections, including their functions on a webpage, in order to be effective in consuming the information that is shown. Visual information is the one consistent factor that is present in every information source meant to be consumed by humans.

Semantic Information Since we are mainly dealing with textual elements, meaning inherent to these texts may also prove to be useful in determining a relation between elements. The textual information itself is contained in the HTML code as well as in the DOM structure generated by the browser.

Semantic information relies heavily on the underlying language. Given the interlingual nature of the Internet, it can be expected that webpages in different languages need to be analyzed, which will be one of the biggest drawbacks for using semantic information in webpages.

When dealing with webpages, most of the methods in other work restrict themselves by relying on structural information. The most prominent drawback is that the methods are often tailored for specific webpages, which requires maintenance when the website structure is updated. Additionally, maintenance is needed if the language itself is revised. This is currently the case with the upcoming HTML5, which introduces a set of new markup tags. The main reason structural data is still used is that it often corresponds to the visual representation of the webpage and contains a lot of easily accessible data about the structure of the information. In our research, however, we want to focus on finding structure without being dependent on the underlying technologies of webpages. This means that we will try to only rely on visual information.

4.2 Data Extraction

Information on webpages is primarily made available through text. Although text can be embedded in images, flash or other objects, in webpages, text is still the predominant way to convey information to the user. Extracting texts from a webpage is straightforward as long as the texts are directly included in the webpage code. Extracting texts from embedded objects like images or flash is a task in itself, so during this project we restrict ourselves to only texts that are included in the HTML code. Texts included in objects other than HTML can in theory always be scanned and extracted with their visual properties, in order to use them in our method. From here on we will refer to a text with its specific properties as a webpage element.

4.2.1 Extracting Elements

For a software agent to interpret a webpage, it starts with a request to a specific URL (Uniform Resource Locator). The response to this request is, in the case of a webpage, the source code that represents that webpage. This source code often includes other scripts like CSS (Cascading Style Sheets) and JavaScript that are, among other things, used to generate a proper visual representation of the webpage.

The visual properties we aim to collect cannot simply be taken directly from the source code. To determine most visual properties, we have to practically render a webpage first. This rendering is not an obvious task, given that it is often done differently by different browsers. This is shown by the Acid[3] test for web browsers; an independent conformance test for web browsers, which is often differently rendered by different engines, resulting in different visual presentations of a single webpage. The most efficient way to obtain the information we need is to use an existing browser to render a webpage and then extract the elements. Among different rendering engines, we will use the open source Gecko rendering engine from Mozilla, a popular, well maintained engine also used by the Firefox browser, to do this rendering for us. Given the popularity of Firefox, we assume it is capable of correctly rendering most websites.

When a webpage is loaded in Gecko, it is transformed into a structure accessible through the Document Object Model (DOM). The DOM[4] is a standardized platform and language independent interface that is included in most popular browsers. Scripts can use this interface among all browsers that adhere to this specification to make adjustments to webpages. The structure of a DOM tree often reflects the visual structure of the webpage, which is why other methods that rely on this structural information can be very successful. This resemblance in structure is however no certainty, and when it differs, methods based on it will fail. It must be noted that well designed webpages do adhere to a logical code structure that reflects the visual structure of the webpage, since this is more easy to maintain. To connect to the gecko engine, the open source XPCOM (Cross Platform Component Object Model) technology is available. XPCOM is a language and platform independent framework also implemented by the mozilla browser.

Following the DOM tree built by the browser, the texts of a webpage are contained in special text nodes, which contain only text and do not contain child nodes or any style

[3] http://www.webstandards.org/
[4] http://www.w3.org/DOM/DOMTR

makeup. Often not all of these text nodes are made visible on a webpage. Two causes account for this. The first is that nodes can be hidden. For example in the case of a navigation menu, sub-menus are often hidden and made visible when a selection in the main menu is made. These hidden nodes can be filtered out by traversing to the top document node and checking for the visibility style with value "hidden", or the display style valued "none". Additionally, comments or alternative (ALT) descriptions can be included in the webpage, which are not visible to the user. The second cause for texts not being (fully) visible is that texts can be larger than the container fields they are shown in, resulting in a text being only partially shown. Since visual order and dimensions are most important when applying a method that only relies on visual characteristics, the properties of text elements we collect, corresponding to the visual information shown on the webpage, must be as accurate as possible.

Most style properties can be collected by processing the parent node of the text node. Some properties however require that we traverse to the root node in the DOM tree. An example of such a property is the background color, which can be transparent. In this case we would traverse further towards the root element until when we find a (non-transparent) color. Another example is the absolute position of an element. The absolute position of the element can be calculated by traversing from the text node to the root node of the document and simply adding all distances, since only the distance relative to the parent node is made available. When we calculate the dimensions of a text node, we need to keep in mind that a text can be larger than the container that embeds it. To find out the correct dimensions of the visually shown text, we traverse to the top document node and keep track of the smallest visual box dimensions, effectively resulting in the visible size of the text element.

Subsuming elements After we extract all text elements with their properties, one final step is needed before the data is ready to be clustered. This pre-clustering step is needed because of a side-effect resulting from the way the text nodes are collected. Every text element with a different markup is placed in a distinct text node of the DOM tree. If we have a large text with, in its center, a word that contains a hyperlink to another webpage, this link word is not contained in the large text node. By using the dimensions and positions of the text node, we can check for overlapping text nodes. Since the larger text subsumes the smaller word, they can be safely merged together into one single text. This pre-clustering step is necessary for our method and will merge only texts that already belong to each other.

4.2.2 Element properties

The following overview lists the properties extracted for every text node:

Text The text and the number of words it contains.

Font Size, family, style, weight, variant, text-decoration, text-transform and letter-spacing of the font.

Color The color of the text and the background color.

Element Size (width and height) and position (top and left) of the text element.

Other

There is one other visual property we need to correctly distinguish different segments from each other. Often there are situations where two distinct segments are visually separated by a background image. The properties we use from the text elements do not give us this kind of information and finding out if and how a background separates different segments on a webpage is a task in itself and not within the scope of this project. However, to successfully segment a webpage we do need to take these 'invisible' gaps into account. We therefore turn to structural information found in the DOM tree built from the webpage, and use information from this structure that corresponds to the visual gaps between elements. This structural distance measure can later be replaced with one that is purely visually oriented, relieving our method from this dependency. We can use this structural data, since each node in the DOM tree represents some part of the webpage and child nodes are often visually contained within the section of the parent node. Child nodes located lower in the tree often are semantically more similar, since they represent a smaller, more coherent, part of the webpage. A result of this is that it is possible to detect visual separators between elements by looking at the shortest distance between the two in the DOM tree. The assumption here is that distances between elements within that segment are relatively short compared to distances between elements from different segments, where distance is measured as the shortest path between two elements in the DOM tree. This shortest distance is calculated by first determining the LCA (Lowest Common Ancestor) for two elements and then adding the distance to this LCA from both elements. This LCA distance is determined for every combination of elements on the webpage and will be used to detect visual separators between elements.

4.3 Conclusion

Up to this point the main aspect we focus on is that we only use visual properties of the textual elements on a webpage. Consequently, at least in theory, this allows us to also use other information sources with our method of clustering and classification. The text elements with accompanying properties serve as a basis for the clustering and classification activities we will undertake in the following chapters.

5 User Experiment

To support our work on clustering and classification, we need to have a clear idea about the segments we want to identify on a webpage and what the contents of these segments are. The segmentation of webpages is a subjective undertaking, because it leads to a goal where we as the end user define the segment types that are important to us. Despite this, certain types of segments are present in most webpages, which we expect to be commonly recognizable by human perceivers. We therefore define a set of the common segments that will be used in the user experiment. For the sake of clarity, we will refer to these common segments as reference clusters.

A full description of the purpose of the user experiment can be found in section 2.1 of the methodology chapter. Note that the user experiment actually delivers two kinds of results. We gather both the data used for measuring inter-classifier agreement and the data used for training the clustering and classification methods. It is likely that some discrepancies are present in the agreement between participants, which we will then use to correct the deviations in the clusters used for training. We will now list the reference clusters we defined. Subsequently we will discuss the software we built to collect them, followed by the procedure we set up for the experiment.

5.1 Reference cluster specification

Our choice of segment categories is a trade-off between specificity and the hours human classifiers have to put in. We therefore only specify the most generic types of segments that can be found on most webpages and that actually do serve some useful function that software agents can use. We want the descriptions to provide enough information for the classifiers to unambiguously discriminate between all parts of the webpage. For the sake of the project, it is important that the participants correctly performed two tasks; selection of the segments, and their classification. A problem often seen with other clustering techniques, is that it is hard to determine the extend, or specificity, of clustering. We noticed in an earlier test that participants were struggling to determine the size of the clusters. In a few iterations with different participants we adjusted the specification, so that it became more clear what we were expecting. The participants were asked to select and label clusters with the following categories:

Main Text *A single, relatively large text with the main focus on a webpage.* This group includes the main, often large, text on a webpage with the aim to inform the user. Some webpages do not offer a main text at all (e.g., landing pages of portals like nu.nl or fok.nl). Other webpages offer multiple texts of different sizes (e.g., a news page displaying a main article and a list of contributions or reactions from users. Such contributions or reactions then do not belong to the main text, but to the Additional Text group described below). Keep in mind that the webpage is built around this main text and that most webpages feature only one main text revolving around a certain theme.

Additional Text *Informative and introductory texts, often relatively small, that do not have the main focus on a webpage.* Besides the Main Text, webpages often display a lot of other (smaller) texts, that serve as an informative text (e.g., comments) or as an introduction

for another webpage (containing a Main Text). Additional texts must contain at least a paragraph of text. Small text lines like a single title, questions in a poll, a copyright notice, or navigational links do not belong to this group. It can be hard to differentiate between the Main Text and Additional Texts. A webpage like geenstijl.nl displays a list of nearly fulltext articles. In these cases keep in mind that a Main text must be the main item on a webpage, which is not the case on the geenstijl.nl portal, so these texts can be classified as additional. Most webpages give an overview of articles in the form of introductions by displaying a short text, title and image. These introductions also can be labeled as Additional Text.

Link Group *A set of links pointing to other webpages, possibly within the same website.* A group of links can often easily be spotted. A link group often includes additional information like the time of creation or a number indicating the amount of clicks the link has received. All this additional information is part of the link group. To make things simple, a link group is simply a series of links that are visually grouped together.

Navigation Menu *A single, relatively large navigation menu often present on a webpage.* The navigation menu is the section that offers global website navigation, linking to different sections of the website. It is essentially a link group, but so prominently present that it receives special attention and is therefore labeled differently. Webpages usually include only one or two of these sections.

Advertisement *Texts that are indicated to be advertisements.* These sections are often clearly marked with terms like 'ads by ...' or 'advertisements' and often attract attention by visually standing out of the webpage. Their contents can be almost anything, but if it contains text and is selectable, mark this as an advertisement. Be careful not to include advertisement-like sections that for example show products that belong to the webpage in question, these belong to the Other Group label discussed below.

Other Group *A set of information that seems to belong together but cannot be classified with the former categories.* All things on the webpage that do not fall under the above mentioned type definitions but do seem to form a coherent whole, can be grouped together and labeled Other Group.

Remaining (No Label) All remaining elements can be left as they are, and do not need to be labeled.

5.2 Software

To support participants with their clustering and classification tasks, we built a Java based application. The webpages that need to be processed by the participants are already downloaded and made selectable in a list. For every webpage the source code, all text elements with their properties, as mentioned in 4.2.2, and a screenshot is stored. This screenshot is shown instead of rendering the webpage from source code, since sometimes

external scripts are loaded that change the webpage layout. When this happens, our stored elements and their details could deviate from those shown in the webpage. When selecting one of the webpages, the corresponding webpage is loaded in a new window where different view modes are present:

1. Original; shows the clean page view, where each element is selectable to check its text contents.
2. Reference Clusters; the view where participants can manage the segments.
3. Clusters; shows the clusters that are generated by our method, mainly used for debugging.
4. Grid; shows the webpage with a grid of the elements drawn in a lay-over screen, mainly used for debugging.

The participants mainly operate in the reference clusters view. An addition toggle option is made available here that indicates the still unclustered elements. Figure 1 gives an impression of the clustering operation. the red fields are text elements that are not yet part of a cluster. A cluster can be created by simply dragging the mouse over a portion of the screen and will be indicated by a green semi transparent area that will automatically snap to the text elements it contains immediately after releasing the mouse. Clusters can be removed by performing the double-click operation on them.

Figure 1: Clustering operation

By using the right mouse button, participants can attach a label to a cluster. Consequently the cluster will get a category specific semi-transparent color, which makes it easier for participants to verify the cluster categories, which can be seen in figure 2. Additionally the label of the category is shown in the top left corner of the cluster. Right clicking on a cluster also allows participants to change or remove a label.

Figure 2: Classification operation

To prevent unnecessary work in the unlikely event of a software crash, each user action is immediately stored to disk. This also allowed participants to stop and resume their work at any time.

5.3 Procedure

To collect the reference clusters, the following procedure is followed for each participant:

1. Explain the project and the role the participant has in it
2. Discuss the categories from the reference cluster specification with the participant
3. Explain the interface and operation of the reference cluster software
4. Have the participant cluster an initial webpage to test his or her understanding of the interface and task
5. Have the participant create the reference clusters from a set of prepared (already downloaded) webpages

The software is deployed as a Java executable, so participants are not forced to work on a computer other than their own. The software is accompanied by a specification document containing the reference cluster descriptions, as outlined in this chapter. The participants can take as much time as they need, as long as the clustering and classification choices are made by themselves and no help from people other than the participant is involved.

A total of 39 websites are used to create the reference clusters (for a complete listing, see appendix A), which mainly consist of news portals. From each of these websites two webpages are stored, the homepage of the website (overview page) and a page of a news or content item (detail page), resulting in 78 different webpages. These overview and detail pages are spread equally among the participants, so that each participant has the same number of overview and detail pages, but never of the same website. The only exception here is that websites used for measuring inter-classifier consistency are present in the form of both overview and detail pages for each participant.

To be able to measure consistency of clustering and classification between participants, an overlap of webpages is necessary. We configure this overlap to be ten webpages, which comes down to about 8% of the total number of pages. i.e., we expect about 8% of the clusters to be clustered and classified by two participants.

After finishing their tasks on all webpages in the list, the participants would hand in their work, which forms the basis for the actual clustering and classification methods we will discuss in the following chapters. The results from this user study can be found in section 8.2.

6 Clustering

The restriction of using only visual information to segment a webpage becomes most concrete in our method of clustering. The purpose of clustering is to combine elements on a webpage that visually appear to belong together. This is a somewhat subjective description, making it hard to translate directly into an algorithm. What we are essentially trying to do, is mimic the process of human perception. Humans determine coherence between elements with the help of visual information like distance and color. But also by analyzing semantics of the text contained in the elements. In our clustering attempt, we assume that segments of a webpage with different types of content can be distinguished only by using visual information, i.e., using only visual characteristics of the elements of a webpage, we deem it possible to cluster them in a proper way so that only one type of information is contained in a cluster. The information types of the clusters correspond to the categories we defined in section 5.1.

The assumption that parts of a webpage with different types of content are visually discernible from each other stems from the fact that designers adhere to specific design guidelines, consciously or not, to present information in an efficient manner for people to perceive. A webpage that was not designed according to these guidelines will form a challenge when consuming information, not only for software, but also for human viewers. The most basic of these guidelines are described as the Gestalt laws of perceptual organization[20], which we will apply with our clustering method.

We will discuss two clustering algorithms that will effectively do most of the clustering, where the reference clusters from the user experiment will serve as a set of examples used to tune their parameters. In this chapter we will first outline the basis of the perception principles: the Gestalt laws, followed by the two algorithms of clustering we developed to apply them. Afterwards, we will discuss two cluster strategies, which combine the two clustering algorithms in different ways.

6.1 Perception Theory

When we extract the different pieces of information with their visual properties from a webpage and try to recover the relations between these elements, we are essentially carrying out work equivalent to that of the designer of the webpage. The designer also takes the different pieces of information and combines them into wholes, which ultimately results in a single webpage. To create a consistent and effective design, designers adhere to specific design guidelines. These design guidelines are themselves often based on the low-level laws of perceptual organization from the Gestalt psychology. Basically, this set of principles indicates how humans perceive patterns by grouping visual elements. These principles thus play a big part in visual systems like webpages, otherwise a viewer of the webpage would never get a grip on the information shown.

Segmenting a website and determining the coherent elements that form a cluster with a single semantic function requires an algorithm that has a human-like conception of what constitutes these clusters. The Gestalt psychology offers a starting point to get from the perception of single elements to clusters of elements based on their spatial geometrical organization and other visual properties, which are outlined in section 4.2.2. Gestalt psychology deals with the phenomenon that wholes are more, or at least different, than

simply the sum of their parts. Within Gestalt psychology, laws are formulated that describe perceptual organization[20]. In our case this comes down to the perception of clusters made up of a number of elements. The set of laws of perceptual organization include the following:

- Law of Proximity - Objects appearing closer to each other are more likely to be perceived as a group.
- Law of Similarity - Objects similar to each other are more likely to be perceived as a group.
- Law of Closure - Objects that form closed shapes are more likely to be perceived as a group
- Law of Continuity - Objects that minimize change or discontinuity as a group are more likely to be perceived as belonging to each other.
- Law of Symmetry - Objects that appear symmetrical are more likely to be perceived as a group.
- Law of Common Fate - Objects that for example share a common moving direction are more likely to be perceived as a group.

Not all of these laws can be utilized for our purposes. They are either too complicated to detect or not suitable to use with the data we have. In our case the most practical laws are those of proximity and similarity, which translate to the positional and visual aspects of the elements on a webpage. The laws of closure, continuity and symmetry correspond to the structure and alignment of elements on a webpage and can also contribute to the clustering process. The remaining law, common fate, is less straightforward and much harder to detect, so we will leave this aside in our methods of clustering.

Some research has been done on the organization of the laws. Quinlan and Wilton[14] tried to find a relation between the laws of proximity and similarity. Since the laws themselves are not organized, the researchers tried to find out how the two laws interact with each other. Results were that this is very subjective, varying per person. If objects with close proximity and different colors feature a conflict between the two different laws, no law seemed to get a significant priority above the other. In that same research, Quinlan and Wilton also found that people do prioritize the laws consistently for themselves, always favoring either proximity or similarity.

The results of Quinlan and Wilton's study show that even with well defined laws of perception, conflicts can still occur and it is up to the perceiver to sort it out. Since this type of conflict between proximity and similarity would be a bad choice in presenting information, a webpage would have to have a really bad designer for these visual contradictions or conflicts to occur and thus present information incorrectly for certain users. Although there is no direct organization of the Gestalt laws, we do have to take note of possible conflicts when we prioritize them in our clustering method, since we are using multiple laws, and it can have an effect on the performance of clustering.

Application of the laws

For a designer, the laws of perception are tools for creating a layout that is understandable for other people. A designer always makes use of these laws, consciously or not, to transform a set of information or functionalities to a structured design. We try to do something similar. With the laws of perception and the elements with their visual properties, we use the design to uncover the original structure of information. Following now is an overview of the different laws of perception and how they relate to the data available to us.

Law of proximity The law of proximity tells us that objects located closer to each other are more likely to form a coherent group. In this case, when two elements are closer to each other, their functional relation tends to become stronger. This law only deals with one property, distance. Distance between elements can however be measured in several ways. The most obvious option is to take the Euclidian distance between the centers of elements. However, this can yield an ambiguous representation. When for example two large elements are located next to each other, the distance between their centers can be substantial, while visually the distance between the two elements is negligible. A better option will therefore be to take the shortest Euclidian distance between the edges of two elements. This gives a better representation of what is visually perceived and is thus a better measure to indicate the proximity between elements.

Distances between elements can differ significantly per given segment for different webpages, making it likely that the use of a threshold or absolute value to cluster elements based on distance will perform bad in some cases. In addition to just the distance relation of two elements, the webpage in question, or context, thus also plays a part in determining the strength of the relation.

Law of similarity The law of similarity is harder to translate compared to the law of proximity because a lot of properties lend themselves for measuring similarity, whereas in the case of the proximity law we had only one property to deal with. Specifically, we can use all the font and color properties from section 4.2.2 for this similarity measure.

When we try to determine the strength of a relation between elements, these properties each contribute differently in different webpages. We must evaluate the contribution these properties make in light of their context, just as in the case of the law of proximity. However, we now have to deal with a whole set of properties. This begs the question if these properties interact with each other in determining the relation between elements. For now we assume these visual properties do not interact with each other and evaluate them independently.

Comparing elements with the help of these properties can be done in many ways. We do not differentiate between properties and simply compare them all, resulting in the number of deviations of properties between the elements, where all differences have an equal weight. This can be seen as a form of Hamming distance[11] of the properties of the elements, where this distance determines the strength of the relation between elements. From hereon we will refer to this distance as the similarity distance. As for the threshold of this distance, we have to deal with the same problem as discussed in the previous paragraph; that context might be taken into account to determine a suitable threshold.

Law of closure, continuity and symmetry The laws of closure, continuity and symmetry are concerned with form and structure of multiple elements rather than properties of two elements. In contrast to the other laws, these laws inherently take the context of the elements into account and can be said to operate on a higher level. Translated to the problem of clustering elements, these laws are related in a way that they all prefer consistency in forms. The most useful property for these laws is the location and size of the relevant elements. The law of closure favors elements that fill up gaps or complete structures. The law of continuity favors elements that continue some trend within a structure, thus punishing elements that deviate from this trend. The law of symmetry favors elements that mirror others around the center of the complete structure.

Capturing these laws in logical functions can be a valuable addition to our clustering method and will make the clustering process more robust.

6.2 Initial Structure

Now that we have some idea about the laws of perception and how they can be used to relate elements on webpages, we will use them in the process of clustering. Webpages sometimes contain hundreds of elements, resulting in a computationally very expensive operation if every combination of elements needs to be evaluated. This can be brought down by only evaluating the relations that matter, which we will determine with the help of the laws of perception. We will create a representation of the webpage that allows the clustering algorithms to work more efficiently, which we will refer to as the relationgrid.

Since the law of proximity contains less ambiguity than the law of similarity, we will first focus on clustering elements based on the distance between them. The law of similarity will be mostly used as a filtering function, removing relations between elements that are not similar enough. The laws of closure, continuity and symmetry are used later on in the process, since these operate on the context of multiple elements, which is represented by our relation grid.

The law of proximity tells us that elements positioned relatively far from each other are less likely to be related (other things being equal). It is also a given that a cluster on a webpage is always a group of elements located directly near each other. As a simplification, we therefore only take the direct upper, lower, left and right neighbors of elements into account during the clustering task. The assumption here is that segments are always presented in a rectangular fashion. Subsequently this is also more efficient, since only the direct neighboring elements need to be evaluated.

The relationgrid is thus a graph structure where for each element the closest horizontal and vertical elements are listed. Vertices and edges in this graph represent the elements and relations, respectively. After the relation grid is created, all non reciprocal edges are filtered, resulting in a undirected graph. This filtering is done to prevent situations where multiple elements point to a single side of another element, which will make clustering unnecessarily complex. This situation is portrayed in figure 3.

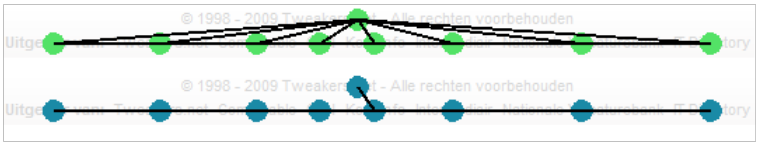

Figure 3: The upper grid is filtered, resulting in the lower grid with only reciprocal connections.

We do not take similarity into account in this phase, so a webpage is represented by only one possible grid. If we were to take similarity between elements into account during the initial build of the grid, a webpage could contain multiple intertwined grids, depending on the similarity of the elements. Figure 4 illustrates this full relationgrid without similarity filtering. Using similarity to value the relation between elements is left up to the clustering methods discussed next.

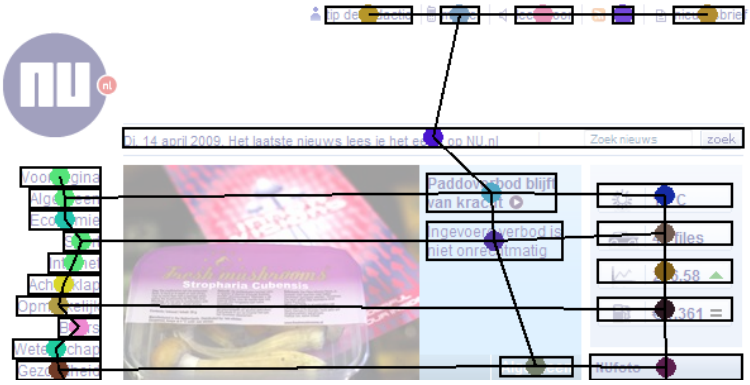

Figure 4: A full relation grid, without similarity filtering.

6.3 Algorithms

With the help of the relationgrid we can apply our clustering algorithms. Note that the relationgrid is only a structure that supports faster processing, since the clustering algorithms can, in theory, also operate directly on the DOM tree of webpages. This would however make the algorithms dependent on the DOM tree, which is something we want to avoid. We devised two algorithms that look for different types of structures; block clustering and pattern clustering. Block clustering focuses on rectangular structures within the relationgrid, while pattern clustering looks for random structures that repeat themselves. Both algorithms can be applied iteratively, where clusters are treated as elements in a following iteration. Before we go into detail for each of the algorithms, we first discuss how elements are merged together to form a cluster, an operation often performed by the clustering algorithms.

Aggregating elements

A cluster is essentially a pack of one or more elements. The properties of a cluster arise from the aggregated properties of its elements, which is not a trivial process. Properties like location and size are fixed and easy to calculate. Font properties and background color are however a different matter. We want the cluster properties to give a correct description of the visual characteristics of the cluster, i.e., we take the properties that cover most of the surface of the cluster, since these will have the most influence on the

visual perception of that cluster. One last property is harder to determine, which is the LCA distance. Following the same approach as with the other properties, we take the LCA distance from the largest element in the cluster. To support multiple iterations of clustering, clusters will have the same set of properties an element has. An element thus is equivalent to a cluster containing that single element. When multiple iterations of clustering take place, gradual differentiation of properties can occur if we keep using the aggregated cluster properties. We therefore determine the aggregated cluster properties using only the initial, basic elements present in that cluster.

6.3.1 Block Clustering

Most data on webpages is presented in a rectangular structure. Especially data within a segment is often organized in a clear tabular form, with rows and columns. With the help of some of the laws of perception, we can use the relationgrid and look for rectangular structures that seem to form a coherent whole. The law of closure requires that every element in the rectangular structure is fully connected. This means that the width of each row equals the width of the cluster and the height of each column is equal to the height of the entire structure. A fully connected (sub)grid thus has no gaps or visual outliers. The law of continuity also plays a role here. If elements are missing in this rectangular sub-grid, this would mean that continuity within the structure is broken, which makes it less likely that the structure forms one coherent whole. Figure 5 shows an example of a relationgrid and the clusters that are found when looking for block structures.

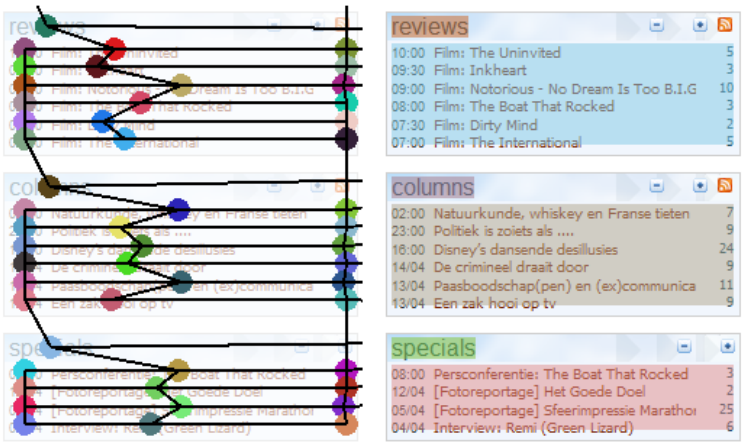

Figure 5: The left side shows the relation grid. The right side the corresponding clusters.

After a rectangular structure has been found in the relationgrid, we want to make sure this structure does not contain multiple content types. We therefore check if the structure satisfies certain conditions with the help of the similarity distance. Most rectangular structures contain repetitive sub patterns that have a similar visual appearance. Because of this, we cannot simply use the similarity distance threshold between all elements. However, to present the data in a correct visual format, the rows and columns are consistent in their difference of similarity. We therefore use the similarity distance to check the consistency of this difference in similarity. Figure 6 portrays this situation, where each black arrow points to a set of connections that are compared with each other.

This strategy of simply taking the elements from the largest rectangular structure as a cluster introduces a problem when two similar structures are positioned next to each other

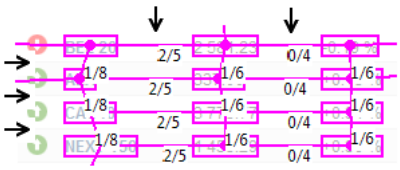

Figure 6: Comparing similarity of rows and columns.
The numbers represent: similarity distance / LCA distance between elements.

and seen as one single structure. The reason this situation can occur is that visually these two distinct structures are separated by whitespace, or by a background image that clearly separates the two (Figure 7, 8 and 9). To detect this kind of gap between elements, we turn to the LCA distance already introduced in section 4.2.2. By using this data however, a new problem is introduced. Namely that we now need to determine a threshold value that indicates an actual gap between elements. This is related to the problem of finding such a threshold for the similarity distance mentioned earlier. For now we simply use a static threshold to look for gaps, which will be determined with the help of the reference clusters.

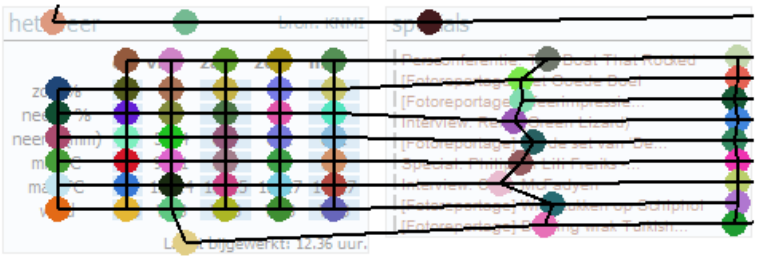

Figure 7: Part of the relationgrid of a webpage.

het weer — bron: KNMI

	do	vr	za	zo	ma
zon %	40	10	20	20	40
neersl %	70	90	40	30	30
neersl (mm)	8	5-14	0-3	0-2	0-2
min °C	10	9/11	5/8	5/8	5/7
max °C	19	12/14	13/15	13/17	14/17
wind	O 3	W 3	W 3	W 3	W 3

Laatst bijgewerkt: 12.36 uur.

specials

Persconferentie: The Boat That Rocked	3
[Fotoreportage] Het Goede Doel	2
[Fotoreportage] Sfeerimpressie...	25
Interview: Remi (Green Lizard)	6
[Fotoreportage] Op de set van 'De...	15
Special: Philip en Lili Freriks -...	3
Interview: Cody McFadyen	1
[Fotoreportage] Wrakstukken op Schiphol	3
[Fotoreportage] Berging wrak Turkish...	12

Figure 8: Clustering, not taking the LCA distance between elements into account.

het weer — bron: KNMI

	do	vr	za	zo	ma
zon %	40	10	20	20	40
neersl %	70	90	40	30	30
neersl (mm)	8	5-14	0-3	0-2	0-2
min °C	10	9/11	5/8	5/8	5/7
max °C	19	12/14	13/15	13/17	14/17
wind	O 3	W 3	W 3	W 3	W 3

Laatst bijgewerkt: 12.36 uur.

specials

Persconferentie: The Boat That Rocked	3
[Fotoreportage] Het Goede Doel	2
[Fotoreportage] Sfeerimpressie...	25
Interview: Remi (Green Lizard)	6
[Fotoreportage] Op de set van 'De...	15
Special: Philip en Lili Freriks -...	3
Interview: Cody McFadyen	1
[Fotoreportage] Wrakstukken op Schiphol	3
[Fotoreportage] Berging wrak Turkish...	12

Figure 9: Clustering, taking the LCA distance between elements into account.

The result of applying the similarity and LCA distance filter to the rectangular pattern will result in a single cluster. Another situation that causes problems is that of single large

columns or rows that, according to the clustering function, have a stronger relation than small rectangular clusters. This is illustrated by figure 10. Two solutions can help solve this problem. The first is that we can give a cluster with a rectangular shape priority over a single line cluster. This is however a restriction that might not always give us the best results, because it would always rule out one of the two shapes. The other option is that we look at structure within the cluster, following the laws of continuity and closure. If we take another look at the example in figure 10, we can see that there is a discontinuity on the fifth row, next to the word 'productreviews'. This kind of discontinuity is detected by explicitly looking for a deviating distance in either the columns or rows. In this case of a single row or column, we cannot apply the similarity distance as we did before, since there are no multiple columns and rows. We therefore use the similarity distance as a threshold for all elements in the row or column. The LCA distance was also used this way, just as with the multi column and row structures. One minor additional condition is that we require all LCA distances between the neighboring elements in a single row or column to be the same. This can be explained by the fact that single row or column patterns are practically always lists that contain the same elements and have a consistent visual distance.

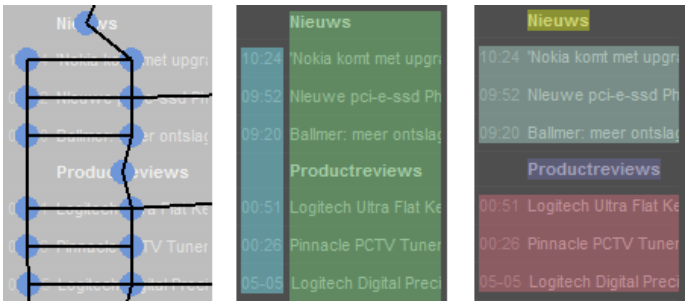

Figure 10: On the left is the grid, the middle shows the problem of selecting the largest block, and the right side is how we would expect it to be.

This clustering based on rectangles continues till all rectangles of a minimum size are extracted from the relationgrid. In most cases single elements remain that are not contained in clusters and left in the relationgrid. These remaining elements are now listed as single element clusters, so that after a clustering iteration only clusters remain. The most important thing at this point is that no clusters are formed that contain elements that do not belong together. Evaluating this comes down to making sure that the precision of elements in clusters is close to one, which is explained in more detail in section 8.1. This process of clustering can be repeated until no new clusters are formed. Note that in each following iteration of clustering, less clusters are available, resulting in less visual structure that can be exploited.

Algorithm description

We start by checking for each element if it is part of a cycle (we register every corner it is part of). A cycle in this case means that four elements are directly connected to each other in a circular fashion. This is used in the cluster algorithm to speed up the check for fully connected rectangles (rectangles where each element is part of such a cycle). The grid in figure 11 contains two cycles, indicated by the two black rectangles, which, as a whole, form a fully connected rectangle.

Figure 11: Cycles in the relationgrid, indicated by black rectangles.

In one iteration, the algorithm extracts all sets of elements that satisfy the given conditions (similarity & LCA distance and minimum cluster size). Only after the iteration is completed we combine all groups of found elements into clusters and create a new relationgrid that can be used in a next iteration. All elements that were not part of a set of elements are untouched and placed in a new single element cluster.

1. For each still unused element, we check what the biggest structure is that can be formed with the unused element at the top left. We continue this until no structure can be found that satisfies the given conditions.

2. Finding the biggest structure is done as follows:

   - Starting with the unused element at the top left, we first determine how much we can move to the right and to the bottom in the relationgrid, respectively the maxwidth and maxheight.
   - We then start with the maxwidth, and look for the width at height + 1 (going down) by following the cycles. If the width is less then maxwidth, this becomes the new maxwidth. This is continued until we reach the maxheight. Each time we move to a new width, we essentially have a sub-structure that could be the biggest structure. This structure however needs to be checked for fragmentation, which is done with the help of the similarity and LCA distances.
   - Searching for the biggest fragmented structure goes as follows: from the starting element, we simply go towards the maxwidth and at each step we check if the LCA and similarity distances are within the threshold range. In case this accepted range is broken, we register that width and after we vertically do the same with the height, we return the new dimensions. In this case, as explained before, the similarity between rows and columns can be very high, since columns are often displayed differently. Because of this we do not directly compare the similarity between elements, but the differences in similarity between elements. This measure gives a good impression of visual deviations.
     In the case of a structure with a single row or column, an extra condition is used to make sure all LCA distances are equal between neighboring elements.
   - At this point we keep track of the width and height with the largest number of elements. After evaluating this biggest structure for all elements in the grid, we return the width and height of the biggest block structure.

To optimize the block clustering algorithm, we can tweak its three parameters: similarity distance, LCA distance and the minimum cluster size. The reference clusters will be used to find values for the three parameters, so that the generated clusters resemble the reference clusters as close as possible.

6.3.2 Pattern Clustering

Rectangular structures are the most obvious patterns to detect in webpages, but that leaves a lot of coherent structures that do not allow detection that easily. A way to detect these coherent non-block structures is to look for repetition of similar structures. In a way this corresponds to the Gestalt law of similarity, only on a more global level. If multiple, repetitive, equivalent structures of equal shape can be found on a webpage, they most likely serve some function that is equivalent. Consequently, the elements from these structures can be clustered together. 12 shows an example of such re-occurring patterns on a webpage.

Figure 12: An example of a repetitive pattern on a webpage.

Our starting point, again, is the relationgrid, but because no other pattern matching algorithms are available that work with such a structure, we implemented a naive algorithm that allows us to find repetitive patterns on a webpage. It quickly became apparent that a lot of heuristics (restrictions in this case) were necessary to lessen the work and be able to look for patterns in an acceptable time. Very important here is that the rectangular structures, mentioned before, are already clustered. These rectangular patterns contain a lot of repetitive structure and by already clustering this, it relieves the pattern finding algorithm of a lot of work. Besides that, different constraints, such as a LCA distance threshold, similarity distance threshold and a limit to the clustersize are used to prune less favorable results during the search.

With this algorithm, we are searching for sets of disconnected equivalent patterns. To do this, we start with a 'base cluster' by taking a single element from the relationgrid and list all other structures that are similar to that base cluster. We then expand this base cluster according to a traversal strategy and update the compatible structures after each step. A scoring function keeps track of the highest scoring set of clusters. We repeat this procedure for each element in the relationgrid to make sure we considered all possible base clusters. The similarity distance measure is not used here to check if elements belong together, since this is less relevant on the more global level this method operates on. Instead, we use the similarity distance to check if new elements added to the

clusters are alike, keeping all structures equivalent. The LCA distance measures serves two functions here. First it is used to check if a new element can be added to the base cluster, i.e., if the element is 'visually' close enough to the base cluster. Second it is used to check if new elements attached to the remaining structures (after a traversal step) share the same distance compared to the new element attached to the base cluster.

For large websites with many elements, this naive approach has some serious performance implications, so we need to find these patterns more efficiently. The performance problem lies in the fact that different clusters can conflict with each other, creating the need to explore multiple possibilities, causing exponential growth in the number of cluster sets that need evaluation. Our target however, is not to evaluate every possible combination of clusters, but to find the largest non-conflicting set of clusters (depending on the score function). When we combine this with the fact that almost all webpages contain isolated subsets of conflicting, i.e., clusters that do not conflict with clusters from other subsets, we can optimize the algorithm. We first separate all subsets of conflicting clusters and for each of these independent subsets we determine the largest set of non-conflicting clusters. After that we simply add those sets together. In a worst case scenario this approach still suffers from the upper bound of $O(2^n)$, but in practice we will never see such a case.

Algorithm description

Just as with the block structure algorithm, the pattern clustering algorithm operates on a relationgrid of the webpage and it works as follows:

1. For each element in the relationgrid, we start with that element and put it in a cluster referred to as the base cluster.

2. We add all other elements of the relation grid as separate clusters in a clusterset, and remove those clusters that contain elements that are not similar enough to the one contained in the base cluster, using the similarity distance. At this point we thus have a base cluster and all the other clusters that are similar to the base cluster.

3. Following a depth first strategy we add connected elements to the base pattern, if they respect our maximum LCA distance restriction and are not already contained in the base pattern. At each step we continue with the best scoring pattern. We continue until only the base pattern is left in the set of clusters, which must occur at some time. With each addition to the base cluster we must update all the other clusters that are still present, which is done as follows:

    - For each cluster that remains other than the base cluster, we try to find an element that corresponds to the one last added to the base cluster. The direction (up, down, left or right) in the relationgrid and attachment (what element of the base cluster the new element is attached to) must correspond. Additionally the similarity of the elements (comparing the new item in the base pattern to the new item in the current cluster) and the LCA distance (between the new element and the pattern) must be within the allowed range.
    - If a new element is already present in another cluster, a conflict arises. Despite this possible conflict, we will still add the element and resolve the problem in the next step.

- After all clusters are updated (or removed) and thus equal to the base clusters, we create independent sets of conflicting clusters. For each of these sets we then list the largest conflict-free combination of clusters.
- To determine the score of the current pattern we combine all largest conflict-free combinations of clusters into one set, and return the score depending on the scoring function.

4. For each addition we keep track of the pattern that achieved the highest score according to a scoring function. When the algorithm ends we return the best performing set of clusters.

The following parameters are used to influence the clustering process.

- The traversal strategy. Currently being depth first search, determines how to extend the best cluster at that moment.
- The maximum number of elements in the base pattern, to restrict the size of the pattern. We set this to unlimited.
- Maximum allowed LCA Distance between elements in patterns.
- Maximum allowed similarity distance between corresponding elements in all clusters.
- Scoring function. This function determines what types of clusters we prefer:
  - If we sum the square of the number of elements in each cluster, fewer bigger clusters are preferred to multiple smaller clusters.
  - If we sum the total number of elements in all clusters, no specific preference is set and we simply prefer the pattern that clusters the most elements together, which is what we want to achieve.
  - If we sum the total number of clusters, this results in smaller clusters.

To optimize the pattern clustering algorithm, we focus on the similarity distance and LCA distance parameters. Again, the reference clusters will be used to find values for the these parameters, so that the generated clusters resemble the reference clusters as close as possible.

6.4 Strategy

The two cluster methods discussed in the previous subsection can help us determine which webpage elements belong together. It is however not yet obvious in what order we should apply them and what values we must use for their parameters, so we need to put together a strategy that describes an application of the clustering algorithms to obtain the best possible performance. To do this, we evaluate two approaches to clustering; greedy and conservative, where we tune the performance by running all possible combinations of parameter values for the cluster algorithms. The set of reference clusters will help us to determine the performance of the strategy. The results of these evaluations can be found in section 8.3. To determine the optimal strategy, there are two things we have to keep in mind:

- We start with block clustering to make sure no excessive repetitive structures remain that would prevent our pattern clustering technique from finishing in a practical time period.
- We must always try and keep the precision at the highest level. When the precision gets lower, this means that we clustered too much elements together and since we do not have any segmentation algorithm that splits up clusters, we cannot correct such errors.

Greedy Approach

With the greedy approach we start by optimizing the parameters for each cluster method individually. When a clustering algorithm is run with a specific configuration, we keep clustering with these settings until no more clusters are formed, which we will refer to as re-clustering. After running the initially required block clustering iteration to make sure pattern clustering can be performed efficiently, we keep adding one of the two cluster algorithms to the strategy, depending on which performs best. We keep doing this until there is no more improvement in performance. The result will be a strategy that includes an execution order of (re-clustering) algorithms, most likely with a different configuration for each of the algorithms.

Conservative Approach

Instead of optimizing individually, we can also choose to optimize combinations of the cluster algorithms. In this case, we do not perform re-clustering, but execute a single iteration of clustering. To have the initial block clustering algorithm behave in a more conservative way, the precision measure can be used. By putting the focus on precision, less clustering will occur, which translates to a more conservative clustering algorithm. This is necessary, because we do not have a clustering algorithm that re-evaluates clusters and breaks them up. To measure the performance of clustering, we use the F_β-score, which combines precision and recall into a single measure (more about this can be found in section 8.1. Because this is a weighted function, it is possible to put more emphasis on precision, by using the $F_{0.25}$ measure. Since the pattern clustering algorithm is too inefficient to start with, we need the block clustering algorithm to cluster just enough of the structures to allow the pattern cluster algorithm to run efficiently within an acceptable time. After this we add either another iteration of block or pattern clustering, depending on which performs best and keep doing this until the performance stops increasing.

The clustering strategy can be applied to a webpage to obtain a set of clusters. This, however, does not give any indication of the contents of the clusters. In the next chapter we will discuss our approach to finding the most likely content type for the generated clusters. The results of the clustering process discussed in this chapter can be found in section 8.3.

7 Classification

To discover what type of content is most likely contained in the clusters that we produced in the clustering phase, we need a model that captures the relation between the properties, or features, of the cluster and the type of content it contains. This type of content corresponds to the categories we defined in section 5.1. We use the term features to describe properties of a cluster that are actually used in the mapping with the category, while the properties themselves are the elementary properties of a cluster, not necessarily directly linked to the category. Not all properties are useful in this mapping, so we only select features that are likely to make a positive difference. Since it is not feasible to manually device a mapping ourselves, we use the reference clusters to create such a model. The reference clusters are already classified with the categories, so these will serve as a training set to create a classification model with the help of supervised machine learning techniques.

Note that this approach assumes that our clustering algorithm will perform as well as the human classifiers who created the reference clusters. If the clustering process does not produce clusters that are 'correct' as determined by the human classifiers, this may impact the performance of our classifier.

In this chapter we discuss how we create the classification model. We start with the useful features that represent a cluster and the normalization of the data that is available to us. Additionally we need to determine which learning method to employ and configure it to optimize its performance.

7.1 Feature Selection

The properties of a cluster, mentioned in section 4.2.2, serve as an input for the features used for learning. Simply using all properties as features is not very useful since most properties do not directly contribute to the problem of determining the type of content and are very contradictory among different webpages, i.e., most properties are not suitable for a cluster representation, caused by the dynamic nature of webpage design. We therefore interpret the properties in a different way so that they add to the correct and useful representation of a cluster. This results in a list of features we deem relevant for the learning methods we will employ.

Standardization Consistency is an important aspect of the features. If we take, for example, the horizontal position of a cluster, it is necessary to view this value in the context of the webpage. If the cluster is located 50 pixels from the left side of a webpage that is 100 pixels wide, we cannot compare this to a webpage with a width of 1000 pixels. To keep the data in a consistent form, it needs to be normalized. In the case of horizontal location, we can express it as a fraction of the total website width to provide a new, invariant, feature.

A trade-off with this normalization of features is that by generalizing in advance, we might exclude specific information that might also positively contribute to the learning problem. In most cases however the advantages of preprocessing are obvious.

Cluster features The visual characteristics of a cluster include the background color, position and dimension. Note that we deal with the font color as a text property, not as a visual property. A property like background color by itself does not tell us anything about the particular cluster, but in a more global context it might give us more useful information. For example, if a color of a cluster is different from the color of surrounding clusters, it is likely that it has been given a higher priority by the designer to make it more noticeable. We can also measure this in the form of a fraction of the surface of that background color against all surfaces with other colors. When this outcome is lower, the cluster most likely has a higher priority. Like this example we must also try and relate the other properties to a more global context in order to make them more useful for classification.

Dimension is another property that differs per webpage. To be a correct representation, this value must be normalized, which can be done in different ways. One is to relate the cluster size to the size of the entire webpage. However, also the dimensions of webpages differ significantly from each other, so taking the ratio of a cluster size and its page size does not guarantee a representation we can count on. Another option is to relate the cluster size to the size of other clusters on the webpage, which will give us a better representation of its size. We therefore take the cluster/average cluster ratio to represent the size feature.

Position corresponds to the starting location of a cluster, measured from the top and the left of the webpage. It was already mentioned that we cannot rely on the sizes of webpages, so we will indicate the position of a cluster by taking its center and expressing it as a fraction of the webpage size.

Classification is performed after the clustering operation is finished. It is possible that the structure of the cluster contains valuable information for classification. Representing the structure of a cluster, however, is not a trivial operation. To simplify things, we take the number of elements contained in the cluster, which we will also use as a feature.

Textual features Visually, not much can be said about the text that is contained in a cluster. The number of lines, for example, cannot be derived from the information we have. Also, using some sort of linguistic analysis restricts us to a particular language, which is something we want to avoid. One thing we can use is the number of words, to indicate the amount of text that is contained in a cluster. For reasons already mentioned in the previous paragraph, we take the wordamount measure to be the ratio of the number of words in the cluster and the average number of words in clusters on that webpage.

For all other textual properties, like the font type and font size, we follow the same strategy as with the previously mentioned background color. By taking the fraction of surfaces with that property value.

Feature Overview Each cluster is represented by the following features:

- Background color: fraction of the surface with that background color.
- Cluster size: cluster size / average cluster size ratio.
- Cluster position as a fraction of the total width and height.

- Cluster elements: the number of elements in a cluster.
- Words: Number of words in cluster / average number of words ratio
- Text property: fraction of the surfaces of clusters that contain that property.

Note that we calculate these values using the aggregated values for the generated clusters, i.e., the calculated fractions will show some discrepancy compared to the actual fraction.

7.2 Learning Method

With the reference clusters at our disposal, we create a dataset of cluster features and their corresponding categories. The goal is to find a relation between the two in the form of a model, so that we can also apply it to clusters of which we do not know the category. Since the dataset can become quite large and devising such a mapping by hand is not much of an option, applying a machine learning technique seems appropriate here. Specifically, we are focusing on a classification algorithm and since we have a training set available, a supervised learning method suits this situation best.

Learning algorithm selection The class of supervised learning algorithms offers a large diversity of algorithms we can use. Looking at comparable studies, the popular algorithms MLP (Multi-Layer Perceptron [1]) and SVM (Support Vector Machine [7]) seemed to perform best, which is a reason for us to take them into account. Before we employ these algorithms, we also need to establish a baseline to check if these algorithms deliver an acceptable performance. To establish this performance baseline, we will make the assumption that features do not depend on each other and employ the Naive Bayes [9] and Logistic Regression [12] algorithms.

WEKA The learning algorithms we will use are provided by the WEKA[5] data mining software. WEKA is written in JAVA and offers an extensive API, enabling us to use it in our own software. Regarding classification algoritms, WEKA includes the most popular ones, among which the Multi-Layer Perceptron networks and Support Vector Machines, all fully configurable. Additionally WEKA offers a wide range of evaluation techniques that we will use to determine the performance of the generated models.

The results of the different learning algorithms can be found in section 8.4.

[5]http://www.cs.waikato.ac.nz/ ml/weka/

8 Results

From the different activities discussed in the previous chapters, we will now disclose the results. Some aspects of these results are recurring, such as comparing two sets of clusters or classifications, so we will first introduce our approach to these general operations and refer to them later on. The results of the different activities are listed in the order they appeared in this document, so we start with the user experiment and evaluate the reference clusters. Subsequently we will list the clustering and classification performance, followed by the combined performance of the complete system.

8.1 Evaluation in general

Evaluating clusters and their classifications is an operation frequently performed in this project. In this section we discuss our approach to these common evaluations.

8.1.1 Evaluating clusters

There are two inherent properties of our system that matter to this evaluation, which are also taken into account by the clustering methods. The first is that an element must always be contained in a cluster (single element clusters are allowed) and the second is that an element can only be contained in a single cluster (i.e., it can not be part of two clusters simultaneously). With these properties in mind, each webpage is essentially a set of elements, where a partition of this set is formed when these elements are clustered. Measuring cluster similarity then comes down to comparing the two partitions of the same set of elements. A measure known as the Rand index [15] can be used for exactly this purpose. The Rand index compares the relations of all pairs of elements between two partitions, where a relation corresponds to checking if both elements are located in a single cluster. A correct relation in this case is a relation (both elements contained in the same cluster: yes or no) that is equal to the relation in the reference webpage. The Rand index is given by the following fraction:

$$Rand\ index = \frac{Corresponding\ element\ relations\ in\ both\ webpages}{Total\ number\ of\ relations\ between\ elements} \tag{1}$$

A problem with the Rand index is that its outcome will always be very high. This is caused by the fact that webpages have a lot of elements, which are mostly not related, i.e., not contained in the same cluster. Since these non-relations are also taken into account, even with a very bad clustering performance, the rand index score will turn out to be very high. A more precise indication of clustering performance is given by the precision and recall metrics, which allow us to be more specific when analyzing the clustering performance. To make sure we do not encounter the same problem as we did with the Rand index, we only use the relations between elements in a cluster, the intra-cluster relations. A correct relation in the case is an intra-cluster relation that is both present in the reference cluster as well as the generated cluster. Precision and recall are defined as follows:

Precision In addition to the correct elements in the reference cluster, a generated cluster can have a surplus of elements. Since elements can only be present in a single cluster,

a consequence of this is that elements are missing in other clusters. We calculate the precision of a webpage using the following fraction:

$$Precision = \frac{\#Correct\ intra - cluster\ relations}{\#Intra - cluster\ relations\ in\ the\ generated\ clusters} \tag{2}$$

If precision is lower than one, this indicates that too much elements are clustered together, which is something we want to avoid.

Recall A generated cluster can contain less elements compared to the reference cluster, indicated by the recall measure. Note that this does not necessarily mean that the missing elements are abundant in other clusters, since single element clusters can also exist.

$$Recall = \frac{\#Correct\ intra - cluster\ relations}{\#Intra - cluster\ relations\ in\ the\ reference\ clusters} \tag{3}$$

A recall score lower than one indicates that there are still elements that need to be clustered together.

It is clear that we must find a balance between precision and recall. To indicate the total performance, the precision and recall measures can be combined, resulting in an F-measure:

$$F_\beta = (1 + \beta^2) \cdot \frac{(Precision \cdot Recall)}{(\beta^2 \cdot Precision) + Recall} \tag{4}$$

If both recall and precision are equally important, a β value of 1 can be used to balance them out. Since precision is, in some cases, very important to us, we can lower the β value, giving precision more weight.

The precision and recall share a very strong relation. Our aim is in the first place to obtain a high precision score, indicating that clusters are relatively small and not over-populated. A low precision score can be remedied by tweaking the parameters of the clustering algorithms to be more strict, but the effect of this is that the recall score will decrease, since less elements will be clustered together. To also increase the recall score we can focus in improving the clustering algorithms, or perhaps even add new clustering algorithms.

8.1.2 Evaluating classifications

Labeling is compared on the level of elements instead of clusters. This way we do not have to take cluster region similarity into account and can focus purely on the labels. Of course, classification performance is inextricably connected with the way elements are clustered. We will ignore this possible effect however, since our goal, ultimately, is not to optimize classification in isolation, but to optimize the combined system. To express the similarity between two different sets of labels, we calculate Cohen's Kappa [4] coefficient. This kappa coefficient is more robust than a simple agreement percentage calculation, since it also takes chance into account:

$$Cohen'sKappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{5}$$

Where Pr(a) is the normal agreement percentage and Pr(e) is the calculated chance that labels would agree based on the classification results.

8.2 User Experiment Results

The reference clusters created during the user experiment allow us to measure and optimize the performance of our clustering and classification methods. These clusters thus form a relatively important aspect of our research. To find out the reliability of the reference clusters and check our assumption that people can, in general, identify different segments in a webpage, a number of webpages were shared by all participants and clustered according to the definitions in section 5.1. Two aspects of the user experiment can be distinguished; the overlap of cluster regions and the overlap of the labels these regions were given. We measure clustering and classification apart from each other.

Two participants were involved in the creation of the reference clusters. Both participants were higher education students with average computer skills, which was enough to understand and perform their task. Each participant processed a total of 44 webpages, which included 10 shared webpages.

Participant consistency

The ten webpages that were shared by the participants allow us to measure the inter participant agreement. We calculated the Rand Index and Cohen's Kappa for each shared webpage, listed in table 1. Both participants forgot to classify two of the pages, so we left these out and continued with eight webpages.

Website	#elements	Rand Index	Cohen's κ	Category Fraction
spitsnieuws.nl	115	1.000	0.831	0.913
trouw.nl	141	0.925	0.631	0.716
rtl.nl	155	0.950	0.513	0.659
leidschdagblad.nl	201	0.995	0.251	0.378
telegraaf.nl	247	0.985	0.185	0.352
elsevier.nl	315	0.786	0.346	0.537
parool.nl	376	0.967	0.495	0.604
computable.nl	461	0.942	0.054	0.128
Weighted Mean	-	0.935	0.329	0.460

Table 1: The agreement between two participants for each of the shared webpages.

Table 1 only gives an outline of the classification results. With regard to clustering, we express this with a weighted F_1 score, which was 0.743. The following confusion matrix shows the agreements of classifications between the two participants:

-	Main	Add.	Link	Nav.	Adv.	Other	Uncl.	Total
Main Text	0	68	0	5	0	1	0	74
Add. Text	0	209	115	0	0	12	0	336
Link Group	0	0	354	91	0	27	10	482
Nav. Menu	0	0	7	148	0	0	12	167
Advertisement	0	7	28	0	40	10	41	126
Other Group	0	25	158	25	0	168	61	437
Unclassified	0	41	251	0	1	89	7	389
Total	0	350	913	269	41	307	131	926

Table 2: The confusion matrix of the classifications by the two participants, where the columns and rows list the results for participant 1 and 2, respectively. The total in the right bottom corner is that of the similarly classified elements for all categories.

The confusion matrix shows some disagreement between participants. Before we continue, some adjustments are needed to make sure that the reference clusters do not contain too much conflicts before they are used for training and evaluation. How we remove these inconsistencies is mentioned in the discussion section (chapter 9). This update operation results in a set of clusters, from 77 webpages (since both participants forgot the same webpage, we left this out), with the following characteristics:

-	#clusters	#elements
Main Text	39 (0.015)	821 (0.051)
Add. Text	580 (0.226)	2132 (0.133)
Link Group	718 (0.280)	7295 (0.454)
Nav. Menu	135 (0.053)	1435 (0.089)
Advertisement	86 (0.034)	683 (0.042)
Other Group	540 (0.211)	3184 (0.198)
Unclassified	467 (0.182)	548 (0.034)
Total	2565 (1)	16098 (1)

Table 3: Properties of the reference clusters. The total number of clusters here includes the unclassified clusters. Without the unclassified clusters the total number amounts to 2098.

These 2098 reference clusters form the objective for our clustering strategy and serves as a training set for the creation of the classification model.

8.3 Clustering Results

The reference clusters serve as a basis that allows us to compose a strategy from the two cluster algorithms we designed. Before we look at the complete strategy, table 4 gives us an impression of the scores obtained prior to clustering. As we would expect, precision is very high since every single element must be present in the reference clusters, while recall is very low because most elements are missing in the (single element) clusters when we compare them to reference clusters.

Recall	0.039
Precision	0.980
F_1 Score	0.075

Table 4: Precision and recall before applying the cluster methods.

Since the pattern clustering algorithm cannot deal with too much repetitive structure, we are forced to always start with a block clustering iteration. The block clustering method requires us to optimize the LCA distance, similarity distance and minimal cluster size parameters. If we simply loop through all possible combinations of these parameters, the values shown in table 5 obtain the highest F_1 score.

Maximal Similarity distance	3
Maximal LCA distance	8
Minimal cluster size	0
Recall	0.774
Precision	0.658
F_1 Score	0.712

Table 5: Best performing block clustering configuration, based on F_1 score.

The above mentioned model parameters were fitted to the complete training set. To verify generalization of the clustering algorithm, we performed a 6-fold cross validation on this first block clustering iteration. The optimized parameter values for every fold were equal to the ones listed in table 5. This makes it very likely that the clustering algorithm will perform similarly on unseen webpages.

The graph in figure 13 displays the effect of the LCA and similarity distance on the F_1 score.

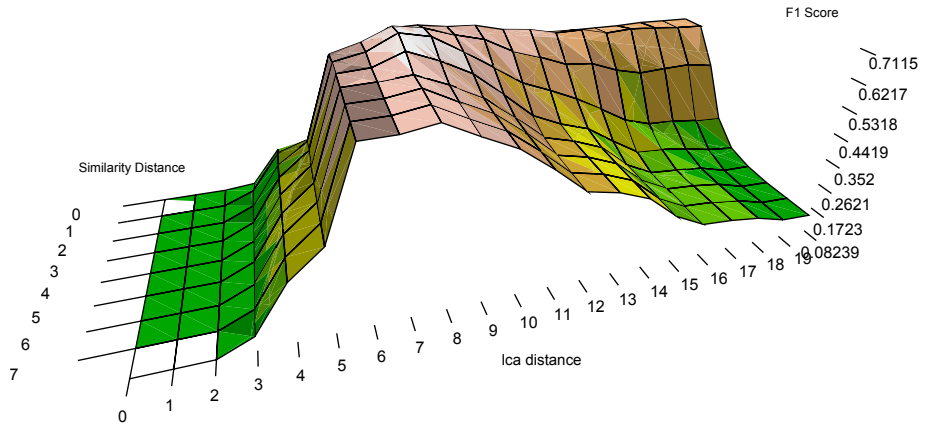

Figure 13: The effect of the maximum LCA and similarity distance on the F_1 score.

We can say a few things about the graph in figure 13:

- The similarity distance has a smaller range than the LCA distance. This is because we deal with a limited number of features that make up the similarity distance by

calculating the hamming distance. Allowing a distance of more than six simply does not make a difference anymore.

- The LCA distance on the other hand is bound by a larger range, since it is related to the largest LCA distances on a webpage. This stabilizes around a distance of twenty as we can see in the graph.

- If we look at both variables in isolation, we find that the similarity measure enjoys most freedom. When we would remove the LCA distance (set it to infinite), a low maximum similarity distance is enough to obtain a relatively high F_1 score. By contrast, the LCA distance would attain a high F_1 score only within a very small range of values (6 - 9).

Greedy approach

We first try to build a strategy using the greedy approach. For each iteration of clustering we keep clustering with the same parameter values until no new clusters can be formed (referred to as re-clustering). Table 6 shows the results of this approach with two iterations, where the first is only the required block clustering. In each iteration, all possible combinations of parameters were evaluated to find the optimal configuration.

Iteration	Type	Sim Dist	LCA Dist	F_1	Precision	recall
1	Block	3	8	0.712	0.658	0.774
2	Block	0	9	0.713	0.657	0.778
2	Pattern	0	2	0.712	0.658	0.774

Table 6: Results from the greedy clustering approach, combining the best methods from each iteration based on the the F_1 score.

Conservative approach

The first step to take for the conservative approach is to execute the block clustering algorithm a single time. Not using re-clustering like we did in the greedy approach. Combined with the $F_{0.25}$ measure this gives us a configuration of block clustering that forms a solid basis for the future cluster iterations to work with. The results of this approach are listed in table 7. Each iteration follows the best configuration from the previous ones, based on the $F_{0.25}$ score. As with the greedy approach, to find the optimal configuration, all possible combinations of parameters were evaluated.

Iteration	Type	Sim Dist	LCA Dist	$F_{0.25}$	F_1	Precision	recall
1	Block	2	8	0.883	0.605	0.941	0.446
2	Pattern	5	4	0.886	0.626	0.938	0.469
2	Block	2	4	0.885	0.639	0.933	0.486
3	Pattern	0	5	0.887	0.634	0.936	0.479
3	Block	0	3	0.886	0.629	0.937	0.474

Table 7: Results from the conservative clustering approach, combining the best methods from each iteration based on the the $F_{0.25}$ score.

8.4 Classification Results

Classification is performed after the clustering process is finished. To find the most suitable and best performing classification model, we evaluate different classification techniques. The reference clusters serve as a basis for the training samples used for each classification model, which describes a mapping between the cluster features and the categories we defined. Note that to create these models, we use the reference clusters created in the experiment. In this subsection we will only deal with classification, the next subsection will show the combined results that follow from applying the best performing classification model to the clusters created by our best performing clustering strategy.

To create the different classification models, WEKA is used. After generating the training set in a WEKA compatible format, the stand-alone WEKA application is used to create the classification models and evaluate them, which is done using 6-fold cross-validation. The dataset was comprised of 2098 training instances, with 15 features describing the clusters.

Multi-Layer Perceptron We start with the training of a Multi-Layer Perceptron network. We configured it with 10 hidden nodes and 6 output nodes (one for each category). Backpropagation is used for learning with a momentum of 0.2 and learning rate of 0.3. During each fold, 500 iterations are used for training, yielding the following results:

Correctly Classified Instances	1533	73.07 %
Incorrectly Classified Instances	565	26.93 %

Table 8: Results of the Multi-Layer Perceptron model created with the reference cluster dataset.

-	Main	Add.	Link	Nav.	Adv.	Other	Total
Main Text	29	6	4	0	0	0	39
Add. Text	2	486	54	2	0	36	580
Link Group	2	36	575	10	8	87	718
Nav. Menu	0	0	7	116	0	12	135
Advertisement	2	14	34	0	3	33	86
Other	2	52	146	11	5	324	540
Total	37	594	820	139	16	492	1533
Correct	0.744	0.838	0.801	0.859	0.035	0.6	0.731

Table 9: Confusion matrix displaying the mistakes of the Multi-Layer Perceptron model. The columns and rows represent the outcome from the model and the correct reference clusters, respectively. The bottom right corner shows the total number of correctly classified elements.

Support Vector Machine The Support Vector Machine classifier used a polynomial kernel, yielding the following results:

Correctly Classified Instances	1359	64.78 %
Incorrectly Classified Instances	739	35.22 %

Table 10: Results of the Support Vector Machine model created with the reference cluster dataset.

-	Main	Add.	Link	Nav.	Adv.	Other	Total
Main Text	27	7	5	0	0	0	39
Add. Text	0	492	26	1	0	61	580
Link Group	0	129	484	7	0	98	718
Nav. Menu	0	4	17	86	0	28	135
Advertisement	0	22	31	0	0	33	86
Other	0	95	149	26	0	270	540
Total	27	749	712	120	0	490	1359
Correct	0.692	0.848	0.674	0.637	0.0	0.5	0.648

Table 11: Confusion matrix displaying the mistakes of the Support Vector Machine model. The columns and rows represent the outcome from the model and the correct reference clusters, respectively. The bottom right corner shows the total number of correctly classified elements.

Naive Bayes To get an impression of a baseline for the dataset, we use the Naive Bayes Classifier:

Correctly Classified Instances	1164	55.48 %
Incorrectly Classified Instances	934	44.52 %

Table 12: Results of the Naive Bayes classifier created with the reference cluster dataset.

-	Main	Add.	Link	Nav.	Adv.	Other	Total
Main Text	38	0	0	0	1	0	39
Add. Text	17	497	29	10	16	11	580
Link Group	60	199	385	11	17	46	718
Nav. Menu	3	3	26	96	1	6	135
Advertisement	0	32	26	1	13	14	86
Other	7	188	139	50	21	135	540
Total	125	919	605	168	69	212	1164
Correct	0.97	0.857	0.536	0.711	0.151	0.25	0.555

Table 13: Confusion matrix displaying the mistakes of the Naive Bayes classifier. The columns and rows represent the outcome from the classifier and the correct reference clusters, respectively. The bottom right corner shows the total number of correctly classified elements.

Logistic regression Another, relatively simple, algorithm we use to indicate a baseline is Logistic Regression.

Correctly Classified Instances	1465	69.83 %
Incorrectly Classified Instances	633	30.17 %

Table 14: Results of the Logistic Regression model created with the reference cluster dataset.

-	Main	Add.	Link	Nav.	Adv.	Other	Total
Main Text	29	8	2	0	0	0	39
Add. Text	4	498	32	0	0	46	580
Link Group	4	43	538	9	8	116	718
Nav. Menu	0	0	3	114	0	18	135
Advertisement	0	17	32	0	2	35	86
Other	2	85	134	30	5	284	540
Total	39	651	741	153	15	499	1465
Correct	0.744	0.859	0.750	0.844	0.023	0.526	0.698

Table 15: Confusion matrix displaying the mistakes of the Logistic Regression model. The columns and rows represent the outcome from the classifier and the correct reference clusters, respectively. The bottom right corner shows the total number of correctly classified elements.

8.5 Combined Results

Now that we have pinned down a clustering strategy and a classification model, both can be used to segment and classify webpages. To measure the performance of clustering and classification combined, we will use the webpages from which the reference clusters were created and analyze the generated clusters in a similar fashion compared to our analysis of the reference cluster consistency in section 8.2.

From the results in the previous subsection we can make up that the MLP (Multi-Layer Perceptron) classification model performs best. We will now combine the best strategy from both the greedy and conservative approach with the MLP classification model and and display the results in table 16 and table 17, respectively. Since we already listed the clustering results in section 8.3, we only list the classification results. Again, we compare the cluster classifications on the level of elements.

-	Main	Add.	Link	Nav.	Adv.	Other	Uncl.	Total
Main Text	525	306	318	0	3	75	55	1282
Add. Text	65	928	470	4	126	183	75	1851
Link Group	114	588	5619	109	205	1271	126	8032
Nav. Menu	12	16	80	1211	13	204	58	1594
Advertisement	0	0	18	0	27	48	0	93
Other	105	294	790	111	309	1403	234	3246
Unclassified	0	0	0	0	0	0	0	0
Total	821	2132	7295	1435	683	3184	548	9713
Correct	0.639	0.435	0.770	0.844	0.040	0.441	0.0	0.603

Table 16: Confusion matrix displaying the mistakes of the strategy based on the greedy approach. The rows and columns represent the outcome of the strategy and the reference clusters, respectively. The totals in the right bottom corner are that of the similarly classified elements.

-	Main	Add.	Link	Nav.	Adv.	Other	Uncl.	Total
Main Text	276	141	532	24	0	28	26	1027
Add. Text	379	1466	2395	5	232	627	133	5237
Link Group	42	83	3091	250	65	770	57	4358
Nav. Menu	0	1	37	987	0	67	22	1114
Advertisement	0	0	5	0	27	7	0	39
Other	124	441	1235	169	359	1685	310	4323
Unclassified	0	0	0	0	0	0	0	0
Total	821	2132	7295	1435	683	3184	548	7532
Correct	0.336	0.688	0.424	0.688	0.040	0.529	0.0	0.468

Table 17: Confusion matrix displaying the mistakes of the strategy based on the conservative approach. The rows and columns represent the outcome of the strategy and the reference clusters, respectively. The totals in the right bottom corner are that of the similarly classified elements.

9 Discussion

The results will be interpreted in the order they appeared in the preceding results chapter.

User Experiment

The user experiment was set up to gather data that served as examples for clustering and classification. Additionally we wanted to find out how participants would segment and classify a collection of webpages using a set of clearly defined content types commonly found on webpages. Because it was a lot of work for participants to annotate the webpages, we combined both activities and served a single set of webpages for both purposes.

To determine the value of this user generated data, we measured the agreement of clusters and classifications between two participants. The results in section 8.2 show that there is a notable difference between the overlap in clustering and the overlap in classification. Clustering, with a weighted F_1 score of 0.743, shows some discrepancy between the clusters of both participants, which mostly stems from the difference in specificity. Where one participant would neatly classify a series of adjacent equivalent clusters individually, the other would select them as a whole (probably to save time). While this does not have any effect on classification results, since all elements in those clusters can still be labeled with the same category, it does largely account for difference in clustering.

With regard to classification results, the overlap shows a poor result. As we expected, the rand index looks very positive, but does not represent the cluster overlap very well. When we do not take chance into account, and look at the percentage of overlap, about 46% (weighted mean) of the elements have been classified with the same category. The confusion matrix in table 2 allows us to make a few observations concerning classification agreement:

1. Participant one (columns) did not use the Main text category. Instead these clusters were classified as being an Additional text. Inquiry showed that this participant completely forgot about this class.

2. Participant two (rows) forgot to classify part of a page, which accounts for the high number of unclassified elements.

3. Where participant two often used the other group, this was often classified as a link group by participant one.

4. There was some confusion between the link group and navigation classes, where participant one tended to classify more elements as navigation while participant two saw these as link groups.

To make sure the reference clusters could serve as proper training data, we used these observations to minimize the discrepancies. In most cases this was done in an objective fashion, correcting the differences mentioned above. In a few cases however, we also had trouble making a distinction between categories, making the adjustments rather arbitrary, or subjective. This was primarily the case with small texts that could be classified as an additional text or an advertisement and can be explained by the fact that most advertisements are created to seamlessly blend in to webpages and are therefore hardly

distinguishable from other content. Apart from the difference in the specificity of clustering, we can state that the participants had a general consensus of cluster regions on the webpages, since they could consistently keep them apart. Consequently we can say that clustering had a minimal effect on the classification results.

Given the observations we made, based on the confusion matrix in table 2, it is likely that our descriptions of the categories fell short, even though we already revised them based on observations with other participants. This is confirmed by one of the participants, who stated that there is simply too much variety of information on webpages, sometimes making it very hard to distinguish between categories. It seems that by making the category descriptions more explicit, participants would have less trouble distinguishing between categories. However, this would also leave less room for interpretation by the participants, which we also needed to set up this user experiment. This was the result of the trade-off we had to make for a single user experiment with two purposes. The fact that the participants first had to be made familiar with the categories we defined already indicates that the categories themselves are not inherent to the understanding of a webpage structure by a perceiver. The clusters, on the other hand, were consistently kept apart, where we expect the category definitions to have contributed to the clustering process by indicating the specificity of the clusters. We can state that our assumption of visual discernibility of the segments of a webpage was correct. We might have gotten a more complete picture if we involved more than the bare minimum of two participants but we found the current results to be sufficient for pointing out the discrepancies that our experiment produced.

Clustering

Intuitively we expected that a strategy created by the conservative approach would have less of a chance of ending up in a local optimum, because of the finer, more gradual improvements, and yield better results compared to the greedy approach. We were proven wrong by the results. Table 7 denotes that the conservative approach does leave more room for improvement compared to the greedy approach in table 6. However when we compare F_1 scores, the greedy approach does deliver better results, already in the first iteration. It seems that the pattern clustering algorithm hardly contributes to the clustering process. We expect this to be caused by the relatively small number of repetitive structures in the webpages that the pattern algorithm operates on. Most of the webpages contain a lot of rectangular structures, already clustered by the block clustering algorithm.

The most prominent reason for some sites to score low is the static LCA distance. An LCA distance that is too low will prevent structures from clustering, causing a low recall, while a value that is too high will cause a low precision because too much is clustered. Results may be improved by calculating an LCA distance threshold for each webpage independently, for example by relating it to the average distance on that webpage. LCA distance is however still structural information which we want to eliminate and replace with a boundary measure truly based on only visual aspects of the webpage.

The influence of the block clustering parameters (LCA and similarity distance) on the F_1 score is depicted in figure 13. This graph clearly indicates that a good performance is attainable with an LCA distance within the range of 6 - 9, irrespective of the similarity distance. However, also taking the similarity distance into account does help increase performance a bit more. If we take the similarity distance in isolation, it performs a

little under the LCA distance, indicating that the LCA distance is the more important parameter. When the two parameters both get too high, the performance collapses, caused by the excessive clustering which decreases precision.

The less than perfect clustering performance has its effect on the classification results. Since the classification model was trained on the reference clusters, clusters deviating from them are likely to decrease classification performance.

Classification

Of the Naive Bayes and Logistic Regression classifiers, the best performance was obtained by using the Logistic Regression function, with a success rate of almost 70%. The Naive Bayes model fell behind with approximately 55% of the samples correctly classified. From the confusion matrices in section 8.4 we can see that Naive Bayes had trouble with most of the categories, in particular with 'advertisement' and 'other group', with an correctness score of 0.151 and 0.25 respectively. These two categories also obtained the worst performance with Logistic Regression, where 'advertisement' scored 0.023 and 'other group' 0.526. Other than that, Logistic Regression scored remarkably well.

Of the more complex models, the Multi-Layer Perceptron model (73% correctly classified) beat the Support Vector Machine model (65%). As with the other classifiers, the 'advertisement' and 'other group' scored lowest. Note that these classification models were created using the (perfect) reference clusters.

Combined

When we combine our clustering strategies with the MLP classification model we find the results in section 8.5. Combined with the strategy from the greedy approach (clustering F_1 score of 0.713), over 60% of the elements were correctly classified, while the conservative approach strategy (clustering F_1 score of 0.634) attained less then 47%. This can partly be explained by the fact that the greedy approach seems to favor the link group more than the conservative approach, leveraging the high weight it has in the complete score. The difference between the two clustering approaches indicates that the clustering performance definitely has a big impact on classification performance. We do expect to see similar scores on unseen webpages, since n-fold evaluation for both clustering and classification was used to verify generalization. In general, we cannot say too much about this performance, since comparable studies dealt with completely different datasets. If we take the most obvious baseline of classifying all instances with the largest occurring category in the greedy approach, this would result in a score of 45,3%, which our method seems to seriously outperform. Looking more closely at the individual categories we see big differences between the two approaches in the first four categories. The link category has a lot of impact since it makes up nearly 45% of all elements, but other than that the main text and navigation categories also show a remarkably better score with the greedy approach. The only noteworthy exception is the additional text category scoring much higher with the conservative approach.

10 Conclusion

The task we set out to do was to create a general purpose method that automates the recognition of common segments in webpages using only visual aspects of the webpage. With the help of different principles from the Gestalt theory, we devised two clustering algorithms and combined these in a cluster strategy that was fitted to a set of sample webpages created in a user experiment.

The user experiment made clear that people consistently distinguish the different sections on webpages, following a set of section definitions. However, the classifications of these segments show less agreement. We ascribe this discrepancy to the nature of human sight. Where principles of Gestalt theory account for the ability to keep different segments apart from each other, there is no intrinsic understanding of the definitions for such segments. Although our segment definitions were very general and present on most webpages, they first had to be studied before participants could apply them. Additionally, the nature of design provides for an endless variety in segment designs, making it very hard to capture them in exact descriptions, thus leaving a lot of room for interpretation by the participants. Our assumption that these common segment types were easily recognizable by the participants therefore did not completely hold.

The optimal configuration of our system provided just over 60% of the text elements on the webpages with the correct label, where we did not take images, flash or other objects containing information into account. This result was obtained with a classification model that was trained using the reference clusters. We expect the results to be better when clusters generated with our clustering algorithms are used as training data for the classification model. This would however require human participants to manually categorize them. The usefulness of our results depend on the segments we are looking for. Segments like advertisements are hardly recognizable when semantics are not taken into account, while navigational segments can be extracted with a lot more certainty. When specific types of information need to be extracted from webpages, structure-specific methods are likely to outperform a general purpose method we proposed in this document. Our research does however indicate that it is possible to use principles from human psychology to create a general purpose segmentation method based on aspects that are not inherent to the medium used. An important result is that clustering based solely on visual properties is possible, excluding the need for semantic or meta information about the elements from the information source. And while this method may not deliver perfect clustering or classification, it can be used to give a rough estimation of the clusters to be found on webpages. Another important aspect of our method is the applicability. With our current method, the performance depends on the size and structure of the webpage. Real-time application of the process can be ensured by imposing restrictions on the clustering process, which was not needed for the webpages we encountered.

The open character of our method allows for many optimizations and extensions, where future improvements can mostly be found in the clustering methods. The two clustering methods we devised incorporated certain principles from gestalt theory. These principles can be interpreted in many different ways. It would be interesting to see how the different interpretations or uses of the laws, such as the laws of symmetry or common fate, would influence the clustering performance. With our methods we expect the biggest improvement to be parameter values that are dynamically evaluated on the basis of a webpage, instead of using threshold values for all webpages. Also, we assumed that the visual

properties used for similarity measurement did not interfere with each other. It might however be the case that these are not as independent as we thought, which may have an impact on the results. Concerning classification, in our project this only had a descriptive function. It can however also fulfill a supporting role for clustering, by indicating content types during the clustering process. This will require that another classification model needs to be created, since the clustering methods typically deal with smaller segments. To speed up the project, the single deviation from the visual only requirement was the usage of distances between elements in the structure of the webpage. The use of this structural information was required, because it represents important visual separators between elements and was based on the assumption that the webpage code structure reflects its visual structure. This specific type of structural information can be replaced by a distance measure that is purely visually oriented. For instance by analyzing a screenshot of the webpage and looking for visual cues such as background color or lines. Another solution would be to use a VIPS like visual detection algorithm to find these separators.

One of the principles of our vision based general purpose method is that it does not need tailoring for specific webpages and can, in theory, also be used with other information sources, such as PDF files. A condition however is that all text elements with their properties need to be extracted first, which is a process that differs for each type of information source. Subsequently we deem it necessary to first optimize the clustering and classification parameters for the information source, before segmentation can take place, which signifies a dependence. Irrespective of this dependence, we think this general purpose vision based segmentation system shows promise for the future, especially if the system is optimized for a specific purpose.

References

[1] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[2] S. Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 33–42, New York, NY, USA, 2006. ACM.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[4] J. Carletta. Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.*, 22:249–254, June 1996.

[5] J. Chen, B. Zhou, J. Shi, H. Zhang, and Q. Fengwu. Function-based object model towards website adaptation. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 587–596, New York, NY, USA, 2001. ACM.

[6] I. Chibane and B.-L. Doan. A web page topic segmentation algorithm based on visual criteria and content layout. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 817–818, New York, NY, USA, 2007. ACM.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.

[8] M. Ester, H. peter Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[9] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29:131–163, November 1997.

[10] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. Dom-based content extraction of html documents. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 207–214, New York, NY, USA, 2003. ACM.

[11] R. W. Hamming. Error detecting and error correcting codes. *Bell Syst. Tech. J.*, 29:147–160, 1950.

[12] D. W. Hosmer and S. Lemeshow. *Applied logistic regression*. Wiley, 2 edition, 2000.

[13] R. R. Mehta, P. Mitra, and H. Karnick. Extracting semantic structure of web documents using content and visual information. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 928–929, New York, NY, USA, 2005. ACM.

[14] P. T. Quinlan and R. N. Wilton. Grouping by proximity or similarity? competition between the gestalt principles in vision. *Perception*, 27:417–430, 1998.

[15] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association (JASA)*, 66:846–850, 1971.

[16] R. Romero and A. Berger. Automatic partitioning of web pages using clustering. In *Mobile HCI, volume 3160 of Lecture Notes in Computer Science*, pages 388–393. Springer, 2004.

[17] V. Snasel. Gui patterns and web semantics. In *CISIM '07: Proceedings of the 6th International Conference on Computer Information Systems and Industrial Management Applications*, pages 14–19, Washington, DC, USA, 2007. IEEE Computer Society.

[18] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning important models for web page blocks based on layout and content analysis. *SIGKDD Explor. Newsl.*, 6(2):14–23, 2004.

[19] M. Wertheimer. Untersuchungen zur lehre von der gestalt. ii [principles of perceptual organization]. *Psychol Forsch*, 4:301–350, 1923.

[20] M. Wertheimer. Laws of organization in perceptual forms. In *A Source Book of Gestalt Psychology*, pages 71–88, 1955.

[21] C. Wu, G. Zeng, and G. Xu. A web page segmentation algorithm for extracting product information. *International Conference on Information Acquisition*, 0:1374–1379, 2006.

[22] P. Xiang, X. Yang, and Y. Shi. Effective page segmentation combining pattern analysis and visual separators for browsing on small screens. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 831–840, Washington, DC, USA, 2006. IEEE Computer Society.

[23] Y. Yang and H. Zhang. Html page analysis based on visual cues. In *ICDAR '01: Proceedings of the Sixth International Conference on Document Analysis and Recognition*, page 859, Washington, DC, USA, 2001. IEEE Computer Society.

[24] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 11–18, New York, NY, USA, 2003. ACM.

A Reference Cluster webpages

The following websites were used to create the reference clusters. For each websites we collected the home (or landing) page, and a detail page listing a news item.

- www.dft.nl
- www.elsevier.nl
- www.rtlz.nl
- www.telegraaf.nl
- www.trouw.nl
- www.volkskrant.nl
- www.frieschdagblad.nl
- www.iex.nl
- www.mkbnet.nl
- www.nieuws.nl
- www.nrc.nl
- www.omroepflevoland.nl
- www.parool.nl
- www.zibb.nl
- www.mt.nl
- www.wsj.com
- www.nyt.com
- www.trends.be
- www.rtlnieuws.nl
- www.metronieuws.nl
- www.spitsnieuws.nl
- www.adformatie.nl
- www.emmen.nl
- www.europarl.europa.eu
- www.amsterdam.nl
- www.computable.nl
- www.sp.nl
- www.leidschdagblad.nl
- www.scp.nl
- www.at5.nl
- www.distrifood.nl
- www.eindhovensdagblad.nl
- www.emerce.nl
- www.foodholland.nl
- www.gelderlander.nl
- www.gooieneemlander.nl
- www.leeuwardercourant.nl
- www.nhd.nl
- www.tweakers.net